VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NEURONOVÉ SÍTĚ A PREDIKCE ČASOVÝCH ŘAD

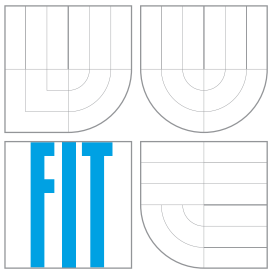
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ SVITÁK

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NEURONOVÉ SÍTĚ A PREDIKCE ČASOVÝCH ŘAD

NEURAL NETWORKS AND PREDICTION OF TIME SERIES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ SVITÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PATRIK PETŘÍK

BRNO 2010

Abstrakt

Bakalářská práce se zabývá neuronovými sítěmi používanými k predikci časových řad. Jde zejména o dopřednou neuronovou síť s algoritmem učení zpětného šíření chyby, neuronovou síť s radiálními bázovými funkcemi a neuronovou síť vyššího řádu. Jsou prováděna měření různých parametrů těchto sítí a jejich srovnání. Testování je prováděno na časových řadách historických cen finančních trhů. Stručně jsou zmíněny další typy neuronových sítí a jiné metody predikce finančních trhů.

Abstract

Bachelor's thesis studies neural networks that are used for time serie prediction. Particularly it is feedforward neural network with backpropagation learning algorithm, neural network with radial basis functions and higher order neural network. Various parameters of these networks are tested and comparised. Testing is performed on time series of historical prices of financial markets. Other neural networks and other forecasting methods of financial markets are mentioned briefly.

Klíčová slova

neuronové sítě, predikce, časové řady, finanční trhy

Keywords

neural networks, prediction, time series, financial markets

Citace

Jiří Sviták: Neuronové sítě a predikce časových řad, bakalářská práce, Brno, FIT VUT v Brně, 2010

Neuronové sítě a predikce časových řad

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Patrika Petříka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Sviták
17. května 2010

Poděkování

Děkuji Ing. Patriku Petříkovi za cenné rady a připomínky, ale hlavně za výborný přístup.

Obsah

Kapitola 1

Úvod

V posledních desetiletích v souvislosti s rozvojem vědy a techniky je možné předpovídat některé jevy s poměrně velkou mírou úspěšnosti. Každý zná například předpověď počasí. Je založena na složitých matematických simulačních modelech. Úspěšnost předpovědi počasí na další den je poměrně vysoká, ale s výhledem do dalších dnů přesnost předpovědi klesá. Tato práce se bude zabývat předpověďmi, které využívají neuronové sítě.

Teorie a návrh umělých neuronových sítí významně pokročily v posledních 25 letech. Mnoho z tohoto pokroku má dopad na zpracování časových řad. Oproti tradičním lineárním modelům jsou neuronové sítě nelineárními modely. Mají schopnost učení se s učitelem i bez něj, umí univerzálně aproximovat funkce, což je dělá velmi vhodnými pro řešení složitých problémů v oblast zpracování signálů. Umělé neuronové sítě jsou statistické modelovací nástroje, které mají široké pole aplikací, včetně predikce časových řad. Předpovídat je možné spotřebu energie, riziko lékařského zákroku, ekonomické a finanční předpovědi, případně předpovědi chaotických časových řad. V mnoha případech tyto modely dosahovaly lepších výsledků než jiné přístupy [4].

V oblasti předpovědí dosáhly modely založené na neuronových sítích určitých úspěchů. Například Moody, Levin a Rehfuss [9] v roce 1993 přesvědčivě demonstrovali převahu předpovědních technik založených na neuronových sítích, když předpovídali některé ukazatele ekonomiky USA. Hutchinson, Lo a Poggio [5] ukázali, že jejich neuronová síť založená na cenovém modelu překonává Black-Scholesův model v denním pohybu cen na indexu S&P. Giles, Lawrence a Tsoi [3] našli významné předpovědní schopnosti v denním vývoji devizových burz díky svým modelům, které byly založeny na neuronových sítích [4].

V komerční praxi je ovšem situace jiná. Přestože mnoho organizací vyvíjelo úsilí o využití technologií založených na neuronových sítích, tak jen malému množství společností se podařilo uspět. Přitom právě největší sponzoři výzkumu neuronových sítí jsou finanční společnosti. Tyto korporace zkoumají všechny nové možnosti, jak zvyšovat svůj zisk. Úspěšné společnosti často investovaly obrovské zdroje do testování svých neuronových sítí. Tyto výsledky ovšem byly dobré jen pro jejich konkrétní potřeby. Mnoho těchto řešení je proto v současnosti uzavřených. Současně je nabízeno poměrně velké množství placených programů, které se predikcí pomocí neuronových sítí zabývají [6].

Práce je členěna do několika kapitol. První kapitoly jsou věnované vybraným typům sítí, jejich strukturám a učení. Budou popsány dopředná neuronová síť s algoritmem učení backpropagation (2) a neuronová síť s radiálními bázovými funkcemi (3). Ostatními typy jako sítí, kterými jsou například Jordanova, Elmanova nebo neuronová síť vyššího řádu, se tato práce bude zabývat jen stručně (4), protože pro predikci finančních trhů se používají dle literatury méně často [6]. Kapitola věnující se predikci (5) se zabývá modelováním časových

řad, predikcí časových řad a také krátce zmiňuje jiné metody predikce než pomocí neuronové sítě. V kapitole Implementace jsou zmíněny některé simulátory neuronových sítí použitelné pro predikce, ale hlavně je popsána vlastní aplikace, s níž byly prováděny experimenty. V kapitole experimentů (7) jsou provedena měření vybraných parametrů neuronových sítí, spolu s porovnáním výsledků pomocí tabulek a grafů. Na závěr (8) práce je provedeno zhodnocení výsledků a diskuze dalších možností vývoje. V práci je ještě jednoduchý manuál k aplikaci a popis obsahu CD.

Cílem práce je měření parametrů neuronových sítí a implementace aplikace, která bude tato měření umožňovat.

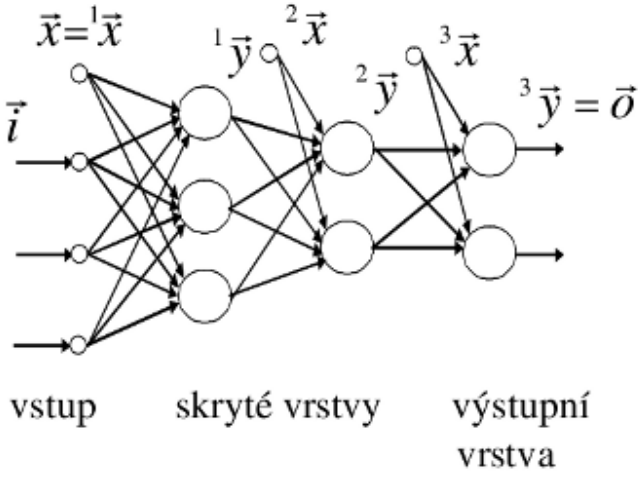
Kapitola 2

Neuronová síť backpropagation

Tato kapitola se zabývá popisem neuronové sítě backpropagation (dále BP), popisem jejích jednotlivých částí, z nichž se skládá. Zabývá se také algoritmy učení a parametry, které ovlivňují kvalitu učení. Pro učení se používá algoritmus zpětného šíření chyby, odtud plyne poněkud nepřesně i název této sítě. Některé sekce, týkající se například přípravy dat, se vztahují i k dalším typům neuronových sítí.

2.1 Popis sítě

Je to dopředná neuronová síť, protože signál se šíří jen jedním směrem. Někdy se také označuje jako vícevrstvá perceptronová síť (multilayer perceptron – MLP). S dostatečným počtem neuronů skryté vrstvy je použitelná jako univerzální prostředek pro aproximaci funkcí [4]. Tato neuronová síť se skládá z několika druhů vrstev, schéma struktury je na obrázku 2.1.



Obrázek 2.1: Dopředná neuronová síť, obrázek převzat z [13]

Vstupní vrstva slouží jen k distribuci vstupních hodnot do všech neuronů další vrstvy. Počet vstupů neuronové sítě je volitelný a záleží na povaze úlohy. Následuje obvykle jedna, zřídka více skrytých vrstev složených z neuronů schopných učení. Vrstva se nazývá skrytá, protože nejsou vidět hodnoty výstupů jejích neuronů. Výstupy jedné vrstvy jsou distribuovány všem neuronům následující vrstvy. Počet skrytých neuronů závisí na řešeném problému

a bude předmětem měření. Poslední vrstva je výstupní, složená pro účely predikce nejčastěji z jednoho neuronu. Použití více neuronů na výstupu nemá praktický význam, protože výstupy se vzájemně nemohou ovlivňovat a je jednodušší zpracování vzorků časové řady po jednom, než po dvojicích, případně jiných n-ticích [6].

2.2 Neuron

Umělý neuron je základním stavebním prvkem neuronových sítí. Oproti biologickému neuronu má jiné vlastnosti. Skutečný neuron pracuje na základě chemických reakcí a slabých elektrických signálů. Umělý neuron je matematický model a jeho činnost lze snadno popsat vzorcem, což je velmi výhodné pro zpracování na počítačích. Neuron má 1 až n vstupů a jeden výstup. Lze jej popsat rovnicí [13]:

$$y = f(u) = f(u(\vec{x})) \tag{2.1}$$

kde je
$\vec{x}$ vektor vstupních hodnot,
u bázová funkce,
f aktivační funkce

Existuje speciální případ umělého neuronu, který se nazývá perceptron. Je to nejjednodušší neuronová síť obsahující jeden neuron s prahovou aktivační funkcí a je používána jako jednoduchý lineární klasifikátor. Perceptron je schopen rozdělit body v hyperprostoru, pokud jsou tyto body lineárně separovatelné [13].

2.2.1 Bázová funkce

Bázová funkce převádí 1 až n vstupů ($\vec{x}$) neuronu na hodnotu $u(x)$, která je použita jako vstup aktivační funkce. Každý vstup má svou váhu. V neuronové síti BP je použita lineární bázová funkce. Jde o jednoduchý skalární součin. Vztah pro výpočet lineární bázové funkce je převzat z [4].

$$u \ = w_0 + \sum_{i=1}^{n} w_i x_i \tag{2.2}$$

kde je
$\vec{x}$ vektor vstupních hodnot,
$\vec{w}$ vektor vah jednotlivých vstupů

Na obrázku 2.1 je možné vidět, že ke všem neuronům skryté a výstupní vrstvy je vyveden jeden vstup navíc o hodnotě konstantní 1, ve vzorci má váhu w_0. Tento vstup zlepšuje učení neuronové sítě. V případě, že na vstupu neuronové sítě by byly samé hodnoty 0, pak by se nebyla schopná nic naučit. V případě, že by tento vstup nebyl přítomen, docházelo by k pouze otáčení hyperplochy ve středu hyperprostoru. Tento prahový vstup umožňuje posunutí hyperplochy.

2.2.2 Aktivační funkce

Existuje několik druhů aktivačních funkcí. Pro účely klasifikace se používá skoková aktivační funkce. Pro potřebu zpracování časových řad se ovšem využívají vhodnější aktivační funkce,

které jsou spojité. Tabulka 2.1 se snaží přiblížit nejčastěji používané aktivační funkce. Je zde taktéž uvedena i jejich derivace, která je používána při učení sítě. Tabulka je převzata z [4].

