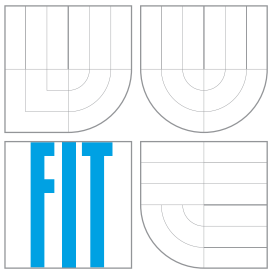
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PŘEDPOVĚĎ POČASÍ NA ZÁKLADĚ RŮZNÝCH ZDROJŮ

WEATHER FORECAST BASED ON DIFFERENT SOURCES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN HOŘÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ NOVOTNÝ

BRNO 2014

Abstrakt

Bakalářská práce se zabývá předpovědí počasí prostřednictvím umělé neuronové sítě s algoritmem učení zpětného šíření chyby. Předpověď je založená na datech získaných z volně dostupných služeb nabízejících informace o počasí. Vytvořená aplikace stahuje a ukládá předpovědi ze serverů a vytváří novou předpověď. Výsledky predikce jsou srovnávány s realitou a původními službami.

Abstract

Bachelor thesis deals with the weather forecast through the artificial neural network with backpropagation method. The forecast is based on data obtained from freely accessible services offering weather information. The created application downloads and saves the forecasts from servers and creates a new forecast. The results of the prediction are being compared with the reality and the original services.

Klíčová slova

předpověď počasí, časová řada, umělé neuronové sítě, python, sqlite, django framework, json, xml, cron, numpy, opencv, orm

Keywords

weather forecast, time series, artificial neural networks, python, sqlite, django framework, json, xml, cron, numpy, opencv, orm

Citace

Martin Hořák: Předpověď počasí na základě různých zdrojů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Předpověď počasí na základě různých zdrojů

Prohlášení

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

.....
Martin Hořák
21. května 2014

Poděkování

Zde bych chtěl poděkovat vedoucímu bakalářské práce Ing. Tomáši Novotnému za odborné vedení a cenné rady. Dále bych chtěl také poděkovat Ing. Štěpánu Mráčkovi za uvedení do problematiky neuronových sítí.

Obsah

Kapitola 1

Úvod

Předpovídat počasí nejde, rozhodně ne na dlouhou dobu dopředu. Jde ale odhadnout s určitou nepřesností budoucí průběh počasí založený na předchozím vývoji. Ze složitého matematického simulačního modelu, který čerpá z naměřených dat, zkušeností meteorologa, případně dalších nástrojů, lze vyhodnotit, jak se bude počasí chovat v následujících hodinách, případně dnech. V nejbližších dnech se obvykle predikce přibližuje realitě, avšak s přibývající délkou přesnost predikce klesá.

Jedním ze způsobů jak nahlížet na počasí je převedení jednotlivých parametrů počasí na stochastické časové řady, které jsou reprezentovány posloupností hodnot. Rozdělením jednotlivých parametrů počasí jdeme proti celému systému počasí, kde všechny parametry se navzájem ovlivňují, ale na druhou stranu nám to umožňuje predikovat jednotlivé vlastnosti počasí, i když na úrok přesnosti.

Předmětem této práce je vytvořit aplikaci, která bude schopna vytvořit předpověď počasí, která vychází z již vytvořených předpovědí. Aplikace bude získávat předpovědi rozložené do jednotlivých parametrů počasí z volně dostupných služeb, které nabízejí informace o počasí. Stažené informace o předpovědích budou reprezentovány jako časové řady hodnot, ze kterých budou pomocí vybrané metody umělé inteligence vytvořeny predikce.

Pro predikci časových řad bude použita umělá neuronová síť. Jako vhodný typ neuronové sítě pro predikci bude zvolena dle [6] dopředná neuronová síť se sigmoidní aktivační funkcí. Neuronové sítě jsou vhodným nástrojem pro predikci, protože jsou univerzálními aproximátory funkcí a jsou schopny objevit v časových řadách nelineární závislosti [5].

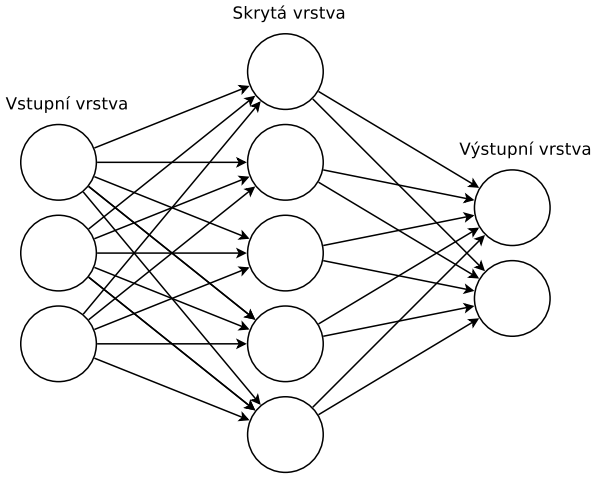
Práce je rozdělena do několika kapitol. V první části (2) je objasněna struktura a funkce neuronové sítě s učením zpětného šíření chyby, pojem časové řady a přehled zdrojů, které budou použity při implementaci aplikace. Další kapitola je věnována návrhu (3) vznikající aplikace. Důležitou částí této práce je také kapitola zabývající se realizací aplikace (4), kde jsou uvedeny jednotlivé implementační detaily. Závěr této práce se zabývá konfigurací (5.1) neuronové sítě, kde je snaha naleznouti vhodné nastavení sítě, tak aby predikce počasí byla co nejpřesnější. Nakonec je uvedeno srovnání (5.2) vytvořené predikce s existujícími předpověďmi. Práce rovněž obsahuje návod (B) k instalaci a použití vytvořené aplikace.

Kapitola 2

Teoretická část

2.1 Neuronové sítě

Neuronové sítě jsou tvořeny neurony, které jsou mezi sebou propojeny spoji ohodnocenými váhami. Po předložení trénovacích dat jsou schopny se učit novým funkcím vycházejícím z těchto dat. Důležitými vlastnostmi neuronových sítí je schopnost hledat závislosti v trénovacích datech a poměrně správně reagovat na neznámá vstupní data. V této kapitole se budeme konkrétně zabývat vícevrstvou perceptronovou sítí, která při učení využívá *metodu zpětného šíření (backpropagation)*. Kapitola také obsahuje popis struktury neuronové sítě, jednotlivých neuronů, popis bázové a přenosové funkce neuronu nebo také proces učení.



Obrázek 2.1: Schéma vícevrstvého perceptronu.

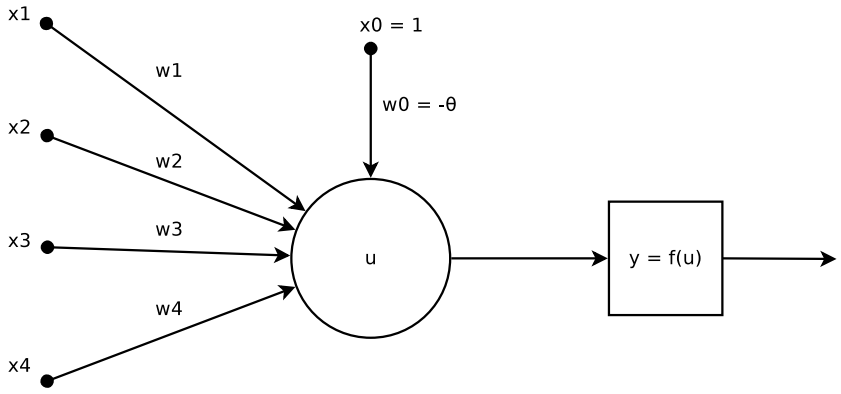
2.1.1 Popis sítě

Umělá neuronová síť je formálním modelem k biologické neuronové síti. Je to systém, který se skládá z výpočetních jednotek – neuronů. Jedná se o dopřednou (feedforward) síť, což znamená, že se signál šíří dopředu od vstupní vrstvy po výstupní. Pro učení sítě je použit algoritmus backpropagation, kdy se chyba výstupní vrstvy zpětně šíří do předchozích vrstev. Síť je tvořena několika vrstvami, vstupní a výstupní vrstvou. Mezi těmito vrstvami se nachází jedna nebo více skrytých vrstev. Vstupy jedné vrstvy jsou napojeny na výstupy

vrstvy předchozí tak, že tvoří *bipartitní graf* [11], což znamená napojení výstupu jednoho neuronu do všech vstupů neuronů v následující vrstvě. Žádné další vazby mezi neurony neexistují. Vstupní vrstva slouží pouze k rozšíření vstupních hodnot. Schéma je uvedeno na obrázku 2.1.

2.1.2 Neuron

Základní jednotkou modelu neuronové sítě je neuron. Vzorem umělého neuronu je biologický neuron, který se nachází v lidském mozku. Má velký počet vstupů – dendridů a jeden výstup – axiom, který se dál může větvit. Propojení mezi dendridem jednoho neuronu a axiomem druhého je nazýváno synapse. Na základě tohoto neuronu byl vytvořen umělý neuron. Základní model neuronu nese jméno po svých tvůrcích: McCulloch-Pittsův model neuronu [8]. Jeho struktura je znázorněna na obrázku 2.2. Neuron obsahuje n vstupů $x_1, ..., x_n$, kde



Obrázek 2.2: McCulloch-Pittsův model neuronu.

každý vstup je ohodnocen váhou $w_1, ..., w_n$, které reprezentují lokální paměť neuronu. Čím větší váhu vstup má, tím je důležitější. Neuron je rovněž obdařen prahem θ. Výstup y má neuron pouze jeden. Ten je výsledkem výpočtu aktivační funkce, která má na vstupu výsledek bázové funkce aplikované na vstupy neuronu [12].

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

y výstup

$\vec{x}$ vektor vstupů

$f()$ aktivační funkce

$g()$ bázová funkce

Jak vstupy tak i jednotlivé váhy u neuronů jsou reálná čísla.

Bázová funkce

Tato funkce vyjadřuje vážený součet vstupů a jejich vah označován jako *potenciál neuronu*. U sítě s učením *backpropagation* se používá lineární bázová funkce.

$$u = \sum_{i=1}^{n} (w_j * x_j) + \theta \tag{2.2}$$

Někdy je prah považován za speciální vstup od neexistujícího neuronu, kdy $x_0 = -1$ je vstupní hodnota a prah je její váha $w_0 = \theta$. Pak vztah bázové funkce je:

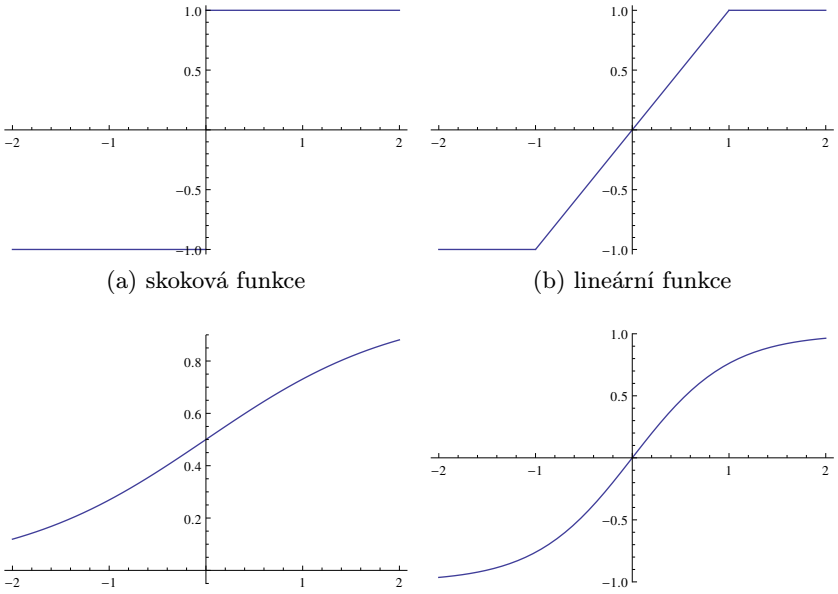
$$u = \sum_{i=0}^{n} (w_j * x_j) \tag{2.3}$$

Přenosová funkce

Někdy je tato funkce označována jako aktivační. Přenáší vnitřní potenciál neuronu na výstup:

$$y = f(u) \tag{2.4}$$

Dle problému, který neuronovými sítěmi řešíme, můžeme vybrat ze čtyř typů aktivačních funkcí [12]:



(a) skoková funkce (b) lineární funkce

(c) sigmoida (d) hyperbolický tangens

Obrázek 2.3: Průběhy přenosových funkcí.

- Nespojitá (skoková) funkce, viz 2.3a

$$f(u) = \begin{cases} a & \text{pro } u < \theta \\ b & \text{pro } u > \theta \\ y & \text{pro } u = \theta \end{cases}$$

- Po částech lineární funkce, viz 2.3b

$$f(u) = au + b$$

- Spojitá funkce – Sigmoida, viz 2.3c

$$f(u) = \frac{1}{1 + e^{-u}}$$

- Spojitá funkce – Hyperbolický tangens, viz 2.3d:

$$f(u) = tgh(u)$$

2.1.3 Učení sítě

Aby síť byla schopna správně fungovat, je potřeba vhodně nastavit váhy u jednotlivých neuronů. Tento proces se nazývá učení nebo někdy trénování neuronové sítě. Přiložením hodnot na vstupní neurony sítě jsou signály pomocí přenosových funkcí šířeny až k výstupním neuronům. Vstupní vrstva slouží pouze k distribuci hodnot k neuronům první skryté vrstvy. Potenciály jednotlivých neuronů jsou počítány postupně od první až po poslední vrstvu tak, aby vypočítané hodnoty předchozí vrstvy mohly sloužit jako vstupní hodnoty následující vrstvy. Výsledné hodnoty poslední vrstvy jsou přímo výstupními hodnotami. Je třeba zmínit, že se jedná o učení s učitelem, tedy že síť má při učení k dispozici i množinu požadovaných hodnot. Během učení je snaha dosáhnout co nejmenší odchylky výstupních hodnot sítě od požadovaných hodnot. Celková chyba sítě je definována v [11]:

$$E = \sum_{k} E_k, \tag{2.5}$$

kde E_k je chyba k-tého prvku trénovací množiny, která je definována v [11]:

$$E_k = \frac{1}{2} \sum_{i} (y_i - d_{kj}), \tag{2.6}$$

kde y je hodnota jednoho výstupního neuronu a d je požadovaná hodnota. Po předložení všech prvků trénovací množiny je vypočtena celková chyba sítě a dle této chyby jsou upraveny váhy jednotlivých neuronů podle vztahu z [11]:

$$\delta w_{ij} = -\eta \frac{\delta E}{\delta w_{ij}}, \tag{2.7}$$

kde η je parametr rychlosti učení. Tento krok se opakuje dokud chyba není menší než zadaná mez.

Rozšířením tohoto postupu je *metoda zpětného šíření (backpropagation)*. Při učení touto metodou dochází k postupnému šíření chyby od výstupních neuronů až po vstupní. Na rozdíl od klasického učení, kdy se váhy upravují až po předložení všech prvků trénovací množiny, zde dochází k úpravě vah již během učení.

Parametry ovlivňující učení

V rámci učení neuronové sítě existuje několik parametrů, které mají vliv na učení sítě. Při vhodném nastavení těchto parametrů je možné dosáhnout dobrých výsledků, ale rovněž při nesprávném použití může dojít k přeučení sítě.

Parametr rychlosti učení η Upravuje rychlost učení. Obvykle se pohybuje kolem hodnoty blízké nule. Vyšší hodnota parametru sice může urychlit učení, ale ovšem na úkor správného výsledku.

Počet neuronů ve skryté vrstvě Nelze předem určit a závisí na typu úlohy, která je řešena.

Moment učení Tento parametr definuje rozdíl vah mezi iteracemi. Zabraňuje uvíznutí v lokálních extrémech [4].

Nastavení vah neuronů Pro počáteční nastavení vah neuronů se používá *gradientní metoda* [11]. Váhy jsou nastaveny náhodně v rozsahu $\langle -\frac{2}{s}, \frac{2}{s} \rangle$, kde s je počet vstupních neuronů.

Počet vrstev Podobně jako počet skrytých neuronů nelze určit předem, ale pouze experimentálním postupem. Počet závisí na typu řešené úlohy.