Název	Předpis funkce	Derivace funkce
Sigmoida	$f(u) = \frac{1}{1+e^{-u}}$	$\frac{df(u)}{du} = f(u)[1 - f(u)]$
Hyperbolický tangens	$f(u) = tanh(u)$	$\frac{df(u)}{du} = 1 - [f(u)]^2$
Lineární funkce	$f(u) = au + b$	$\frac{df(u)}{du} = a$
Prahová funkce	$f(u) = \begin{cases} 1 & u > 0; \\ -1 & u < 0. \end{cases}$	Derivace není pro $u = 0$.

Tabulka 2.1: Aktivační funkce a jejich derivace

2.3 Učení

Pro učení dopředné sítě je používán algoritmus zpětného šíření chyby. Nejprve bude vysvětleno učení jednoho neuronu, následně celé sítě. Celé odvození algoritmu backpropagation lze nalézt v literatuře [4, 13].

2.3.1 Učení neuronu

Učení probíhá jako nastavování synaptických vah neuronu. Na počátku jsou váhy inicializovány náhodně. Jejich další změna je popsána vzorcem [4]:

$$w_i(t+1) = w_i(t) + \eta \sum_{k=1}^{K} \delta(k) x_i(k) \tag{2.3}$$

w_i je váha i-tého vstupu neuronu, η koeficient učení, $\delta(k)$ je chybový signál. Pokud je použita aktivační funkce sigmoida, pak pro výpočet chybového signálu $\delta(k)$ platí:

$$\delta(k) = \frac{\partial E}{\partial u} = [d(k) - y(k)] \cdot y(k) \cdot [1 - y(k)] \tag{2.4}$$

2.3.2 Učení sítě

Aby neuronová síť byla schopná se učit, pak je nutné sestavení trénovací množiny. Ze vzorků je pomocí metody časového okna, která je vysvětlena v podkapitole 5.3, vytvořena množina dvojic $(\vec{i}, \vec{o})$, kde $\vec{i}$ je vstupní vektor a $\vec{o}$ je výstupní vektor.

Nejprve je vložen vstupní vektor na vstup neuronové sítě. Pro všechny neurony skryté vrstvy je spočítán jejich nový výstup. Tento výstup skryté vrstvy je případně poslán další skryté vrstvě, pokud je skrytých vrstev více, nebo rovnou výstupní vrstvě. Nakonec je na výstupu sítě získána nová hodnota. Je vypočtena chyba odečtením nového výstupu od požadovaného výstupu. Tato chyba se spočítá zpětně pro další vrstvy. Pro každou vazbu (synapsi) je nastavena nová váha v závislosti na velikosti chyby a to zpětně od výstupní vrstvy k první skryté vrstvě. Tento postup se opakuje pro všechny prvky trénovací množiny.

Jedno učení celé trénovací množiny se nazývá epocha. Počet epoch učení může být stanoven napevno nebo je možné, aby se učení ukončilo samo v případě růstu chyby. Chyba je na počátku velká a při správném učení s rostoucím počtem epoch klesá. Následující vztah

[4] slouží pro úpravu vah neuronu v kontextu celé sítě. Ve vzorci je rovněž pro zlepšení kvality učení přidáno momentum α, důvod pro tento parametr je v podkapitole 2.4.2.

$$w_{ij}^{L}(t+1) = w_{ij}^{L}(t) + \eta \sum_{k=1}^{K} \delta_i^L(k) y_j^{L-1}(k) + \alpha[w_{ij}^{L}(t) - w_{ij}^{L}(t-1)] \tag{2.5}$$

2.4 Parametry ovlivňující učení

Existuje několik parametrů, které byly již byly zmíněny a které ovlivňují kvalitu a rychlost učení. Jejich konkrétní nastavení závisí na konkrétní úloze a na úrovni znalostí a zkušeností návrháře neuronové sítě.

2.4.1 Koeficient učení

Koeficient učení ovlivňuje především rychlost učení. Menší koeficient znamená pomalejší, ale kvalitnější učení a naopak. Koeficient učení je nutné nastavovat v intervalu $(0, 1)$. Dle literatury [4] je vhodná hodnota z intervalu $(0, 3\rangle$. Někdy je vhodné hodnotu koeficientu během učení postupně snižovat.

2.4.2 Momentum

Změna váhy není řízena pouze aktuální změnou, ale i předchozí změnou s menší intenzitou, což dodává učení jistou setrvačnost. Momentum se přidává do rovnice pro změnu váhy, aby se předešlo uvíznutí v lokálním minimu. Parametr momenta leží v intervalu $\alpha \in (0, 1)$. Často se volí velikost momenta větší než koeficientu učení, literatura [4] doporučuje $(0,6; 0,9)$.

2.4.3 Počet epoch

Počet epoch je jiný název pro počet iterací učení. Jedna epocha představuje jedno učení všech trénovacích záznamů z trénovací množiny. Malý počet epoch může znamenat malé naučení sítě, naopak velký počet epoch může způsobit přetrénování. Na správný počet epoch pro konkrétní parametry sítě i data se dá přijít experimentálně, případně je možné pomocí vhodného algoritmu zastavit učení v době, kdy začíná růst opět chyba. Jedním z použitelných algoritmů je křížové ověřování správnosti (Cross-validation).

2.4.4 Počáteční nastavení vah

Počáteční nastavení vah je klíčové pro učení sítě. Jako váhy se nejčastěji používají pseudonáhodná čísla v rovnoměrné rozložení. Důležitá je správný rozsah generovaných hodnot. Obvykle se používají hodnoty z intervalu $\langle -0.5, 0.5\rangle$. Pro určení rozsahu lze použít také následující interval $\langle -\frac{3}{\sqrt{n}}, \frac{3}{\sqrt{n}}\rangle$, kde n značí počet vstupů neuronu, jehož váhy se nastavují [13].

2.4.5 Počet vrstev sítě

Obvykle stačí jedna skrytá vrstva neuronů. Více skrytých vrstev většinou nedává lepší výsledky, naopak může vést k delšímu učení a přetrénování sítě. V praxi se můžeme občas setkat se dvěma skrytými vrstvami, větší počet vrstev se používá zřídka [6].

2.4.6 Počet skrytých neuronů

Neexistuje univerzální pravidlo pro určení vhodného počtu skrytých neuronů. Je potřeba volit experimentálně v závislosti na konkrétním problému. Obecně platí, že malý počet skrytých neuronů neumožní síti se trénovací data dostatečně naučit a naopak příliš velký počet skrytých neuronů oslabuje schopnost zevšeobecňování, vede k přetrénování sítě. Přetrénování nastává v případě, kdy neuronová síť má příliš malý počet trénovacích dat vzhledem k počtu skrytých neuronů. Má sklony pamatovat si každý samostatný vstup namísto zevšeobecňování [6].

2.5 Příprava dat

Pro zpracování neuronovou sítí je vhodné si nachystat data. Je několik způsobů přípravy dat, které mohou ovlivnit chování sítě. Velké hodnoty se například neuronové sítě špatně učí. Velmi velké hodnoty mohou zastínit význam velmi malých hodnot. Často nám učení znehodnocuje trendová složka signálu, nebo jiné sezónní vlivy.

2.5.1 Normalizace dat

Normalizace dat nebo také standardizace dat je prováděna vždy, jelikož neuronové sítě pracují v malých intervalech oproti datům libovolné velikosti v časových řadaách. Časové řady mohou obsahovat libovolné hodnoty, ale neuronové sítě vnitřně pracují s malými rozsahy. Vstupy se transformují do intervalu $\langle -1, 1 \rangle$, protože sigmoidální aktivační funkce má v tomto intervalu největší změnu hodnot. Výstupy neuronové sítě se transformují do intervalu $\langle 0, 1 \rangle$, což je obor hodnot sigmoidy. Tyto transformace se nazývají normalizace, někdy také standardizace. Neuronová síť pracuje s normalizovanými daty, výstupy neuronové sítě je pak nutné zpětným způsobem denormalizovat, abychom získali hodnoty v původním rozsahu. Vzorce jsou převzaty z [10].

Vstupy je vhodné normalizovat na střed 0 a směrodatnou odchylku 1. Tímto postupem dosáhneme požadovaného intervalu $\langle -1, 1 \rangle$. Pro výpočet jsou použity všechny vstupní hodnoty trénovací množiny.

$$x' = \frac{x - \bar{x}}{\sigma} \tag{2.6}$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{2.7}$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{2.8}$$

kde x jsou vstupy,
x' normalizované vstupy,
$\bar{x}$ aritmetický průměr vstupů,
σ směrodatná odchylka vstupů,
N počet vstupů.

Normalizace vstupů je počítána z minima a maxima z výstupních hodnot trénovací množiny.

$$y' = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{2.9}$$

y výstupy,
y' normalizované výstupy,
y_{min} nejmenší výstup,
y_{max} největší výstup.

2.5.2 Diference dat

Jednou z oblíbených technik přípravy dat je výpočet diference časové řady. Diference spočívá v odečtení hodnot sousedních vzorků časové řady, viz vztah (2.10). Místo s původními daty pracujeme s jejich změnou. Diference eliminuje konstantní trend, což je jedna z hlavních výhod. Zvláště na finančních trzích nás zajímá spíše růst nebo pokles trhu, než konkrétní nová hodnota trhového indexu. Při používání diference je potřeba postupovat opatrně, protože pokud je v signálu nějaká důležitá informace uložená s malou frekvencí, pak může být tato informace ztracena [8].

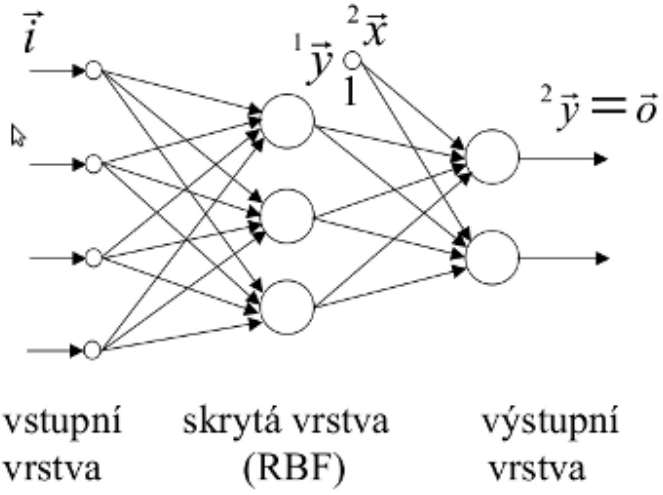
$$y'(t) = y(t) - y(t-1) \tag{2.10}$$

Kapitola 3

Neuronová síť s radiálními bázovými funkcemi

Tato kapitola se zabývá základním popisem neuronových sítí s radiálními bázovými funkcemi. Jsou zde vysvětleny základní algoritmy učení této sítě. Tato neuronová síť aproximuje křivku lokálně pomocí hyperkoulí na rozdíl od předchozí sítě BP, která aproximuje globálně pomocí hyperploch.

3.1 Popis sítě



Obrázek 3.1: Struktura RBF sítě, obrázek převzat z [13]

Struktura této sítě je na obrázku 3.1. Má tři vrstvy. První je vstupní vrstva, druhá je skrytá vrstva s RBF neurony, třetí je výstupní vrstva s jedním lineárně bázovým neuronem a lineární aktivační funkcí. Je podobná dopředné síti, ale liší se chováním neuronů a také odlišným postupem učení. Ve vrstvě s RBF neurony je používána radiální bázová funkce. Počítá euklidovskou vzdálenost vstupního vektoru (bod v hyperprostoru) od středu hyperkoule v hyperprostoru. Obecně lze popsat RBF síť následujícím vztahem [4]:

$$y = w_0 + \sum_{i=1}^{n_h} w_i f(\|\vec{x} - \vec{c_i}\|) \tag{3.1}$$

kde f jsou radiální bázové funkce, w_i jsou váhy výstupní vrstvy, w_0 je váha prahového neuronu, $\vec{x}$ je vstup neuronu, $\vec{c_i}$ jsou středy patřící bázovým funkcím, n_h je počet bázových funkcí v síti a $\|\cdot\|$ značí euklidovskou vzdálenost.