Normalizace hodnot

Neuronové sítě nebo přesněji neurony pracují s realnými čísly. Aby síť byla schopna správně plnit svou funkci je třeba, aby hodnoty, které jsou předkládány na vstupy sítě, byly ve vhodném intervalu na rozdíl od hodnot v časové řadě, kde mohou být jakékoliv. Tento interval se odvíjí dle použité přenosové funkce. Například pro sigmoidní přenosovou funkci je vhodný vstupní interval $\langle -1, 1 \rangle$, kdy dochází k největší změně hodnot, zatímco výstupní interval by měl být $\langle 0, 1 \rangle$, což odpovídá jejímu oboru hodnot. Normalizace vstupních dat pro rozsah $\langle -1, 1 \rangle$ je dle vzorců z [3]:

$$m = \frac{max(Y) + min(Y)}{2} \tag{2.8}$$

$$r = max(Y) - min(Y) \tag{2.9}$$

$$S_i = \frac{Y_i - m}{r/2} \tag{2.10}$$

m střed rozsahu

r rozsah

Y_i prvek vstupní časové řady

S_i normalizovaný prvek vstupní časové řady

Normalizace výstupních dat pro rozsah $\langle 0, 1 \rangle$ je dle vzorce z [3]:

$$S_i = \frac{Y_i - min(Y)}{max(Y) - min(Y)} \tag{2.11}$$

Y_i prvek výstupní časové řady

S_i normalizovaný prvek výstupní časové řady

2.2 Predikce časových řad

Předpovídání budoucího pokračování časové řady vychází z pozorování jejího minulého průběhu. Cílem predikce je z historického průběhu řady naučit vhodný matematický aparát předpovídat budoucí vývoj řady. Čím více dopředu se snažíme předpovídat, tím se kvalita předpovědi snižuje.

Časová řada je posloupnost vektorů, které závisí na čase [6]:

$$\vec{x}(t), t = 0, 1, 2, ..., n \tag{2.12}$$

Prvky vektoru mohou být jakékoliv měřitelné jevy:

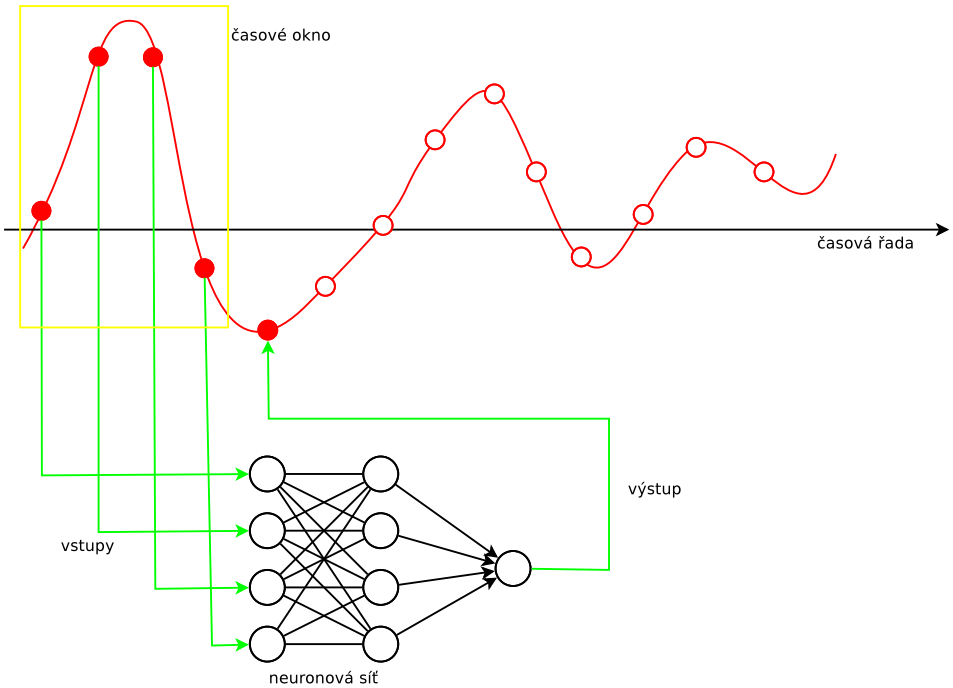
- teplota vzduchu

- spotřeba elektrické energie
- cena na burze

Časové řady, u kterých můžeme přesně určit jaký budou mít průběh, lze označit jako deterministické. Avšak u většiny řad se jedná o stochastické procesy, protože obsahují ruchy, které mohou jakkoliv měnit jejich průběh. Jedná se o spojitou funkci času, ale z praktických důvodů ji bereme jako diskrétní, kde časové úseky mají fixní velikost [6].

2.2.1 Metoda klouzajícího okna

Jako standardní metoda pro tvorbu trénovací množiny se používá technika klouzajícího okna [7]. Někdy označováno jako metoda časového okna. Máme časovou řadu $x(t)$, kde $t = 1, 2, 3, ..., n$ a neuronovou síť s k vstupy a jedním výstupem. Trénovací množina nám vznikne tak, že vezmeme několik prvních hodnot časové řady $x(t-1), x(t-2), x(t-3), ..., x(t-k)$ a dáme je na vstup neuronové sítě. Počet hodnot k musí odpovídat počtu vstupních neuronů sítě. Na výstup sítě umístíme následující hodnotu $x(t)$ jako očekávaný výstup. Takto vznikne časové okno. Posunem tohoto okna po celé časové řadě vznikne soubor položek trénovací množiny, které obsahují vstupní a výstupní vzorky pro trénování sítě. Metoda klouzajícího časového okna je znázorněna na obrázku 2.4.



Obrázek 2.4: Klouzající okno.

Vstupní část klouzajícího okna je možné rozšířit o další doplňující informace, které reprezentují okolnosti průběhu časové řady. Toto rozšíření se nazývá intervenční proměnné.

2.2.2 Přesnost predikce

Pro kontrolu přesnosti predikce časových řad existuje několik statistických metod, které jsou založeny na porovnávání predikovaných a reálných hodnot. Jednotlivými metodami

jsou [9]:

- Střední kvadratická chyba (MSE – Mean Square Error)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(d_i - x_i)^2 \tag{2.13}$$

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

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|d_i - x_i| \tag{2.14}$$

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

$$RMSE = \sqrt{\frac{1}{\sigma^2 N}\sum_{i=1}^{N}(d_i - x_i)^2} \tag{2.15}$$

d_i reálná hodnota

x_i predikovaná hodnota

N počet vstupních hodnot

σ rozptyl

2.3 Zdroje dat předpovědí počasí

Základem pro vytvoření nové předpovědi počasí je získání již existujících předpovědí. Aktuálně existuje mnoho služeb, které se na tyto služby zaměřují, a to jak na celosvětové úrovni, tak existují i servery, které se zaměřují speciálně na Českou republiku. Zjednodušeně to funguje tak, že tyto servery sbírají data z meteorologických stanic z celého světa (případně jen určité oblasti) a dle vlastních metod vytvářejí předpověď počasí. Tuto předpověď nabízejí pro běžné uživatele přes grafické webové rozhraní nebo v textové, serializované podobě, kterou lze programově zpracovávat. Jsou přístupné přes webové rozhraní prostřednictvím protokolu HTTP. Kromě předpovědi většina služeb nabízí i skutečně naměřená data.

2.3.1 Formát dat

Všechny prostudované služby nabízejí data v serializačním formátu Javascript Object Notation (JSON) nebo Extensible Markup Language (XML).

Javascript Object Notation (JSON)

Jedná se o často používaný serializační formát. Jeho syntaxe vychází z jazyku Javascript, jehož je podmnožinou. Formát je to textový, a proto je na programovacím jazyku nezávislý. Součástí většiny jazyků jsou knihovny, které umí pracovat s tímto formátem (například simplejson[1] pro jazyk Python nebo json-c[2] pro jazyk C).

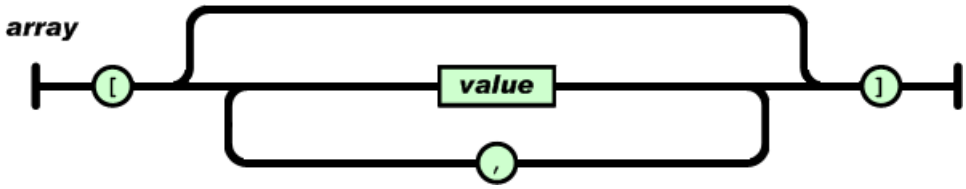
[1] https://pypi.python.org/pypi/simplejson/
[2] https://github.com/json-c/json-c/wiki

Základem JSONu jsou dvě struktury [1] objekt a pole. Objekt je neuspořádaná množina položek, které jsou tvořeny dvojicí *klíč: hodnota*. V programovacích jazycích je objekt obvykle reprezentován strukturou, slovníkem nebo záznamem. Na druhou stranu pole je uspořádaná množina hodnot, která je v programovacích jazycích realizována jako seznam, pole nebo vektor.

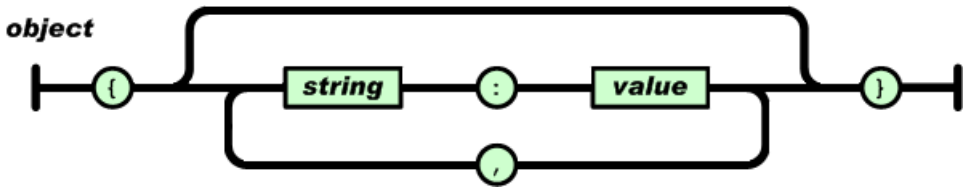
Hodnota se může rovnat datovým typům:

- řetězec
- číslo
- objekt
- pole
- true
- false
- null

Tím, že hodnota může být zase objekt (obrázek 2.6) nebo pole (obrázek 2.5), je umožněno vnořování těchto struktur.



Obrázek 2.5: JSON pole [1].



Obrázek 2.6: JSON objekt [1].

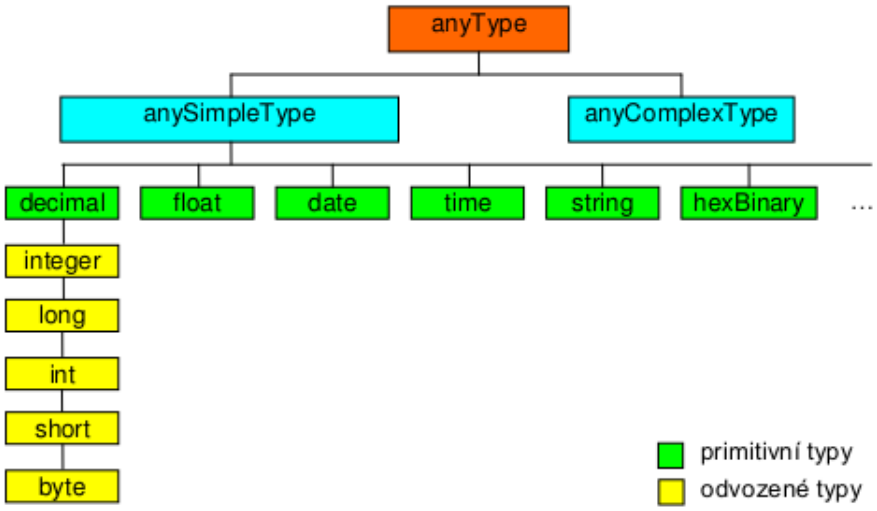
Extensible Markup Language (XML)

Tento serializační formát patří mezi značkovací jazyky. Jedná se o textový formát většinou kódovaný v UTF-8. Struktura dokumentu je volitelná. U XML lze jednotlivé značky libovolně pojmenovat. Avšak tyto vlastnosti nejsou vhodné při výměně XML zpráv, protože je potřeba znát jejich strukturu. Za tímto účelem vznikly jazyky, kterými lze popsat syntaxi nového jazyka, který má syntaxi XML (XHTML, SVG, KML). Mezi první jazyky, které se začali používat pro popis XML, patří DTD (Document Type Definition), který má však několik nedostatků jako nepodpora jmenných prostorů nebo omezení podpory datových

typů. Novějším jazykem, který řeší tyto nedostatky, je XSD (XML Schema Definition). XSD definuje [10]:

- prvky
- atributy
- potomky prvků
- pořadí potomků
- počet potomků
- zda je prvek prázdný nebo může obsahovat hodnotu
- typy dat prvků a atributů
- výchozí a pevné hodnoty prvků a atributů

Datové typy formátu XML jsou rozděleny do dvou kategorií – jednoduché a složené. Přičemž složené se mohou skládat z těch jednoduchých a připomínají struktury z programovacích jazyků. XML rovněž umožňuje definování si vlastních datových typů, které mohou vycházet ze zabudovaných typů (obrázek 2.7).



Obrázek 2.7: Neúplné zobrazení zabudovaných datových typů [10].

2.3.2 Délka předpovědi

Při výběru služeb hrála velkou roli délka předpovědi. Většina služeb nabízí předpověď na 36 hodin, na 3 dny, a nebo až na 10 dní dopředu. Předpovědi na delší dobu jsou obvykle hodně obecné a neurčité, a proto je vhodnější vybrat služby s předpovědí na několik hodin dopředu, které je poměrně propracovaná a detailní. Tyto kratší předpovědi jsou servery nabízeny v časových intervalech po jedné až třech hodinách.

2.3.3 Sledované parametry počasí

Jednotlivé záznamy o předpovědi obsahují poměrně velké množství parametrů a ne všechny se z vybraných serverů shodují s ostatními, proto je nelze porovnávat. Některé parametry předpovědi je potřeba převést na stejné jednotky například rychlost větru. Některé služby používají jednotku míle za sekundu jiné kilometr za hodinu. Jako sledované parametry bylo vybráno těchto šest základních:

- vlhkost
- teplota
- počet srážek
- rychlost větru
- směr větru
- oblačnost

2.3.4 Použité služby

Jako zdroje předpovědi byly vybrány čtyři webové služby, jejichž dostupnost serverů se značně liší. Obecně všechny služby nabízející předpovědi počasí umožňují dva přístupy k jejich serverům. Na výběr jsou buď zpoplatněné účty, a nebo účty zdarma. Omezení je především zaměřeno na počet dotazů na server za hodinu případně za celý den. Zpoplatněných verzí je několik (obvykle 2–3). Cena účtu se odvíjí od počtu povolených dotazů na server. U účtů, které jsou úplně zdarma, je počet dotazů na server striktně omezen na určitý počet. Navíc dostupnost serverů se pohybuje kolem 95 %, což může stahování dat občas zkomplikovat.

Wunderground Weather Underground je americká služba, která ale nabízí předpovědi počasí zaměřené na celý svět. Vlastní přes 34 tisíc meteorologických stanic, ze kterých sbírá data. Předpovědi vytváří na vlastním systému nazývaném Best Forecast™. Data z jejich serverů jsou dostupné ve formátech JSON i XML.

OpenWeatherMap Jedná se o začínající projekt v oblasti předpovídání počasí. Služba OpenWeatherMap prosazuje myšlenku volně dostupných informací o počasí po vzoru OpenStreetMap[3], která nabízí volně dostupné geografické data. Nevlastní meteostanice, ale pro tvorbu předpovědi používá nezpracovaná data z oficiálních meteorologických předpovědí, stanic, radarů, které jsou dostupné. Svá data nabízí ve formátu JSON.

Yr.no Norská služba yr.no, která je součástí Norského Meteorologického Institutu, patří mezi největší na světě. Pro své předpovědi čerpá data z více jak 7 milionů míst. Data nabízí zdarma ve formátu XML.

Hamweather Přesněji jde o službu AerisWeatherAPI, kterou nabízí web Hamweather.com spolu s dalšími produkty zaměřenými na počasí. Data nabízejí pouze ve formátu JSON.

Náhledy odpovědí ze serverů ve formátech JSON a XML jsou umístěny v příloze C.

[3] http://www.openstreetmap.org/

Kapitola 3

Návrh

Dle zadání byla navržena aplikace, která se bude skládat ze tří částí, jejíž jednoduché schéma je na obrázku 3.1. Nejprve bude potřeba získat předpovědi z dostupných serverů. K tomuto účelu byla navrhnuta serverová část aplikace. Tato část nebude klasickou serverovou aplikací, která neustále běží v systému, ale budou ji tvořit dva skripty, které budou spouštěny systémovou aplikací `cron`. Jeden skript bude určen pro stahování předpovědí a druhý pro stahování reálně naměřených hodnot. Před spuštěním těchto skriptů bude potřeba definovat, pro které lokace se data budou stahovat. K tomuto účelu bylo navrhnuto vstupní rozhraní aplikace, které bude schopno načíst soubor ve formátu KML. Vzhledem k charakteru nabízených dat bude vhodný interval stahování tři hodiny s počátkem o půlnoci viz podkapitola 3.2. Veškerá stažená data bude potřeba převést ze serializačních formátů na záznamy, které se ihned po stažení uloží do databáze.