Euklidovská vzdálenost se počítá následujícím vztahem [13]:

$$u_k = \sqrt{\sum_{i=1}^{n} (x_i - c_i)^2} \tag{3.2}$$

Existuje několik druhů používaných radiálně bázových funkcí [11]:

- Gaussova funkce
$$f(x) = e^{\frac{-(x-c)^2}{r^2}} \tag{3.3}$$
- Multikvadratická funkce
$$f(x) = \sqrt{(x-c)^2 + r^2} \tag{3.4}$$
- Inverzní multikvadratická funkce
$$f(x) = \frac{1}{\sqrt{1 + \frac{(x-c)^2}{r^2}}} \tag{3.5}$$
- Cauchyho funkce
$$f(x) = \frac{1}{1 + \frac{(x-c)^2}{r^2}} \tag{3.6}$$

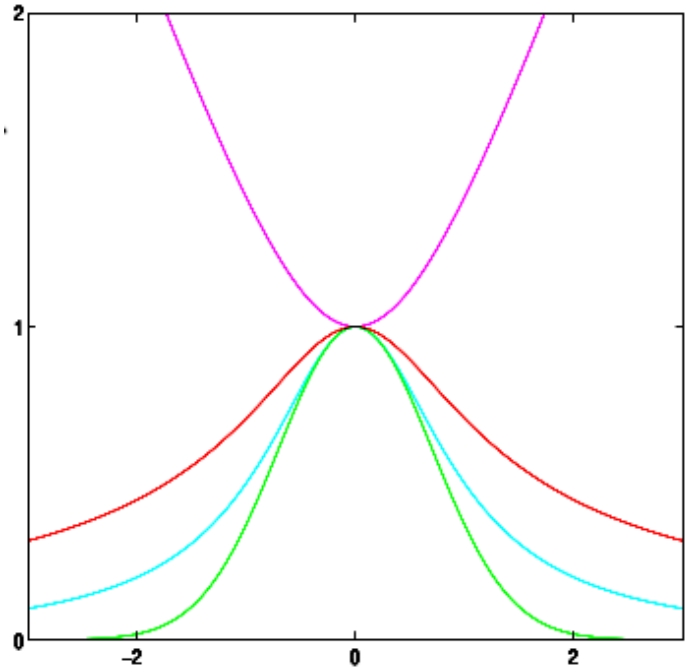
Na obrázku 3.2 jsou jednotlivé radiální bázové funkce, pro které platí $c = 0$ a $r = 1$. Ve výstupní vrstvě je jeden neuron, do něhož jsou přivedeny výstupy všech RBF neuronů. Je v něm použita lineární bázová funkce a lineární aktivační funkce [13]:

$$y_j = u_j = \sum_{k=0}^{n_1} w_{jk} x_k = w_{j0} + \sum_{k=1}^{n_1} w_{jk} y_k \tag{3.7}$$

3.2 Učení

Učení této sítě se liší od dopředné neuronové sítě. Je potřeba nastavit tři druhy parametrů. Pro každý RBF neuron skryté vrstvy je třeba nastavit souřadnice středu hyperkoule c a poloměr této hyperkoule σ. Každá taková hyperkoule v hyperprostoru odpovídá jednomu RBF neuronu. Ve výstupní vrstvě je potřeba nastavit váhy výstupního neuronu.

Nejprve jsou vhodným postupem jednorázově zvoleny středy hyperkoulí a jejich poloměry. Teprve potom je použit algoritmus pro úpravu vah, který už upravuje jen váhy výstupní vrstvy. Středy hyperkoulí je možné zvolit náhodně z trénovací množiny. Je ovšem vhodné použít nějaký algoritmus pro zjištění středu hyperkoulí, v této práci je použit algoritmus K-means.



Obrázek 3.2: Radiální bázové funkce – multikvadratická (fialová), inverzní multikvadratická (červená), Cauchyho (tyrkysová), Gaussova (zelená), obrázek převzat z [11]

3.2.1 Algoritmus K-means

Vstupy neuronové sítě je možné si představit jako souřadnice bodu v n-rozměrném hyperprostoru, kde n je počet vstupů. Jeden takový bod je tedy n-rozměrný vektor $\vec{x} = {}^T[x_1, x_2, \ldots, x_n]$. Algoritmus K-means [14] se snaží nalézt shluky těchto bodů. Počet hledaných shluků odpovídá počtu RBF neuronů a v algoritmu je označen písmene k. Předpokládá se, že je možné rozdělit do shluků všech p vektorů z trénovací množiny $T = \{\vec{x_1}, \vec{x_2}, \ldots, \vec{x_p}\}$. Výstupem algoritmu jsou těžiště $\vec{c_i}$ nalezených shluků.

1. Inicializuj k prototypů, například náhodně vybrané různé vektory z trénovací množiny: $\vec{c_j} = \vec{x_p}, j \in \langle 1, k\rangle, p \in \langle 1, P\rangle$.

2. Opakuj pro každý vektor $\vec{x_p}$ z trénovací množiny: Přiřaď tento vektor do shluku $C_j, j \in \langle 1, k\rangle$, jehož prototyp $\vec{c_j}$ má od vektoru $\vec{x_p}$ nejmenší vzdálenost:
$$|\vec{x_p} - \vec{c_j}| \leq |\vec{x_p} - \vec{c_i}|$$

3. Opakuj pro každý shluk $C_j, j \in \langle 1, k\rangle$: Přepočítej prototyp $\vec{c_j}$ tak, aby byl těžištěm koncových bodů všech vektorů, které jsou k tomuto shluku právě přiřazené (nechť n_j je počet těchto vektorů):
$$\vec{c_j} = \frac{\sum_{\vec{x_i} \in C_j} x_i}{n_j}$$

4. Vypočítej chybu aktuálního stavu shlukování:
$$E = \sum_{j=1}^{k} \sum_{\vec{x_i} \in C_j} |\vec{x_i} - \vec{c_j}|^2$$

5. Pokud chyba E klesla, nebo pokud byl některý vektor přiřazen k jinému shluku, vrať se na bod 2.

Po nalezení těžišť shluků je potřeba spočítat poloměr hyperkoule, což je odchylka všech bodů patřících shluku od jejich těžiště. Vzorec převzat z [13].

$$\sigma_k = \sqrt{\sum_{i_k \in C_k} \|\vec{i_k} - \vec{u_k}\|} \quad k \ \in \{k \ \ldots K\} \tag{3.8}$$

3.2.2 Učení výstupní vrstvy

Po inicializaci RBF neuronů algoritmem K-means je hledána optimální velikost vah výstupní vrstvy. Je možné si vybrat z více algoritmů. Výstupní vrstvu je možné učit již popsaným algoritmem zpětného šíření chyby nebo například iterační metodou nejmenších čtverců. Zvolil jsem první možnost a ta zde bude popsána. Pro změnu váhy k-tého vstupu j-tého neuronu výstupní vrstvy platí vztah [13]:

$$\begin{aligned} \Delta w_{jk} &= -\eta \frac{\partial E}{\partial w_{jk}} \\ &= \eta (d_j - {}^2y_j) \frac{\partial {}^2y_j}{\partial w_{jk}} \\ &= \eta (d_j - {}^2y_j) \frac{\partial (w_{j0} + \sum_{k=1}^{n_1} w_{jk} y_k)}{\partial w_{jk}} \\ &= \eta (d_j - {}^2y_j) y_k \end{aligned} \tag{3.9}$$

w_{jk} je váha k-tého vstupu j-tého neuronu,
η koeficient učení pro změnu vah,
d_j požadovaná hodnota na výstupu j-tého neuronu,
${}^2y_j$ vypočtená hodnota výstupu j-tého neuronu druhé (výstupní) vrstvy,
y_k výstup k-tého neuronu v předchozí RBF vrstvě.

Kapitola 4

Další typy neuronových sítí

Tato kapitola se zabývá popisem dalších druhů neuronových sítí, které se používají také pro predikci, ale obvykle méně než BP a RBF sítě. Bude zmíněna neuronová síť vyššího řádu a také rekurentní neuronové sítě.

4.1 Neuronová síť vyššího řádu

Běžná dopředná síť BP je síť prvního řádu. Pro zachycení korelací vyšších řádů se používají neuronové sítě vyššího řádu [12] (HONN - Higher Order Neural Network). Vztahy mezi jednotlivými vstupy jsou zachyceny jejich vzájemným vynásobením. Tuto operaci je možné provést ještě před vstupem dat do sítě nebo vhodnou volbou neuronů ve skryté vrstvě. Mějme vektor vstupních dat $x_1, \ldots, x_n$. V prvním případě použijeme jako vstup sítě nejen tuto n-tici, ale také všechny součiny $x_i x_j$ pro $i \neq j$. Matematický popis neuronu bude vypadat následovně:

$$y = f(w_0 + \sum_{i1}^{n} w_{i1} w_{i1} + \sum_{i1,i2}^{n} w_{i1,i2} x_{i1} x_{i2}) \tag{4.1}$$

kde $x = [x_1, x_2, \ldots, x_n]^T$ je vstupní vektor, y je výstup neuronu, f je aktivační funkce. Ve zmiňovaném druhém případě jde o použití speciálních neuronů. HONN mají podobnou strukturu jako klasické dopředné sítě, pouze jejich lineární bázové funkce jsou nahrazeny násobnými. Počet neuronů v těchto sítích je obvykle nižší než u dopředných sítí. Počet vah se zvyšuje exponenciálně se zvyšováním počtu vstupů, proto se architektury třetího a vyššího řádu používají zřídka. V této práci je v dalším textu pod neuronovou sítí vyššího řádu myšlena neuronová síť druhého řádu.

4.2 Elmanova síť

Elmanova síť [7] je jednoduchá rekurentní síť, byla navržena Jeffem Elmanem v roce 1990. Slovo rekurentní zde znamená, že kromě dopředných vazeb mezi neurony obsahuje zpětné vazby mezi neurony. Takový systém má jakousi setrvačnou paměť, která si pamatuje předchozí vývoj. Tato paměť pak ovlivňuje budoucí výstup. Její struktura je podobná dopředné síti, ale ze skryté vrstvy je výstup vyveden nejen do výstupní vrstvy, ale také do kontextové vrstvy. Z této kontextové vrstvy je výstup vyveden na vstup skryté vrstvy. Každý kontextový neuron je propojen se všemi neurony skryté vrstvy. Počet kontextových neuronů je

stejný jako počet neuronů skryté vrstvy. Popsané vazby způsobují, že výstupy skryté vrstvy v čase t ovlivňují své výstupy v čase $t + 1$.

4.2.1 Výpočet výstupu

Výpočet výstupu je vzhledem ke zpětným vazbám odlišný od dopředné sítě. Nejprve je vložen v čase t na vstup sítě vstupní vektor. Pak jsou vypočteny výstupy skryté vrstvy ze vstupů ze vstupní vrstvy a kontextové vrstvy. Následně je vypočten výstup sítě a výstupy skryté vrstvy jsou zkopírovány do kontextové vrstvy.