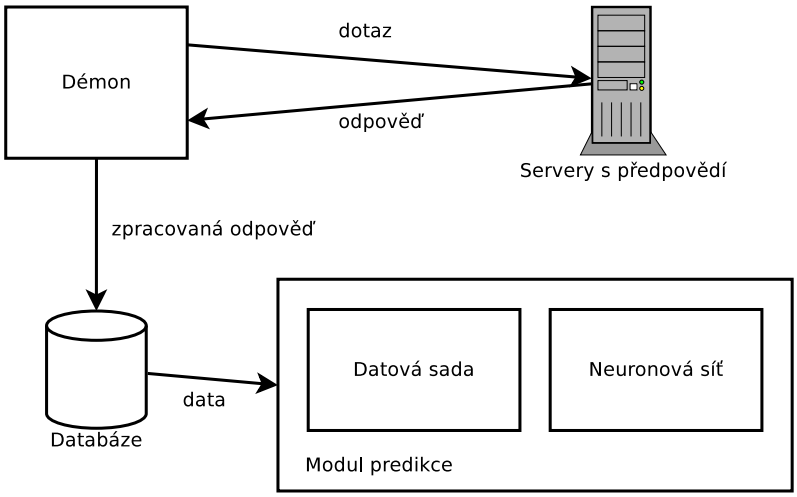
Právě databáze bude další částí navrženého systému. Bude potřeba, aby databáze obsahovala několik tabulek pro uložení adres serverů, stažených předpovědí, naměřených dat a lokací, pro které bude vytvořena předpověď. Podrobnější popis databáze je uveden v sekci 3.3.

Třetí a poslední částí bude modul, který bude vytvářet novou predikci. Tento proces se bude skládat ze dvou kroků. V první fázi bude potřeba vytvořit data pro neuronovou síť. Tyto data se budou skládat ze tří množin – trénovací, testovací a základ predikce. Nejprve budou data vybrána z databáze a převedena do vhodného rozsahu. Pro kombinaci jednotlivých zdrojů bude použita metoda popsaná v sekci 3.1. Poté budou data rozčleněna do množin podle zadaných intervalů. Na trénovací a testovací data bude použita metoda klouzajícího okna (viz podsekce 2.2.1). Po přípravě dat bude následovat druhý krok – vytvoření neuronové sítě. Jako knihovna s implementací neuronové sítě byla vybrána knihovna OpenCV Machine Learning, ze které bude použita dopředná vícevrstvá síť s učením zpětného šíření chyby a sigmoidní aktivační funkcí. Pro práci s touto knihovnou bylo navrženo objektově orientované rozhraní, které rozšiřuje její funkčnost a umožňuje její použití s předem připravenými daty.

Výstupem tohoto systému bude soubor dat obsahující test naučenosti sítě a predikované hodnoty.

3.1 Metoda přesnější předpovědi

Jak bylo zmíněno v podsekci 2.3.4, aplikace stahuje předpovědi ze čtyř různých zdrojů. Všechny tyto předpovědi je potřeba zkombinovat v jednu předpověď – jednu časovou řadu



Obrázek 3.1: Schéma aplikace.

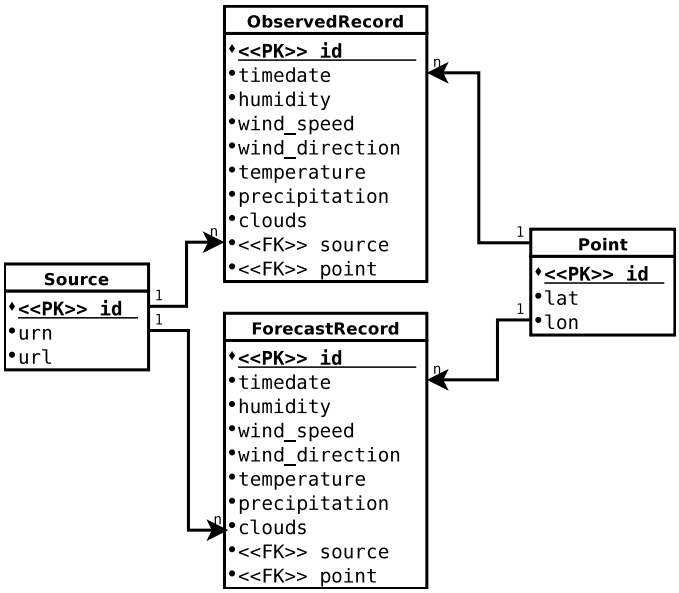
tak, aby výsledná předpověď byla co nejblíže realitě, tedy skutečně naměřeným hodnotám. Tato výsledná nejvhodnější předpověď bude sloužit jako zdrojová časová řada pro vytvoření trénovací množiny neuronové sítě. Máme čtyři předpovědi $x_1(t), x_2(t), x_3(t), x_4(t)$ a naměřená data $x_o(t)$, kde $t = 0, 1, .., n$. Výsledná časová řada vznikne tak, že vezmeme prvek z řad x_1, x_2, x_3, x_4 v čase t, pro každý zdroj vypočítáme odchylku od naměřené hodnoty z x_o v čase t a hodnotu zdroje, který má nejmenší odchylku umístíme do výsledné časové řady v čase t. Tento postup opakujeme pro všechna t. Takto vznikne nová předpověď, která střídavě vybírá hodnoty z použitých časových řad. Četnost použití jednotlivých zdrojů závisí na jejich přesnosti.

3.2 Omezení dostupnosti serveru

V této práci je využíváno volně dostupných webových služeb, které nabízejí serializovaná data s předpověďmi. Jak je uvedeno v podsekci 2.3.4, dostupnost serverů se pohybuje kolem 95 %, což je jeden z faktů, na které bude potřeba brát ohled. Pokud některý z dotazovaných serverů nebude dostupný, aplikace nebude moci v daný čas stáhnout potřebná data a časová posloupnost sbíraných předpovědí může být narušena. Kvůli tomuto problému bylo navrhnuto následující řešení. V každý daný čas, kdy se aplikace připojí na servery, se stáhne předpověď po třech hodinách na následujících 30 hodin, což je deset hodnot. Při dotazu na určité časové období se projdou náležité záznamy, ze kterých bude odebrána pouze první hodnota, tedy předpověď na následující 3 hodiny. Takto vznikne souvislá časová řada předpovědí určitého časového období, která bude obsahovat nejaktuálnější hodnoty. V případě, že některý časový záznam v databázi bude chybět kvůli nedostupnosti serveru, bude do časové řady zařazena druhá hodnota předchozího záznamu. Díky tomu, že aplikace bude ukládat deset hodnot, bude možné tolerovat nedostupnost serverů až na 30 hodin. Ovšem za předpokladu snížení přesnosti předpovědi.

3.3 Návrh databáze

Pro cílovou aplikaci byla navrhnuta poměrně jednoduchá databáze. Návrh databáze obsahuje čtyři tabulky. První tabulka `Point` slouží k uložení informace o souřadnicích, pro které jsou data stahována. Tabulka obsahuje kromě identifikátoru dva atributy – zeměpisnou šířku (lat) a délku (lon). Oba tyto atributy jsou reálná čísla. Druhá tabulka slouží k uchování zdrojů, odkud se data stahují. Jedná se o tabulku `Source`. Tabulka kromě názvu zdroje obsahuje také adresu zdroje. Následující dvě tabulky databáze jsou téměř stejné. Obě mají stejné položky. Cizí klíče k tabulkám `Point` a `Source`, a také atributy představující parametry z podsekce 2.3.3. Zatímco u tabulky `ObservedRecord` jsou tyto atributy reálná čísla, tak u tabulky `ForecastRecord` to jsou seznamy reálných čísel. Pro tento projekt byla zvolena databáze SQLite. Celé schéma databáze je uvedeno na obrázku 3.2.



Obrázek 3.2: Schéma databáze.

Kapitola 4

Realizace

V rámci této práce byla realizována aplikace podle návrhu uvedeného v kapitole 3. Výsledná aplikace je napsána v jazyku Python. V následujících podkapitolách jsou uvedeny použité technologie a popis implementace aplikace.

4.1 Použité technologie

V této podkapitole budou popsány aplikace, knihovny a formáty, které byly použity při implementaci aplikace.

4.1.1 Objektově relační mapování

Objektově relační mapování (ORM) je programově aplikační vrstva, která spojuje relační databázi a objektově orientovaný programovací jazyk [2]. Jde o programové rozhraní, které se snaží zobecnit přístup k odlišným relačním databázím. Princip této techniky vychází z návrhu, kde objekt programovacího jazyku reprezentuje tabulku v relační databázi. Jednotlivé atributy objektu pak sloupce v tabulce databáze. Dalo by se říct, že se jedná o abstraktní vrstvu nad databází, kdy je programátorovi skryt dotazovací jazyk databáze a je mu umožněno pracovat pouze programovým rozhraním objektu představující tabulku v databázi.

Hlavní výhodou tohoto přístupu je nezávislost na relační databázi. Samozřejmě je to omezeno výčtem databází, které ORM podporuje. Další výhodou je absence znalosti dotazovacích databázových jazyků například SQL. Jako nevýhodu je třeba zdůraznit, že složitější dotazy nad databází provedené pomocí rozhraní ORM nemusí být vždy tak efektivní jako čistý SQL dotaz. Navíc, při dotazování na více záznamů z databáze, musí být drženy v operační paměti objekty všech vybraných záznamů, což může být paměťově náročná operace. Ovšem tuto slabinu ORM řeší například jazyk Python zavedením objektových slotů. Aktuálně existuje mnoho knihoven pro téměř všechny moderní programovací jazyky.

4.1.2 Django Framework

Django[4] je webový aplikační framework, jehož architektura vychází z návrhu Model-View-Controller (MVC). Díky tomu, že struktura projektů založených na tomto frameworku je prakticky volná, je Django vhodné jak pro rozsáhlé projekty, tak i pro jednodušší aplikace. Mezi silné vlastnosti tohoto frameworku rozhodně patří jeho databázové API nebo jinak

[4] https://www.djangoproject.com/

také ORM (viz podsekce 4.1.1), automatické generování administrace z nadefinovaných databázových modelů, vícejazyčnost nebo také šablonovací systém. Běžná struktura projektu je rozdělena do několika adresářů a souborů. Dva základní soubory jsou `settings.py`, který slouží ke konfiguraci projektu, a `manage.py`, který spouští příkazy jako je inicializace databáze, spouštění externích skriptů. Dále se v projektu mohou nacházet adresáře, které reprezentují tzv. aplikace, což jsou menší moduly nesoucí samostatnou funkčnost. V těchto adresářích se obvykle nacházejí soubory s databázovými modely a pohledy. Django framework má velmi dobré databázové API, prostřednictvím kterého nabízí databáze SQLite, PostgreSQL, MySQL nebo Oracle.

4.1.3 Knihovna OpenCV – Machine Learning

OpenCV je multiplatformní knihovna pro práci s obrazem. Je to otevřená knihovna, vydaná pod licencí BSD. Její poslední verze 2.x je napsaná v jazyku C++, ale obsahuje rozšíření, které umožňuje její využití z jazyku Python. Nedílnou součástí rozhraní pro jazyk Python je knihovna NumPy (viz podsekce 4.1.4). Struktura celé knihovny je modulární. Základem je modul `core`, který obsahuje základní funkce a datové struktury. Dále se skládá v několika modulů, která jsou zaměřeny na jednotlivé oblasti počítačového vidění a zpracování obrazu:

imgproc zpracování obrázků, filtry, transformace

video zpracování videa, rozpoznávání objektů ve videu

objdetect detekce objektů jako obličeje, lidé, auta a další

Modul Machine Learning (ML) implementuje metody strojového učení jako jsou rozhodovací stromy, podpůrné vektory, bayesovské sítě, a nebo právě neuronové sítě. Tento submodul obsahuje pouze jeden typ neuronové sítě, a to dopřednou umělou neuronovou síť často nazývanou jako vícevrstvý perceptron. Síť je reprezentována třídou `ANN_MLP`, která nabízí několik základních metod pro práci s neuronovou sítí. Metody `train`, `predict`, `create` umožňují vytvořit a provést trénování sítě s všemi parametry, které ovlivňují efektivitu neuronových sítí.

4.1.4 Knihovna NumPy

Numerical Python je otevřená matematická knihovna zaměřená na práci s mutli-dimenzionálními maticemi a poli. Kromě toho obsahuje funkce, které provádí operace nad maticemi včetně lineární algebry, statistiky, Fourierovi transformace a další. Je implementována v jazyku C a Python. Základem je pole `ndarray`, což je n-dimenzionální pole. Toto pole se mírně liší od klasického pole, které známe z jazyku Python. Pole z NumPy má fixní velikost, kterou nelze měnit a je homogenní. Takže všechny prvky pole musí být stejného datového typu. Jádro knihovny je tvořeno moduly `core` a `lib`, které obsahují základní objekty a funkce. Některé další moduly knihovny NumPy:

linalg lineární algebra

fft Fourierova transformace

random generování náhodných čísel

testing unit testování

Knihovna NumPy se současně používá s Python API knihovny OpenCV verze 2, kde veškeré číselné vektorové vstupy různých objektů musí být datového typu právě `ndarray`.

4.1.5 Keyhole markup language

Jedná se o značkovací jazyk, který vychází ze standardu XML (viz podsekce 2.3.1). Vznikl současně se službami jako jsou Google Earth[5] a Google Maps[6]. Jazyk slouží k ukládání geografických informací jako jsou body, čáry, plochy, obrázky, které je možno zobrazit například v Google Earth.

4.1.6 Aplikace Cron

Tento program je neodmyslitelnou součástí všech unix-like[7] operačních systémů. Cron běží v systému po celou dobu jako démon a spouští příkazy v určitý čas, který je předdefinovaný v souboru `/etc/crontab`. Jednotlivé řádky v konfiguračním souboru reprezentují příkazy ke spuštění. Každý řádek by měl obsahovat 3 údaje. První údaj se skládá z pěti položek (minuta, hodina, den v měsíci, měsíc a den v týdnu), kde každá obsahuje časový údaj určující kdy se příkaz spustí. Místo každé položky může byt uvedena hvězdička, což znamená, že se příkaz vykonává vždy. Například hvězdička na místě minut znamená spouštění příkazu každou minutu. Následující údaj označuje jméno uživatele, pod kterým bude příkaz spuštěn. Poslední údaj reprezentuje samotný skript včetně celé cesty k němu a případných parametrů. Ukázka skriptu `manage.py`, který se pod uživatelem `root` spouští každé tři hodiny:

```
* 0,3,6,9,12,15,18,21 * * * root python manage.py runscript observed
```

4.1.7 Databáze SQLite

Jedná se o multiplatformní otevřenou relační SQL databázi. Je to poměrně jednoduchá aplikace, která vyžaduje jen minimální podporu externích knihoven nebo operačního systému. Je napsaná v jazyku C. Tato databáze není serverového typu, tedy v systému neběží ve zvláštním procesu, ale veškerá data se ukládají do speciálního souboru, který je uložen na disku. Výhodou toho přístupu je nulová konfigurace před spuštěním. Prakticky jakýkoliv program je ihned schopen přistupovat k této databázi. Na druhou stranu tím, že přístup k tomuto souboru není řízen žádným procesem, může dojít k problémům u souběžného přístupu z více aplikací. Velmi časté využití je u webových prohlížečů jako cache databáze.

4.2 Implementace

Tato podkapitola se zabývá konkretními implementačními detaily. Bude popsán vstup a výstup aplikace a její celá struktura.

4.2.1 Struktura aplikace

Celá aplikace je postavena na frameworku Django, a to hlavně kvůli kvalitnímu Django ORM. Struktura je rozdělena do několika celků.

Modul `conf` obsahuje soubor `settings.py`, ve kterém je veškeré nastavení celé aplikace. Tento soubor obsahuje nastavení souborů pro logování a připojení k databázi. Další modul `tools` obsahuje soubor `erorrs.py`, jenž obsahuje definice výjimek, které mohou nastat při

[5] http://www.google.com/earth/
[6] https://maps.google.com/
[7] Toto označení se používá pro operační systémy, které vycházejí z architektury systému UNIX.

běhu aplikace. Další soubor tohoto modulu je `utils.py`, který obsahuje pomocné funkce jako jsou převody jednotek, vykreslování grafů nebo dekorátor pro měření rychlosti funkce učení. Poslední soubor modulu je `deserializers.py`, ve kterém jsou funkce zpracovávající odpovědi ze serverů ve formátech JSON nebo XML. Tyto funkce převedou odpovědi na objekty, které jsou definovány v modulu `database` obsahující model databáze dle podkapitoly 3.3. Dalším modulem této aplikace je `mlp`, který obsahuje soubor s třídami pro neuronovou síť a datové množiny. Další významný modul je tvořen adresářem `scripts`, ve kterém jsou umístěny skripty ovládající celou aplikaci. Tyto skripty jsou spouštěny pomocí hlavního Django souboru `manage.py`, který rovněž zajišťuje načtení konfiguračního souboru a ostatních modulů v projektu. Výčet některých spustitelných skriptů:

import.py načtení a zpracování vstupního souboru ve formátu KML s body, pro které se budou stahovat předpovědi

check_forecast.py kontrolní skript, ověřuje stažené předpovědi

querier.py skript, který se dotazuje na předpovědi jednotlivých serverů

observed.py skript, který stahuje naměřená data

run.py ukázka použití rozhraní neuronové sítě

4.2.2 Inicializace aplikace

Před spuštěním aplikace je třeba definovat, pro která místa se budou stahovat předpovědi a následně se bude vytvářet nová předpověď. Místo je reprezentováno zeměpisnou šířkou a délkou. Inicializovat aplikaci lze skriptem `import.py`, který jako první a jediný parametr přebírá název vstupního souboru. Tento soubor musí byt ve formátu KML, který lze vygenerovat například ze služby Google Maps. V této geografické službě lze pomocí jednoduchého webového rozhraní vytvořit mapu s vlastními body. Tato mapa se snadno vyexportuje do souboru právě ve formátu KML. Po spuštění inicializačního skriptu jsou geografická místa načtena ze souboru a uložena do databáze do příslušné tabulky. Po inicializaci aplikace je možné spustit stahování předpovědí.