4.2.2 Učení

V čase $t = 0$ jsou kontextové neurony nastaveny na nulovou hodnotu, obvykle pomocí přidání vzorku s nulovými hodnotami do trénovací množiny. V případě, že trénovací množina obsahuje vzorky se samými nulami, pak je přidán vstupní neuron pro reset sítě. Pokud je neuron pro reset nastaven na nulu, pak dojde k vymazaní všech aktivací uvnitř sítě. Na konci každého kroku učení kromě prvního jsou známy aktivace neuronů uvnitř sítě. Pro učení lze použít algoritmus zpětného šíření chyby. Vzhledem k přítomnosti kontextové vrstvy je pro učení použit chybový signál ze současného i předchozího kroku. Někdy je vhodné ukládat aktivace kontextové vrstvy pro víc kroků nazpět. Tento postup se nazývá zpětné šíření chyby v čase [7].

4.3 Jordanova síť

Jordanova síť [7] je pojmenována po Saulu Jordanovi, který ji vytvořil v roce 1986. Hlavní rozdíl oproti Elmanově síti je ten, že na vstup kontextové vrstvy není přiveden výstup skryté vrstvy, ale přímo výstup sítě. Tato síť není plně rekurentní, ale nazývá se částečně rekurentní. Tím, že do kontextové vrstvy se pouze kopíruje předchozí výstup, dochází ke ztrátě závislostí, které je schopná Elmanova síť zachytit. Pro vyhodnocení výstupu a učení platí totéž, co pro Jordanovu síť.

Kapitola 5

Predikce

Tato kapitola vysvětluje pojem časové řady, predikce časové řady. Zabývá se také problémy, které je nutné řešit u neuronových sítí v souvislosti se zpracováním časových řad. Lehce budou zmíněna základní kritéria kvality predikce. Pro pochopení pojmu predikce je tedy nejprve nutné definovat, co je to časová řada.

5.1 Časová řada

Časová řada [2] je sekvence vektorů závislá na čase t:

$$\vec{x}(t), t = 0, 1, ... \tag{5.1}$$

Prvky vektorů mohou být jakékoli měřitelné veličiny, například

- teplota vzduchu v budově
- cena určitého zboží na daném trhu
- počet narození v daném městě
- množství spotřebované vody v dané oblasti

Na vektor $\vec{x}$ lze také nahlížet jako na spojitou funkci času t. Z praktických důvodů se ale na čas nahlíží jako na diskrétní časové úseky, což vede k výskytu $\vec{x}$ na konci každého úseku pevně dané velikosti. Velikost časového intervalu často závisí na řešené úloze a může být v jakýchkoli jednotkách - od milisekund až po roky.

Tato práce se zabývá modelováním časových řad pomocí neuronových sítí, což jsou nelineární modely. Existují ovšem jiné metody modelování, mezi něž patří různé lineární modely, například AR (Autoregressive) modely, MA (Moving average) modely a ARMA (Autoregressive moving average) modely [12]. Studium těchto modelů ovšem není předmětem této práce.

5.2 Predikce

Predikce je jiný název předpověď. Je-li možné předpovědět časovou řadu přesně, pak ji označujeme jako deterministickou. Obvykle jsou běžné časové řady stochastické procesy ovlivněné šumem, jejichž charakteristika se v čase mění. Jako důsledek je budoucí vývoj jen

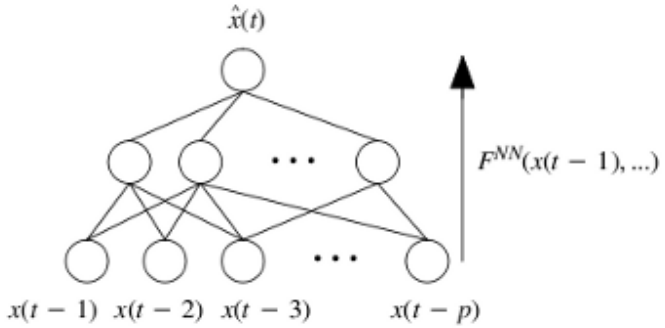
částečně určen předchozím vývojem a má pravděpodobnostní rozložení určené pozorováním v minulosti a dalšími dřívějšími znalostmi. Cílem predikce časových řad je modelovat podkladový mechanismus, který bude generovat časovou řadu pro krátký až střední časový interval do budoucnosti [4].

Hlavním úkolem predikce je nalezení takové funkce, která je schopná aproximovat vývoj časové řady [7].

$$y(t+1) = f(y(t), y(t-1), ..., y(1)) \tag{5.2}$$

Nikdy ovšem nejsme schopni předpovědět přesně budoucnost jen na základě minulosti. Budoucnost je vždy nevypočitatelná. Dochází k pouze odhadům a měřitelná je pouze míra úspěšnosti aproximace. Předpověď je úspěšná na nejbližší následující hodnoty, čím dále do budoucnosti se snažíme předpovídat, tím více roste chyba predikce. Tato práce se navíc zabývá predikcí jen na základě minulého vývoje časové řady. K zachycení vlivu jiných okolností, které mohou ovlivnit kvalitu predikce se používají intervenční proměnné. Tato práce se predikcí s intervenčními proměnnými nebude zabývat.

5.3 Metoda časového okna



Obrázek 5.1: Časové okno, obrázek převzat z [2]

Metoda časového (někdy také klouzajícího) okna [2] umožňuje neuronovým sítím pracovat s časovými řadami. Ze vzorků původní časové řady se tvoří prvky pro učení a to tak, že prvních n hodnot je dáno na vstup a následující hodnota časové řady je dána jako požadovaná na výstup. Další prvek pro učení je vytvořen posunutím o jeden prvek a jako vstup je vybráno opět n hodnot, hodnota $n+1$ je opět jako požadovaná. Tímto postupem je z původních trénovacích vzorků vytvořena celá trénovací množina, která se skládá ze seznamu prvků, kde každý prvek je tvořen vstupním vektorem a požadovaným výstupním vektorem. Toto učení je někdy popisováno jako posouvání časového okna po poli trénovacích vzorků. Jak jsou vstupy a výstupy dány neuronové síti ke zpracování, je vidět na obrázku 5.1.

5.4 Kvalita predikce

Kvalita predikce je rozhodující vlastnost při měření použité neuronové sítě. Často se používají takové metody, které používají rozdíl skutečné a predikované hodnoty časové řady. Chybu predikce i-tého vzorku je možné vyjádřit jako [7]:

$$E_i = d_i - x_i \tag{5.3}$$

E_i je chyba i-tého vzorku
d_i požadovaná hodnota i-tého vzorku
x_i predikovaná hodnota i-tého vzorku
N počet vstupních vzorků
σ rozptyl dat

- Aritmetický průměr absolutních odchylek (MAD - Mean Absolute Deviation)

$$MAD = \frac{1}{N} \sum_{i=1}^{N} |d_i - x_i| \tag{5.4}$$

- Střední kvadratická chyba (MSE - Mean Square Error)

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (d_i - x_i)^2 \tag{5.5}$$

- Druhá odmocnina z aritmetického průměru druhých mocnin odchylek (RMSE - Root Mean Square Error)

$$RMSE = \sqrt{\frac{1}{\sigma^2 N} \sum_{i=1}^{N} (d_i - x_i)^2} \tag{5.6}$$

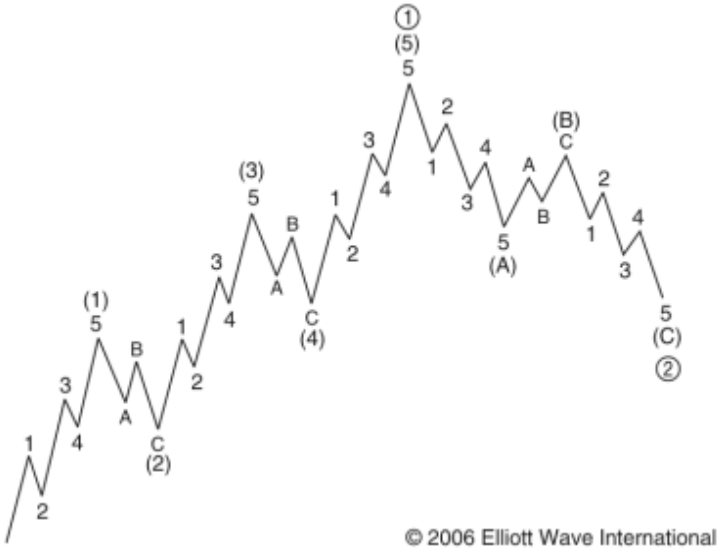
Pro měření bude využívána chyba RMSE. Tato statistická chyba má výhodu oproti MSE v tom, že je ve stejných jednotkách, jako vstupní data, kdežto MSE je kvadratická oproti vstupním datům.

5.5 Jiné možnosti predikce

Existují i jiné způsoby predikce vývoje finančních trhů než pomocí neuronových sítí. Finanční analytici pracují s fundamentální, technickou nebo psychologickou analýzou finančních trhů. Zpracováním časových řad se zabývá technická analýza, jednou z jejích možností jsou Elliottovy vlny.

5.5.1 Elliottovy vlny

Elliottovy vlny [1] jsou jednou z metod technické analýzy finančních trhů, jejich autorem je Ralph Nelson Elliott, který je objevil v 30. letech 20. století. Jsou založeny na identifikaci opakujících se vzorů v časové řadě finančního trhu. Základním principem Elliottových vln je, že trh výrazně roste v impulzní vlně, která je složena z pěti podvln (1-2-3-4-5). Trh pak mírně poklesne vlivem opravné vlny, která je složena ze tří podvln (a-b-c). Celá Elliottova vlna se pak skládá z celkem osmi vln a dvou fází. Ukázka Elliottovy vlny je na obrázku 5.2. Tyto vlny jsou ve skutečné časové řadě strukturované a vytvářejí opakující se vzory (fraktály) postupně větší velikosti. Existuje mnoho variací těchto vln, například každý vzor těchto vln má své požadavky a vývojové tendence.



Obrázek 5.2: Elliottova vlna, obrázek převzat z [1]

Kapitola 6

Implementace

V této kapitole budou na úvod zmíněny existující systémy použitelné mimo jiné také pro predikci pomocích neuronových sítí. Dále bude následovat popis aplikace, jejíž implementace je součástí této bakalářské práce. Samotná aplikace byla napsána v jazyce Python, což je v poslední době velmi oblíbený skriptovací jazyk. Popisy implementací jednotlivých částí aplikace si nekladou za cíl podrobnou dokumentaci, ale chtějí ukázat výhodné využití objektového návrhu a jeho filozofii.

6.1 Současné systémy

Aplikací pro predikci pomocí neuronové sítě existuje velké množství. Konkrétně pro predikci finančních trhů se často jedná o proprietární placené řešení. I samotné vývoje indexů finančních trhů jsou většinou placené, zdarma se dají sehnat spíše historické ceny než aktuální data. Následující výčet se snaží stručně vyjmenovat některé používané systémy pro predikci [10]:

- JavaNNS

  Java Neural Network Simulator je pokračovatelem projektu SNNS (Stuttgart Neural Network Simulator). Jedná se o volně dostupný software napsaný v Javě s přívětivým uživatelským rozhraním. Umožňuje vytvářet připravené sítě nebo definovat vlastní. Tyto sítě pak je možné trénovat, analyzovat, vizualizovat. Domovské stránky tohoto systému jsou na adrese `http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/welcome_e.html`.

- Emergent

  Emergent je open source simulátor neuronových sítí. Je napsán v jazyce C++ ve frameworku Qt a má grafické uživatelské rozhraní. Umožňuje použití více druhů neuronových sítí a stejně také algoritmů pro jejich učení. Stránky projektu jsou na adrese `http://grey.colorado.edu/emergent/index.php/Main_Page`.

- Matlab Neural Network Toolbox

  Jde o modul neuronových sítí pro známý komerční matematický software Matlab. Obsahuje velké množství architektur sítí, ale také mnoho technik jejich učení. Dále má v sobě názorné ukázky konkrétních neuronových sítí.

6.2 Implementace datových struktur

Nejprve bylo potřeba implementovat vhodné datové typy pro ukládání dat. Tyto datové struktury byly řešeny objektově. Jsou uloženy v souboru `samples.py`. Základem je třída `Sample`. Tato třída slouží pro uložení jednoho vzorku časové řady. Skládá se z času vzorku a jeho hodnoty. Pole objektů této třídy vytvoří časovou řadu. Další důležitou třídou je `Record`, která je určená pro uložení jednoho záznamu trénovací nebo testovací množiny. Tento záznam obsahuje vstupní a výstupní vektor sítě, kde každý prvek vektoru je objekt třídy `Sample`. Třída `Records` je pak polem záznamů a představuje trénovací či testovací množinu. `SamplesDataSource` je již třída, která umožňuje práci s časovou řadou. Umožňuje získat trénovací či testovací množinu, případně spočítat diferenci časové řady. Aby byla práce s časovými řadami nezávislá na formátu zdrojového souboru, tak je implementována třída `YahooFinance`, která dědí všechny vlastnosti třídy `SamplesDataSource`. Implementuje pouze vlastní konstruktor, ve kterém je zpracován soubor se zdrojovými daty v CSV formátu pocházejícími z portálu `http://finance.yahoo.com/`. Další zdroje dat je možné doprogramovat pomocí dalšího zdědění třídy SamplesDataSource.

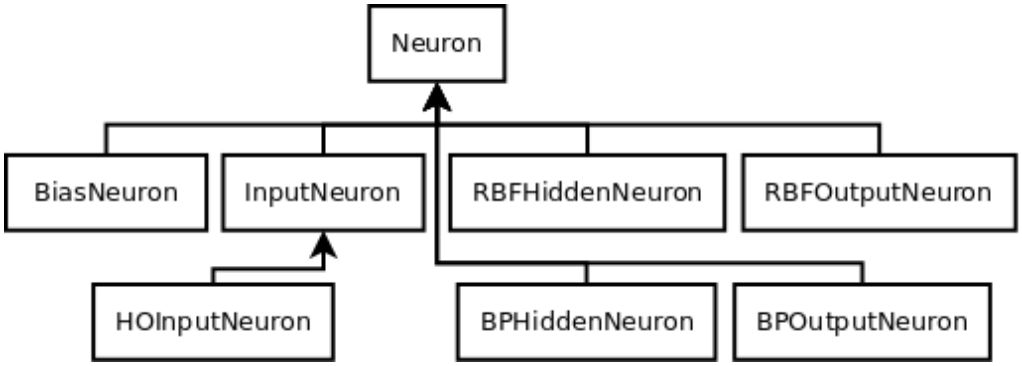
6.3 Implementace neuronových sítí

Základem je soubor `network.py` obsahující třídy implementující neuron, vrstvu sítě a síť samotnou. Některé třídy jsou zde rozšířeny o další vlastnosti, například existují potomci neuronu, jako je vstupní neuron, prahový neuron, nebo v případě vrstvy existuje potomek vstupní vrstva. V samotných souborech jednotlivých sítí `bp.py`, `ho.py` a `rbf.py` jsou již třídy zděděné. V diagramech 6.1, 6.2 a 6.3 je přehledně znázorněna hierarchie dědičnosti tříd `Neuron`, `Layer` a `Network`. Byla snaha implementovat aplikaci tak, aby implementace jedné stejné činnosti nebyla v aplikaci na více místech najednou.

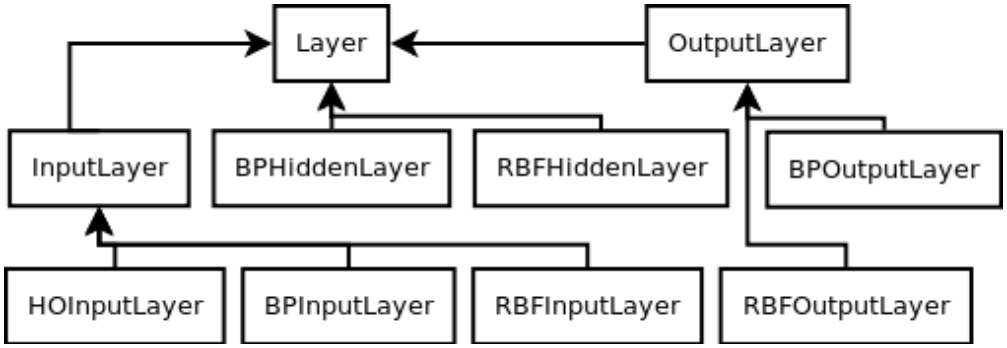
Uživatel aplikace ovšem přijde nejčastěji do styku s třídou `Prediction`. Její použití je popsáno v příloze v manuálu. Jejím smyslem je zastřešit práci s predikcí pomocí neuronových sítí. Zastřešuje načtení dat, jejich přípravu, vytvoření neuronové sítě a také vykreslení grafů. Snadno umožňuje provádět hromadná měření. Bohužel vyžaduje od uživatele základní znalost programování v jazyce Python. Druhou možností jak provádět měření je pomocí spuštění skriptu `main.py`. Tento způsob umožňuje provádět měření bez znalosti jazyka Python, protože zadávání všech parametrů probíhá v příkazové řádce, což je na druhou stranu poněkud nepohodlné.

6.4 Implementace statistických chyb

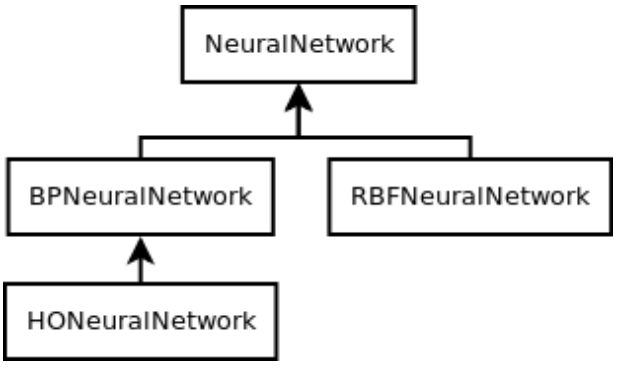
Statistické chyby jsou implementovány v souboru `error.py`. Jednoduchá třída `Error` slouží pouze pro uložení vypočtené a požadované hodnoty jedné výstupní hodnoty. Pro uložení pole chyb jedné trénovací či testovací množiny slouží třída `EpochError`. Z těchto dat je pak možné vypočíst různé statistické chyby, v této práci pro účely srovnávání je pak nejčastěji využívána chyba RMSE. Třída `TrainingError` představuje pole chyb epoch a používá se jen při učení, zejména pro sledování trénovací chyby vývoje chyby v závislosti na epoše.



Obrázek 6.1: Diagram dědičnosti třídy `Neuron`



Obrázek 6.2: Diagram dědičnosti třídy `Layer`



Obrázek 6.3: Diagram dědičnosti třídy `NeuralNetwork`

Kapitola 7

Experimenty

Cílem této kapitoly je zkoumat vybrané parametry neuronových sítí. Budou prováděna měření počtu neuronů skryté vrstvy, počtu vstupů sítě, velikosti koeficientu učení, počtu skrytých vrstev sítě. V posledním měření bude zkoumána schopnost predikce diferencované časové řady.

Ve všech měřeních bude použita pro učení neuronových sítí metoda časového okna. Neuronová síť BP má aktivační funkci sigmoidu, v případě RBF sítě je použita Gaussova funkce. Pro inicializaci parametrů skrytých neuronů RBF sítě byl použit algoritmus K-means. Pro všechny experimenty byl použit notebook s dvoujádrovým procesorem AMD Turion TL-60 (2GHz, 2x 512 KB L2 cache), operační pamětí 2GB DDR2 a operačním systémem Ubuntu Linux 9.10 s jádrem 2.6.31-21.

Nejdříve bylo třeba zvolit parametry ovlivňující učení sítě. Koeficient učení jsem zvolil 0,1 na základě doporučení z literatury [4], kde je doporučeno volit tuto hodnotu z intervalu $(0; 0, 3)$. Dále bylo potřeba stanovit momentum, které jsem zvolil 0,6 opět na základě doporučení stejné literatury, kde byla doporučena hodnota z intervalu $(0, 6; 0, 9)$. Počet epoch učení jsem zvolil 100. Hodnoty počátečních vah byly generovány v rovnoměrném rozložení z intervalu $\langle -0, 2; 0, 2\rangle$ v případě sítě BP a HO a v intervalu $\langle -0, 001; 0, 001\rangle$ u sítě RBF. Výstup sítě byl ve všech případech jeden. Jako trénovací data byly zvoleny historické ceny indexu Nasdaq Composite za duben až listopad 2009, a to cena v době otevření trhu. Testovací data byla za duben až prosinec téhož roku stejného indexu. Tyto informace platí pro všechna měření, není-li uvedeno u konkrétního experimentu jinak. Hlavním kritériem kvality predikce je druhá odmocnina střední kvadratické chyby (RMSE).

7.1 Měření vlivu počtu skrytých neuronů

Vhodné množství neuronů skryté vrstvy se liší podle konkrétní časové řady, typu neuronové sítě, konkrétního problému. Pro tento pokus byly zvoleny 2 vstupy sítě, což znamená celkem 167 prvků trénovací množiny vytvořené z výše uvedených trénovacích dat. Celé měření trvalo 226 sekund. Srovnání chyby testovacích dat je vidět tabulce 7.1.

Síť	**Počet neuronů skryté vrstvy**						
	2	**3**	**5**	**10**	**20**	**30**	**50**
BP	41,66	40,44	39,33	38,76	38,10	37,92	38,83
HO	41,49	40,34	39,39	38,35	37,21	37,46	37,80
	2	**10**	**35**	**66**	**100**	**133**	**167**
RBF	79,23	86,95	96,40	86,76	77,49	90,16	75,81

Tabulka 7.1: Chyba RMSE testovacích dat

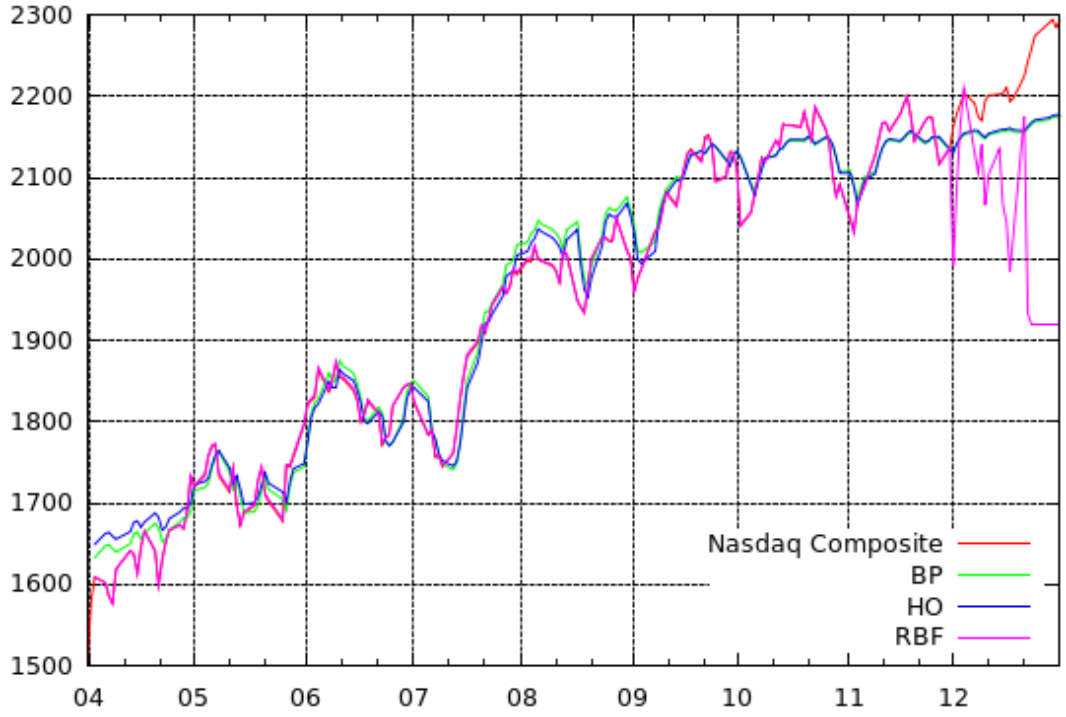
Nejlepšího výsledku dosáhla neuronová síť vyššího řádu s 20 neurony skryté vrstvy. Neuronová síť BP ovšem dosahovala velmi podobných výsledků. Na testovacích datech naopak vůbec neuspěla RBF síť. V tabulce 7.2 je chyba trénovacích dat. Zde nejlepších výsledků dosahovala právě RBF síť. Nejlepší predikce dosáhla při 167 neuronech skryté vrstvy, což je počet prvků celé trénovací množiny. Pro RBF síť tedy platí, že s vyšším počtem neuronů skryté vrstvy dosahuje lepších výsledků. Dále platí, že nejlepších výsledků dosahuje pouze pro data, na kterých byla učená.