4.2.3 Serverová část

Serverovou aplikaci tvoří několik částí. Databáze, jejíž schéma je definováno v souboru `database/models.py` a také skripty umístěné v adresáři `scripts`. Jedná se o skripty `observed.py` a `querier.py`. Tyto skripty mají velmi podobnou funkci. Oba se dotazují serverů na určitou předpověď a čekají na odpovědi. Oba rovněž volají pomocné funkce z `tools/deserializers.py`, které zpracovávají odpovědi. Po zpracování odpovědi je přidán záznam do databázové tabulky `ForecastRecord`, případně `ObservedRecord`. V obou případech jak úspěchu, tak i neúspěchu získání dat ze serveru, je výsledek dotazu logován do souboru. Skript `querier.py` se dotazuje všech zdrojů, které jsou uvedeny v databázi pod tabulkou `Source`, zatímco skript `observed.py` se dotazuje pouze na adresu serveru, který je definovaný v `conf/settings.py` nabízející naměřené hodnoty. Skripty jsou spouštěny pomocí nástroje `cron` v pravidelných intervalech po třech hodinách počínaje půlnocí.

4.2.4 Modul predikce

Zbytek aplikace tvoří dvě třídy `BackpropNetwork` a `DataSet`, které se nacházejí v souboru `mlp/models.py`. Nejprve je nutné vytvořit instanci třídy `DataSet`. Tato třída slouží k vytvoření trénovací a testovací množiny ze stažených dat. Konstruktor této třídy přebírá čtyři parametry – identifikátor místa, sledovaný parametr počasí a velikost časového okna. Po vytvoření instance se předpřipraví data z databáze a normalizují se do vhodného rozsahu $\langle -1, 1 \rangle$. Dále je potřeba zvolit, ze kterých částí databázových dat budou vytvořeny jednotlivé množiny pro neuronovou síť. K tomu slouží *public* metody této třídy. Metoda `make_train_set` vytváří pomocí metody klouzajícího okna (viz podsekce 2.2.1) trénovací množinu. Její dva parametry určují rozsah. Podobnou metodou je `make_test_set`, která ale vytváří testovací množinu. Všechny vzniklé množiny jsou dostupné v instančních proměnných:

- `train_input`, `train_output`
- `test_input`, `test_output`
- `predict_input`, `predict_output`

Po přípravě dat je možné vytvořit instanci třídy `BackpropNetwork`. Tato třída slouží k práci s neuronovou sítí, při vytváření instance se zároveň vytvoří i instance třídy `ANN_MLP` z knihovny OpenCV, takže třída `BackpropNetwork` tvoří její rozšířené rozhraní. Konstruktor této třídy přebírá celkem pět parametrů. První tři jsou celá čísla, jedná se o nastavení schéma neuronové sítě, tedy určují počet neuronů ve vstupní, skryté a výstupní vrstvě. Další parametr je instancí třídy `DataSet`, která obsahuje všechny potřebné množiny dat k práci s neuronovou sítí. Posledním parametrem je název, který je používán pro pojmenování výstupních souborů (viz podsekce 4.2.5). I tato třída má několik *public* metod. Před učením neuronové sítě je potřeba nastavit parametry učení. K tomu slouží funkce `set_train_params`. Tato funkce má celkem čtyři parametry, které mohou ovlivnit učení sítě. Je to parametr rychlosti učení, moment učení, maximální počet iterací učení a maximální chyba mezi jednotlivými iteracemi. Další metoda `train` provádí učení sítě s parametry z předchozího nastavení. Tato metoda vytiskne na standardní výstup počet provedených iterací během učení sítě. Po naučení sítě je možné spustit test naučenosti metodou `test`. Po otestování sítě je na standartní výstup vypsána kvalita naučenosti, která je spočítána jako odmocnina střední kvadratické chyby z podsekce 2.2.2. Poslední funkcí třídy je samotná predikce. Metoda `predict_and_test` má pouze jeden parametr, který určuje délku předpovědi. U této metody dochází k denormalizaci dat tak, aby výsledná predikce byla ve stejném intervalu jako původní hodnoty.

4.2.5 Výstup aplikace

Současně s použitím metod `test` a `predict_and_test` třídy `BackpropNetwork` se pro každou vytváří výstupní soubory. Jako výsledek metody vznikají dva soubory. První soubor je textový a obsahuje číselné výsledky dané metody. Druhým souborem je obrázek obsahující graf, který je vytvořen z předchozího souboru dat. Graf je tvořen funkcí `plot` z modulu `tools.utils`. Tato funkce nastavuje parametry grafu a pomocí nástroje `gnuplot`[8] vykresluje graf do souboru.

[8] http://www.gnuplot.info/

Kapitola 5

Dosažené výsledky

První část závěrečné kapitoly se věnuje testování parametrů neuronové sítě. Budou testovány parametry, které ovlivňují učení sítě. Druhá část bude vycházet z těchto experimentů a bude vytvářet predikci o různých délkách.

5.1 Experimenty

Předmětem této podkapitoly je nalezení nejvhodnějšího nastavení jednotlivých parametrů z podsekce 2.1.3, které ovlivňují učení neuronové sítě. Snahou je dosáhnout takového nastavení, aby neuronová síť byla schopna co nejpřesněji předpovídat. Při experimentování byla jako kritérium kvality použita odmocnina ze střední kvadratické chyby (RMSE). Pro testování byla použita data za období únor až duben 2014. Pro všechny testy byla použita stejná data. Jako testovaná lokace byla náhodně vybrána jedna z pěti sledovaných. Testování provádí funkce `test`, která předloží testovací množinu naučené neuronové síti a výstupy porovná s očekávanými hodnotami. Tato podkapitola demonstruje pouze experimenty jedním parametrem počasí z podsekce 2.3.3, kterým je teplota. Zdrojové soubory aplikace obsahují vhodnou konfiguraci i pro ostatní parametry počasí. Celkem bylo zkoumáno pět parametrů:

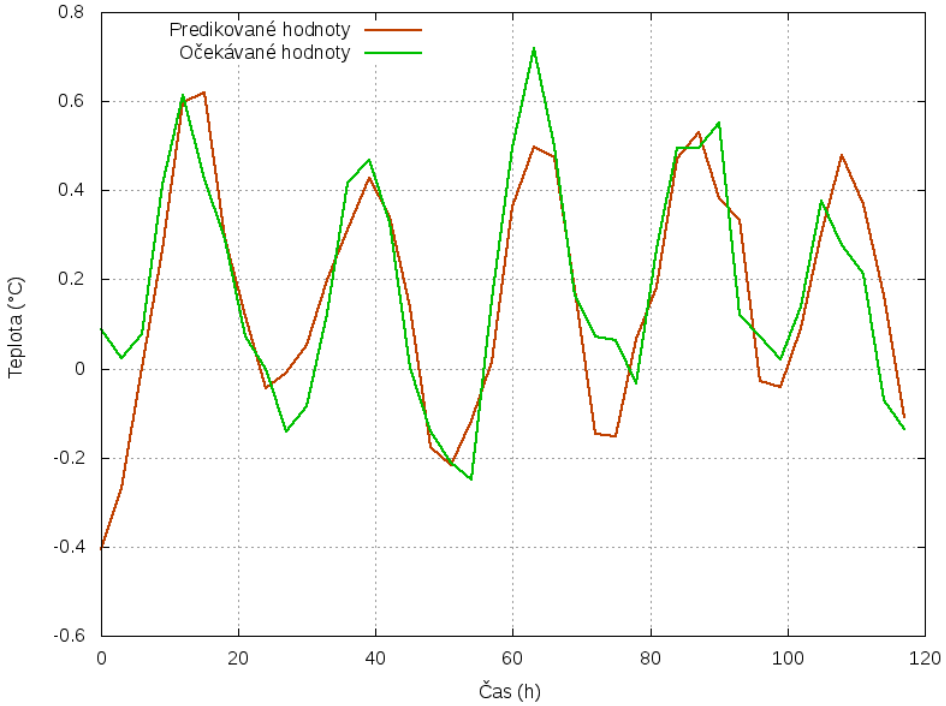
- parametr rychlosti učení
- moment učení
- počet skrytých neuronů
- počet skrytých vrstev
- velikost časového okna

5.1.1 Počet neuronů ve skryté vrstvě

První experiment se snaží zjistit vliv počtu neuronů ve skryté vrstvě. Neuronová síť obsahuje pouze jednu skrytou vrstvu. Parametr rychlosti a momentu učení má hodnotu 0,1. Jako velikost časového okna a zároveň počet vstupních neuronů byla zvolena hodnota 8, což odpovídá časovému úseku 24 hodin. V tabulce 5.1 jsou uvedeny chyby RMSE v závislosti na počtu neuronů. Rovněž je v tabulce uveden počet iterací, který proběhl u jednotlivých testů. Nejlépe dopadl test sítě, která obsahuje 55 neuronů v jedné skryté vrstvě. V grafu 5.1 je znázorněn průběh očekávaných a predikovaných hodnot.