Síť	**Počet neuronů skryté vrstvy**						
	2	**3**	**5**	**10**	**20**	**30**	**50**
BP	32,73	32,05	31,13	31,04	30,86	31,19	33,01
HO	33,17	32,60	31,79	31,19	30,78	31,18	32,31
	2	**10**	**35**	**66**	**100**	**133**	**167**
RBF	80,38	61,40	53,22	35,99	18,70	13,22	1,59

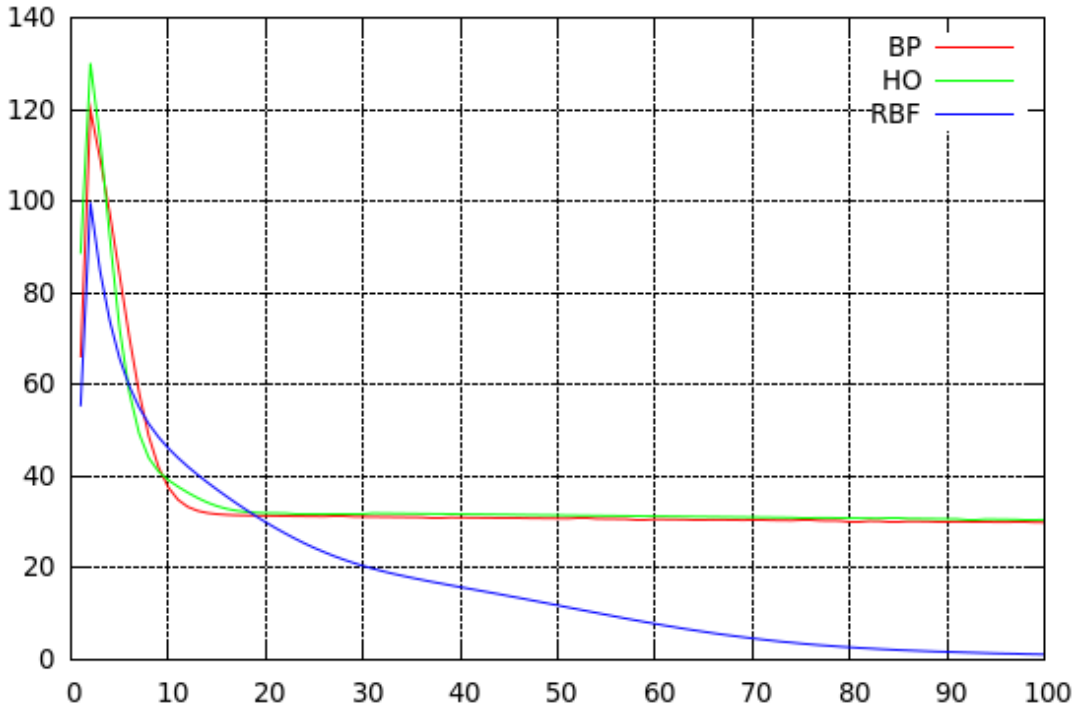
Tabulka 7.2: Chyba RMSE trénovacích dat

Na obrázku 7.1 je graf vývoje finančního trhu Nasdaq Composite za duben až prosinec roku 2009. Současně jsou vidět výsledky nejlepších predikcí jednotlivých druhů neuronových sítí. RBF neuronová síť se téměř po celá trénovací data překrývá s původním signálem, ale v posledním měsíci již aproximuje špatně. Neuronová síť BP a HO dosahují velmi podobných výsledků.

Jak klesá chyba během učení, je vidět na obrázku 7.2. Zobrazené grafy chyb jsou ze tří nejlepších predikcí testovacích dat z každé neuronové sítě. Jak je vidět, během několika prvních epoch chyba rychle klesá, ale pak se postupně ustaluje. Kdyby začala chyba během učení růst, znamená to přetrénování sítě.



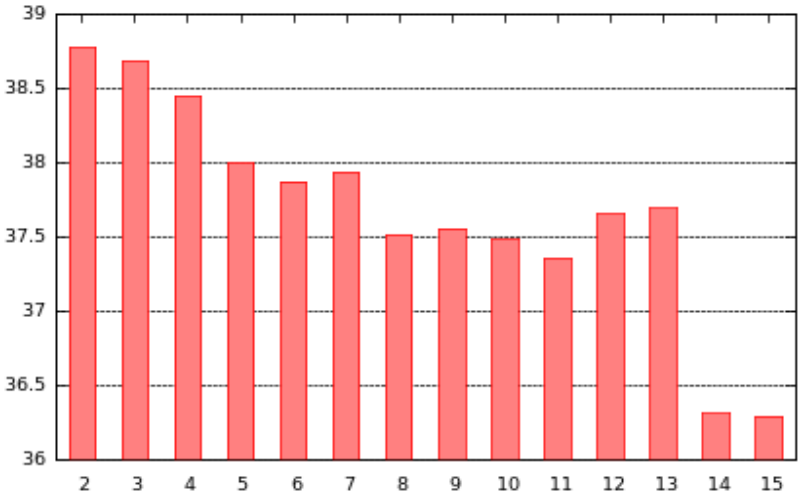
Obrázek 7.1: Graf vývoje indexu Nasdaq Composite za duben až prosinec a jeho predikce



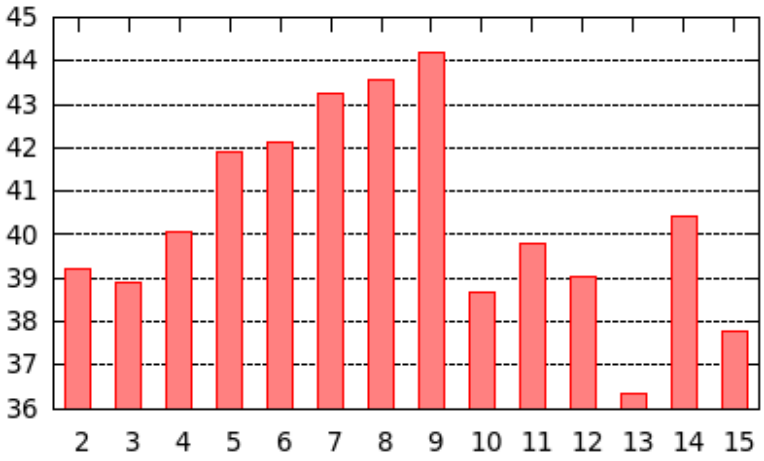
Obrázek 7.2: Chyba RMSE v závislosti na epoše učení

7.2 Měření vlivu počtu vstupů sítě

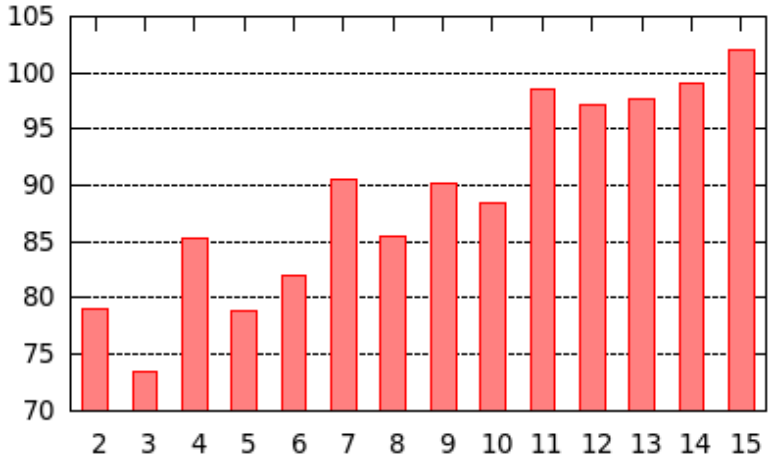
Toto měření zkoumá vliv počtu vstupů na chování neuronové sítě. Počet vstupů je jiné označení pro velikost časového okna. Pro měření bylo použito stejných testovacích i trénovacích dat jako v předchozím experimentu. Testovány byly opět tři sítě. Pro sítě BP a HO byl zvolen počet skrytých neuronů 10, pro síť RBF to bylo 154 skrytých neuronů, aby i při největším testovaném učícím oknu o velikosti 15 odpovídal počet RBF neuronů skryté vrstvy počtu prvků trénovací množiny. Na obrázku 7.3 lze vidět, že nejmenší chyby dosahuje BP síť při velikosti okna 14. Naopak neuronová síť vyššího řádu dávala nejlepší výsledky pro nejmenší měřenou velikost okna 2, viz obrázek 7.4. Podobně RBF síť dosahovala nejlepších výsledků při velikosti okna 3, viz obrázek 7.5.



Obrázek 7.3: Chyba RMSE testovacích dat v závislosti na počtu vstupů BP sítě



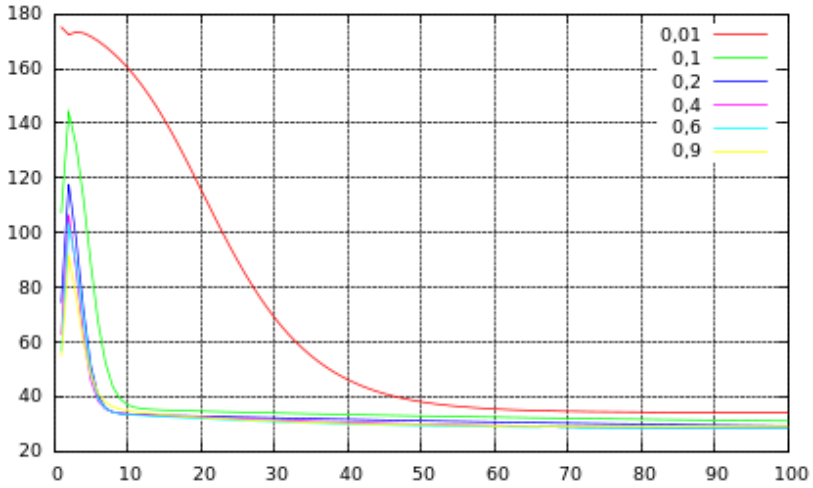
Obrázek 7.4: Chyba RMSE testovacích dat v závislosti na počtu vstupů HO sítě



Obrázek 7.5: Chyba RMSE testovacích dat v závislosti na počtu vstupů RBF sítě

7.3 Měření vlivu koeficientu učení

Koeficient učení má vliv na rychlost a kvalitu učení. V předchozích experimentech byla použita hodnota 0,1. Tento pokus se snaží zjistit jaký koeficient učení je nejvhodnější. Testována byla pouze síť BP se 2 vstupy a 10 skrytými neurony. V grafu 7.6 je vidět, jak jednotlivé koeficienty učení mají rychlost na konvergenci trénování. V tabulce 7.3 je zobrazeno srovnání podle chyby RMSE testovacích dat v závislosti na velikosti koeficientu učení. Celé měření trvalo pouhých 28 sekund. Z měření plyne, že není vhodné používat hodnoty blízké krajním hodnotám intervalu $(0, 1)$.



Obrázek 7.6: Chyby RMSE jednotlivých koeficientů učení v závislosti na epoše

Síť	**Koeficient učení**					
	0,01	**0,1**	**0,2**	**0,4**	**0,6**	**0,9**
BP	42,91	38,63	37,28	37,18	38,63	41,91

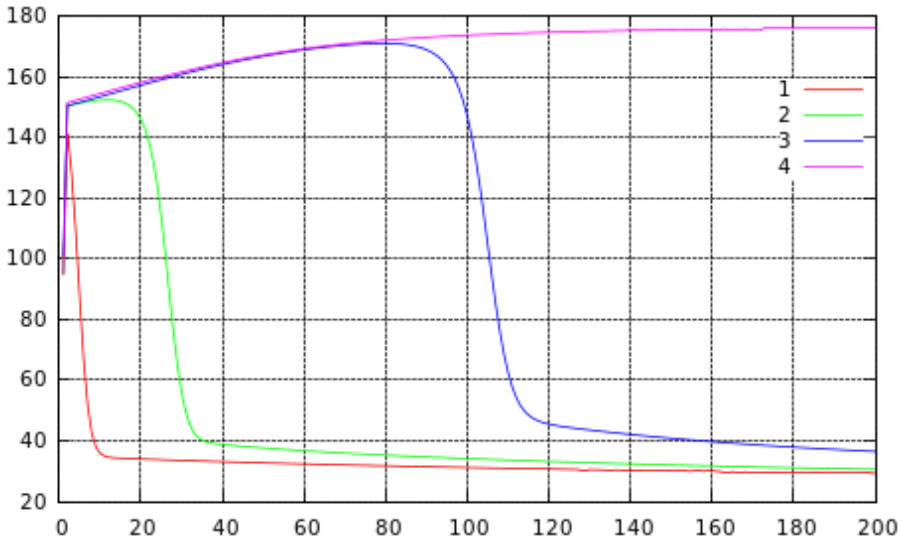
Tabulka 7.3: Chyba RMSE sítě BP v závislosti na velikosti koeficientu učení

7.4 Měření vlivu počtu skrytých vrstev

Cílem tohoto měření je zjistit, zda u sítě BP dochází ke zlepšení kvality predikci použitím více skrytých vrstev neuronové sítě na rozdíl od předchozích měření, kde byla výhradně používána pouze jedna skrytá vrstva. Měřena byla pouze neuronová síť BP s 2 vstupy, 10 neurony skryté vrstvy po dobu 200 epoch na stejných testovacích i trénovacích datech jako v předchozích případech. Z tabulky 7.4 vyplývá, že s rostoucím počtem skrytých vrstev nedochází ke zlepšení kvality predikce a navíc rychle roste doba učení sítě. Doba učení je uvedena v tabulce v závorce a je v sekundách. Na obrázku 7.7 lze pozorovat, že v případě dvou nebo tří vrstev dochází později ke konvergenci učení než u jedné v vrstvy. V případě 4 vrstev došlo k přetrénování, protože takové síti už nestačilo 200 epoch k úspěšnému učení.

Síť	**Počet skrytých vrstev**			
	1	**2**	**3**	**4**
BP	36,82 (9)	38,28 (24)	44,65 (40)	206,72 (55)

Tabulka 7.4: Chyba RMSE sítě BP v závislosti na počtu skrytých vrstev, v závorce je uveden časová délka měření zaokrouhlená na sekundy



Obrázek 7.7: Chyby RMSE podle počtu vrstev NN v závislosti na epoše učení

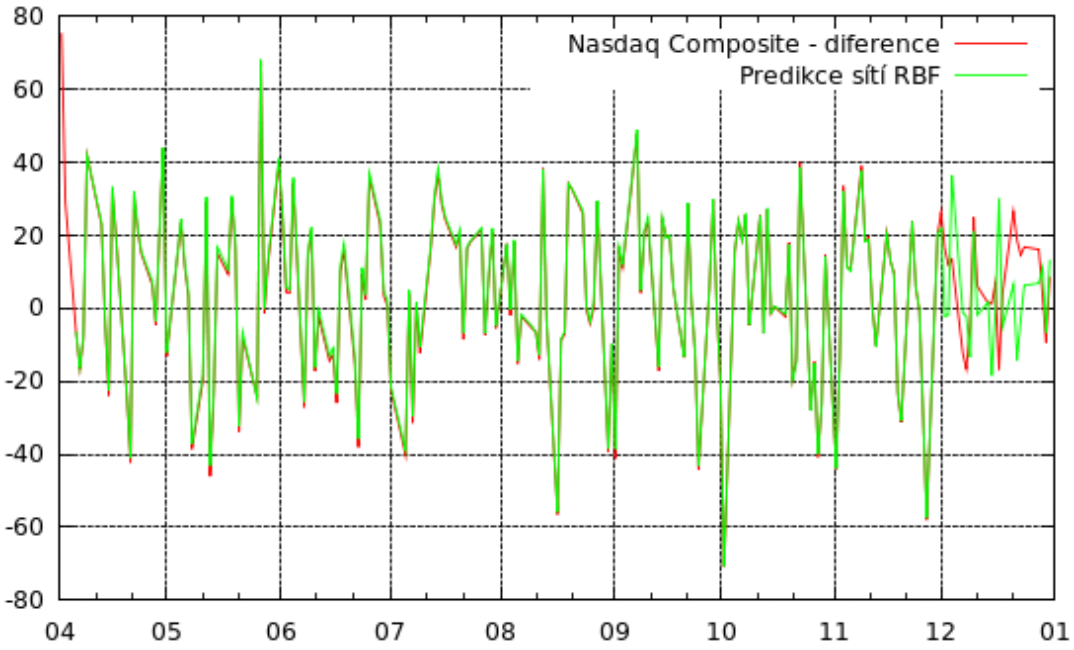
7.5 Měření diferencované časové řady

Diference časové řady způsobí odstranění konstantního trendu. Při předpovědi pak příliš nezáleží na příští konkrétní hodnotě, ale na tom, zda časová řada bude dále růst nebo klesat. Toto je pro finančního analytika důležitější informace, protože znamená, zda bude obchodník podle své strategie kupovat či prodávat. Z pohledu do tabulky 7.5 je zřejmé, že nejlepšího výsledku na testovacích datech dosáhla RBF síť s počtem neuronů 166, což je také velikost trénovací množiny. Na obrázku 7.8 pak lze pozorovat diferencovanou časovou řadu indexu Nasdaq Composite za období duben až prosinec 2009 a predikci této řady pomocí RBF sítě se 166 neurony skryté vrstvy. Trénovací množinou byla diferencovaná

data za období duben až listopad 2009. I přes nejlepší výsledek RBF sítě se 166 neurony skryté vrstvy nelze říci, že na použitých datech byla technika diference úspěšná. RBF síť už nedokáže, jak bylo naměřeno dříve, správně aproximovat data, na něž nebyla naučená.

Síť	**Počet neuronů skryté vrstvy**						
	2	**3**	**5**	**10**	**20**	**30**	**50**
BP	22,60	22,62	22,63	22,70	22,81	22,99	23,74
	2	**10**	**35**	**66**	**100**	**133**	**166**
RBF	27,11	38,70	34,64	22,47	16,26	16,78	5,89

Tabulka 7.5: Chyba RMSE testovacích dat diferencované časové řady



Obrázek 7.8: Diferencovaná časová řada vývoje trhu a její predikce pomocí RBF sítě

Kapitola 8

Závěr

Cílem této práce bylo testování vybraných parametrů a jejich vlivu na kvalitu predikce pomocí neuronových sítí. Testovány a srovnány byly výsledky měření počtu neuronů skryté vrstvy, počtu vstupů sítě, velikosti koeficientu učení, počtu skrytých vrstev a počtu skrytých neuronů při použití diferencovaných dat. Vedlejším produktem je aplikace v jazyce Python umožňující všechna tato měření provádět. Aplikace je primárně určena pro predikci vývoje finančních časových řad, ale je možné ji používat obecně pro libovolné časové řady.

Testováním bylo zjištěno, že pro běžnou časovou řadu jsou neuronové sítě BP a HO dobrou volbou. RBF síť dosahovala lepších výsledků na trénovacích datech, ale na testovacích neuspěla. RBF síť naopak prokázala lepší predikční schopnosti na diferencované časové řadě, ale s malou mírou schopnosti generalizace. Konkrétní volba počtu skrytých neuronů a počtu vstupů sítě ovšem vždy závisí na konkrétní časové řadě. Koeficient učení je vhodné volit uprostřed svého intervalu, ne extrémní hodnoty. Ukázalo se, že vyšší počet skrytých vrstev sítě nezlepšuje kvalitu predikce. Jedna skrytá vrstva tedy plně postačuje.

V práci je dále možné mnoha způsoby pokračovat. Lze rozvíjet vhodné počáteční nastavení parametrů bez přítomnosti experta s využitím evolučních algoritmů. Dále by bylo možné implementovat další typy neuronových sítí, například rekurentní a částečně rekurentní sítě jako je Jordanova a Elmanova síť. Pro učení by nemusel být použit jen algoritmus zpětného šíření chyby, ale existují i jiné algoritmy. V konkrétních případech, jako je například RBF síť, existuje mnoho různých algoritmů pro počáteční inicializaci parametrů RBF neuronů.

Tato práce se zabývala predikcí pomocí neuronové sítě zejména z technické stránky. Samotná výsledná aplikace slouží hlavně k měření různých parametrů, pro použití ke skutečnému obchodování na finančnímu trhu není vhodná. Tato skutečnost by také mohla být předmětem dalšího bádání a práce by byla již více ekonomicky zaměřená. Taková práce už by musela řešit obchodní strategii a rozhodovací mechanismus, zda v daný moment na burze nakupovat či prodávat a podobně. V takovém případě by nebyla kritériem úspěšnosti sítě pouze některá statistická chyba výstupu sítě, ale také ekonomické měřítko jako je ziskovost. Dalším rozšířením pro praktické použití na burze by mohla být možnost zadávání intervenčních proměnných, což je využití údajů přímo nesouvisejících s časovou řadou, ale majících vliv na její budoucí vývoj. Mohou to být například informace o růstu, poklesu, vývoji na jiných trzích, protože žádný trh nefunguje izolovaně.

Hlavní přínosem práce je vytvoření testovacích příkladů pro měření parametrů neuronových sítí a aplikace umožňující snadné provádění těchto měření.

Literatura

[1] Introduction To The Elliott Wave Principle. [online], 2006, [navštíveno 13.5.2010]. URL http://www.optionsoutlet.com/trading_technicals/elliot_wave_principle.html

[2] Dorffner, G.: Neural Networks for Time Series Processing. *Neural Network World*, ročník 6, 1996: s. 447–468.

[3] Giles, L.; Lawrence, S.; Tsoi, A. C.: Rule Inference for Financial Prediction using Recurrent Neural Networks. In *In Proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering (CIFEr)*, 1997, s. 253–259.

[4] Hu, Y. H.: *Handbook of Neural Network Signal Processing*. Boca Raton, FL, USA: CRC Press, Inc., 2000, ISBN 0849323592.

[5] Hutchinson, J.; Lo, A.; Poggio, T.: A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. *Journal of Finance*, ročník 49, 1994: s. 851–889.

[6] Kaastra, I.; Boyd, M.: Designing a Neural Network for Forecasting Financial and Economic Time Series. *Neurocomputing*, ročník 10, 1996: s. 215–236.

[7] Lomovciv, P.: *Predikce pomocí neuronové sítě*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2005.

[8] Masters, T.: *Practical neural network recipes in C++*. San Diego, CA, USA: Academic Press Professional, Inc., 1993, ISBN 0-12-479040-2.

[9] Moody, J.; Levin, U.; Rehfuss, S.: Predicting the U.S. Index of Industrial Production (Extended Abstract). In *In proceedings of the 1993 Parallel Applications in Statistics and Economics Conference, Zeist, The Netherlands. Special issue of Neural Network World*, 1993, s. 791–794.

[10] Newsgroup: Neural Networks FAQ. [online], 2002, [navštíveno 13.5.2010]. URL ftp://ftp.sas.com/pub/neural/FAQ.html

[11] Orr, M. J. L.: Introduction to radial basis function networks. Technická zpráva, Centre for Cognitive Science, University of Edinburgh, 1996.

[12] Orsaghová, J.: *Predpovedanie finančných časových radov pomocou neurónových sietí*. Diplomová práce, Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, 2009.

[13] Zbořil, F. V.: Soft Computing. [online], 2010, [navštíveno 13.5.2010].
URL http://www.fit.vutbr.cz/study/courses/SFC

[14] Zbořil, F. V.: Základy umělé inteligence. [online], 2010, [navštíveno 13.5.2010].
URL http://www.fit.vutbr.cz/study/courses/IZU

Seznam použitých zkratek a symbolů

$\vec{i}$	vstupní vektor sítě
$\vec{d}$	požadovaný výstupní vektor sítě
$\vec{o}$	výstupní vektor sítě
$\vec{x}$	vstupní vektor vrstvy
$\vec{y}$	výstupní vektor vrstvy
$\vec{c}$	střed hyperkoule RBF neuronu
E	chyba sítě
w_{ij}^{L}	váha i-tého vstupu j-tého neuronu vrstvy L
u	bázová funkce neuronu
f	aktivační funkce neuronu
α	momentum
η	koeficient učení
σ	poloměr hyperkoule RBF neuronu
θ	práh neuronu
BP	zpětné šíření chyby (backpropagation), pro síť backpropagation
HO	vyšší řád (higher order), pro neuronovou síť vyššího řádu
MAD	aritmetický průměr absolutních odchylek (mean absolute deviation)
MSE	střední kvadratická chyba (mean square error)
NN	neuronová síť (neural network)
RBF	radiální bázová funkce (radial basis function)
RMSE	druhá odmocnina střední kvadratické chyby (root mean square error)

Dodatek A

Obsah CD

A.1 Seznam složek

application/doc	programová dokumentace aplikace
application/examples	testovací příklady
application/src	zdrojové soubory aplikace
report	technická zpráva
report/src	zdrojové kódy technické zprávy

A.2 Seznam zdrojových souborů aplikace

bp.py	implementace tříd neuronové sítě backpropagation
ho.py	implementace tříd neuronové sítě vyššího řádu
error.py	implementace tříd pro zpracování statistických chyb
main.py	implementace rozhraní pro spuštění z příkazové řádky
network.py	implementace základních tříd pro neuronové sítě
prediction.py	implementace třídy zastřešující práci s neuronovou sítí
rbf.py	implementace tříd neuronové sítě s radiálními bázovými funkcemi
samples.py	implementace tříd pro práci s časovou řadou

A.3 Seznam testovacích příkladů

difference.py	test vlivu počtu neuronů skryté vrstvy diferencovaných dat
hidden.py	test vlivu počtu neuronů skryté vrstvy
input.py	test vlivu počtu vstupů sítě
layer.py	test vlivu počtu skrytých vrstev sítě
learningrate.py	test vlivu velikosti koeficientu učení

Dodatek B

Manuál

Aplikace pro predikci vývoje finančních trhů, ale i obecně pro časové řady, je implementována v skriptovacím jazyce Python 2.6.4. Vyvíjel jsem na platformě PC v 32 bitovém operačním systému Ubuntu Linux 9.10 s jádrem 2.6.21-31. Implementace byla testována pouze na této platformě. Teoreticky nic nebrání použití na jiných platformách, ale pro generování obrázků grafů je využíván software Gnuplot 4.2. Uživatel by si tedy na jiné platformě musel vizualizovat data sám, nebo zajistit funkčnost programu Gnuplot v příkazové řádce.

Jsou dva způsoby jak trénovat a testovat neuronovou síť. První a pohodlnější možností je napsat velmi jednoduchý skript v Pythonu, který pouze vytvoří instanci třídy `Prediction` a pomocí jejích metod nastaví konkrétní neuronovou síť a její parametry. Tento způsob díky možnostem jazyka Python umožňuje snadné hromadné zpracování více měření najednou. Druhý způsob je spuštění skriptu `main.py` z příkazové řádky, bohužel toto řešení je méně přehledné vzhledem k velkému počtu parametrů.

B.1 Popis měření přes instanci třídy Prediction

Tento způsob práce s aplikací je pohodlnější, nicméně vyžaduje minimální znalost jazyka Python. Ovšem pro člověka, který umí programovat, nebude problém pochopit použití tohoto způsobu z ukázkových příkladů. Kompletní ukázky jsou k vidění v testovacích příkladech. Díky zpracování skriptovacím jazykem je možné snadno provádět hromadná měření. Například vytvořit pole hodnot jednoho parametru a v cyklu pak vytvářet instance třídy `Prediction` pokaždé měřit s jinou hodnotou parametru.

Na úvod je potřeba si naimportovat do skriptu třídu `Prediction`. Často testujeme v jiném umístění, než máme zdrojové soubory aplikace. Proto před importem modulu můžeme ještě přidat do importovacích cest nová umístění.

```
import sys
sys.path.append("../")     # pro nadřazenou složku
from prediction import Prediction
```

Vytvoříme instanci třídy `Prediction`. Parametrem je název CSV souboru s daty. Podrobnější popis zdrojového souboru CSV souboru je v sekci B.3.

```
p = Prediction("nasdaq.csv")
```

Dále se soustředíme na popis jednostlivých metod a jaké parametry lze měnit. V případě zadání neplatných hodnot se skript ukončí a upozorní uživatele na chybu.

```
p.setTrainDataInterval("2009-04-01", "2009-11-30")
```

Tato metoda nastavuje časový interval, jehož vzorky se použijí pro učení.

```
p.setTimeSerieFileName("timeserie.dat")
```

Určuje název souboru pro uložení časové řady, která byla použita pro učení.

```
p.setNeuralNetwork("bp", 2, 5, 1, 1)
```

Nastavuje typ neuronové sítě a její parametry. V tomto případě jde o síť backpropagation se dvěma vstupy, pěti neurony skryté vrstvy a jedním neuronem výstupní vrstvy. Poslední parametr je počet skrtých vrstev sítě, ale může být vynechán. Výchozí je 1 vrstva.

```
p.setEpochCount(100)
```

Nastavuje počet epoch učení neuronové sítě.

```
p.setLearningRate(0.1)
```

Nastavuje hodnotu koeficientu učení.

```
p.setMomentum(0.6)
```

Nastavuje hodnotu hybnosti učení.

```
p.setErrorFileName("error.dat")
```

Nastavuje název souboru pro uložení statistických chyb v závislosti na počtu epoch učení.

```
p.setPredictionInterval("2009-04-01", "2010-12-31")
```

Zde volíme interval, v němž chceme predikovat vývoj časové řady. Pokud neexistuje koncové datum v časové řadě, pak je vytvořen vzorek navíc s tímto datem. Aplikace předpokládá predikci maximálně jednoho dne dopředu.

```
p.setPredictionFileName("prediction.dat")
```

Do tohoto souboru bude uložena predikce časové řady v závislosti na čase.

```
p.setDifference("difference.dat")
```

Tato metoda uloží diferenci původní časové řady do určeného souboru a zároveň aplikace bude s diferencí pracovat místo původního signálu.

```
p.setInitialWeightInterval(-0.2, 0.2)
```

Nastaví interval pro generování počátečních vah neuronů.

```
p.run()
```

Tato metoda spustí samotné vytvoření sítě dle zadaných parametrů, její učení a předpověď. Vygeneruje soubory s vypočtenými údaji.

```
p.createGraphs("graph.plt", "timeserie.png", "prediction.png",
    "both.png", "mse.png", "rmse.png")
```

Tato metoda vyžaduje pro svou činnost program Gnuplot spustitelný z příkazové řádky. Vygeneruje PLT soubor, což je skript pro vizualizér Gnuplot. Následně dojde k vytvoření PNG obrázků grafů časové řady, její predikce, grafu, v němž je jak původní časová řada, tak predikce zároveň, grafů chyb MSE a RMSE. Pokud uživatel o některý z obrázků nemá zájem, stačí vyplnit prázdný řetězec namísto názvu soubor pro konkrétní obrázek.

B.2 Popis měření přes parametry příkazové řádky

Použití aplikace jako spuštění skriptu se zadáním potřebných hodnot jako parametrů je sice možné, ale poněkud nepohodlné. Nicméně je podporováno pro případ, že by uživatel chtěl používat vlastní skripty. Pro spuštění predikce se používá skript `main.py`. Příklad spuštění skriptu je následující:

```
./main.py dowjones.csv timeserie.dat prediction.dat error.dat
```

První argument je zdroj dat, druhý je soubor pro uložení původního signálu, třetí je pro uložení predikce a čtvrtý pro uložení statistických chyb. Dále pokračují jednotlivé parametry, vysvětleny jsou stručně.

--tf datum začátku trénovacích dat

--tt datum konce trénovacích dat

--pf datum začátku testovacích dat

--pt datum konce testovacích dat

-i počet vstupů neuronové sítě

-o počet výstupů neuronové sítě

-n počet skrytých neuronů

--hl počet skrytých vrstev

-m momentum

-l koeficient učení

-e počet epoch

-d soubor pro uložení diferencovaného signálu a jeho použití

--wlb dolní hranice intervalu pro generování počátečních vah

--wub horní hranice intervalu pro generování počátečních vah

B.3 Zdroje dat pro aplikaci

Aplikace je zaměřená na práci s finančními trhy, ale jako zdroje dat je možné použít libovolné časové řady. Vnitřní struktura je klasické CSV, v prvním sloupci je datum v anglickém formátu, ve druhém sloupci je hodnota časové řady. Tyto hodnoty jsou odděleny čárkou. Každý záznam je oddělen znakem konce řádku. Sehnat ovšem reálná data vývoje finančního trhu není tak jednoduché. Na internetu je mnoho portálů, které se zabývají distribucí jak aktuálních denních vývojů trhů, nebo také historií vývoje trhů. Často však jsou placené nebo vyžadují instalaci specializovaného softwaru. Jako zdroj volně dostupných dat v CSV jsem používal stránky http://finance.yahoo.com/. Z těchto stránek jsem stahoval historická data pro indexy Dow Jones Industrial Average a Nasdaq Composite ve formátu CSV. Smluvní podmínky portálu Yahoo zakazují redistribuci získaných dat, proto si je zájemce musí stáhnout sám.