Počet neuronů	5	12	24	42	55	90	150
RMSE	16,23	16,04	16,24	15,24	14,91	16,80	15,73
Počet iterací	3	3	2	2	2	2	3

Tabulka 5.1: Chyba RMSE podle počtu neuronů.



Obrázek 5.1: Průběh predikované a očekávané teploty při síti s 55 neurony.

5.1.2 Parametr rychlosti učení

Tento parametr ovlivňuje rychlost učení. Cílem tohoto testu je najít nejlepší hodnotu z intervalu (0,1). Test byl proveden na jednovrstvé neuronové síti s 8 vstupy, 55 skrytými neurony a momentem učení nastaveným na hodnotu 0,1. Podle tabulky 5.2 se jako nejvhodnější

Parametr rychlosti	0,1	0,3	0,5	0,7	0,9
RMSE	14,91	15,92	15,04	15,07	15,47

Tabulka 5.2: Chyba RMSE podle parametru rychlosti učení.

nastavení tohoto parametru jeví hodnota kolem 0,1, jejíž chyba RMSE byla nejnižší.

5.1.3 Moment učení

Moment učení vyjadřuje rozdíl vah mezi jednotlivými iteracemi. Testovány byly hodnoty v rozsahu (0,1) [4]. Dle tabulky 5.3 je nejvhodnější hodnota tohoto parametru kolem hodnoty 0,5.

Moment	0,1	0,3	0,5	0,7	0,9
RMSE	14,91	14,77	14,70,	14,97	17,27

Tabulka 5.3: Chyba RMSE dle hodnoty momentu učení.

5.1.4 Počet skrytých vrstev

Měření bylo provedeno na neuronové síti s 8 vstupy, 55 skrytými neurony v každé vrstvě a parametrem rychlosti a momentu učení o hodnotě 0,1.

Počet skrytých vrstev	1	2	3	4	5
RMSE	14,91	15,27	16,33	27,04	29,54
Doba učení (s)	0,032534	0,172177	0,298029	0,283664	0,368697

Tabulka 5.4: Chyba RMSE podle počtu skrytých vrstev.

Tento experiment prokázal, že pro tento typ úlohy je nejvhodnější pouze jedna skrytá vrstva. Navíc z tabulky 5.4 můžeme vidět, že s přibývajícími vrstvami se prodlužuje čas učení.

5.1.5 Velikost časového okna

V rámci tohoto měření byla snaha najít vhodnou velikost časového okna tedy počet vstupních neuronů sítě. Opět byla použita stejná konfigurace sítě tedy jedna skrytá vrstva s 55 neurony, moment a parametr učení o hodnotě 0,1. Tento test potvrdil trend, který lze vyvo-

Velikost časového okna	2	4	8	16
RMSE	17,45	19,68	14,91	14,52

Tabulka 5.5: Chyba RMSE podle velikosti časového okna.

dit z prostého pozorování průběhu teploty. V nočních hodinách je teplota nižší a dosahuje minima, zatímco během noci dosahuje opačných hodnot. Tento jev se podobně opakuje každých 24 hodin, avšak pro různé roční období v různých rozmezích. Proto původní odhad 8 vstupních neuronů (případně 16) odpovídající časovému úseku 24 hodin se jeví jako nejvhodnější.

5.2 Srovnání predikce

V podkapitole 5.1 bylo provedeno několik experimetů, které měly naleznout nejvhodnější konfiguraci neuronové sítě pro predikci teploty. Po otestování několika základních parametrů, které ovlivňují učení sítě, byla nalezena následující konfigurace neuronové sítě:

parametr rychlostí učení 0,1

moment učení 0,5

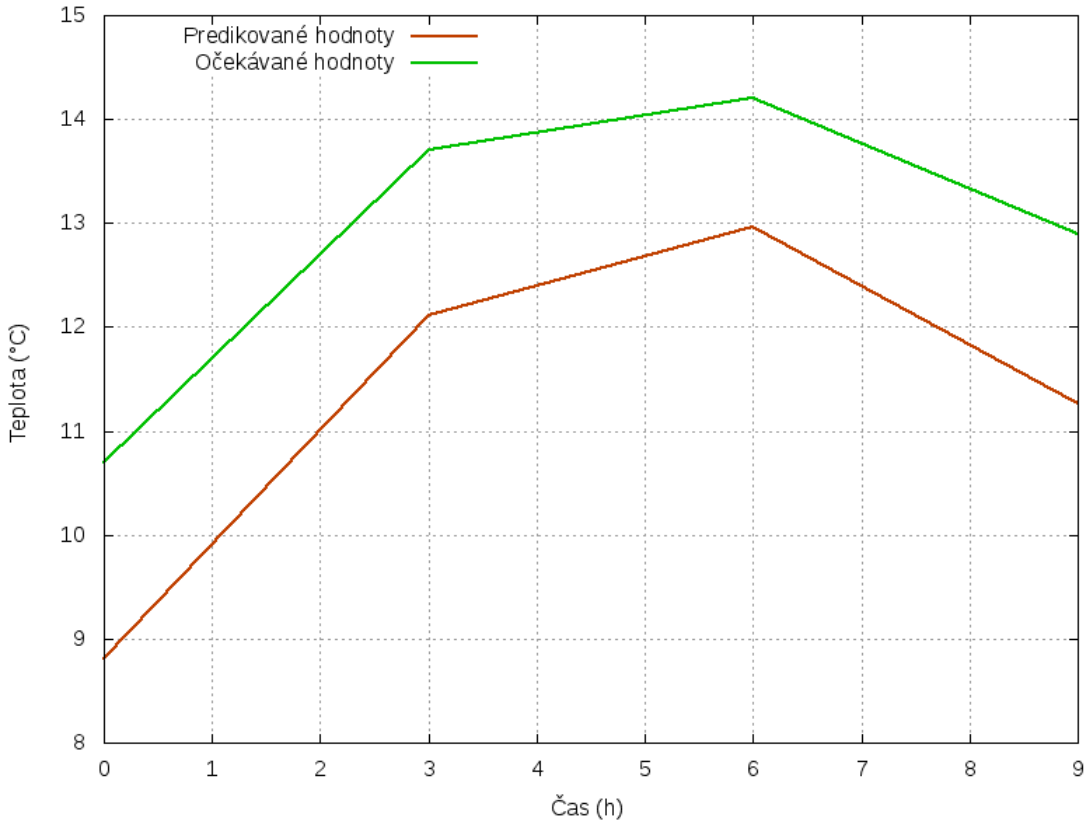
počet skrytých neuronů 55

počet skrytých vrstev 1

velikost časového okna 8

Pro hodnocení byla použita predikce vytvořená pomocí neuronové sítě této konfigurace. Vzhledem k tomu, že snaha odhadnout kratší dobu bývá obvykle přesnější, byla pro vyhodnocení vytvořena predikce pouze o délce 3 hodnot, což odpovídá období 9 hodin. Tato predikce byla porovnána s reálně naměřenými daty. Pro predikci byla použita funkce `predict_and_test`, která vytváří predikci dle požadované délky a vrací denormalizované hodnoty.

Na grafu 5.2 je vidět průběh hodnot teploty predikované a reálně naměřené během 9 hodin. Neuronová síť byla schopna vytvořit predikci, která poměrně přesně kopíruje průběh teploty, avšak predikované hodnoty jsou menší asi o 2 stupně.

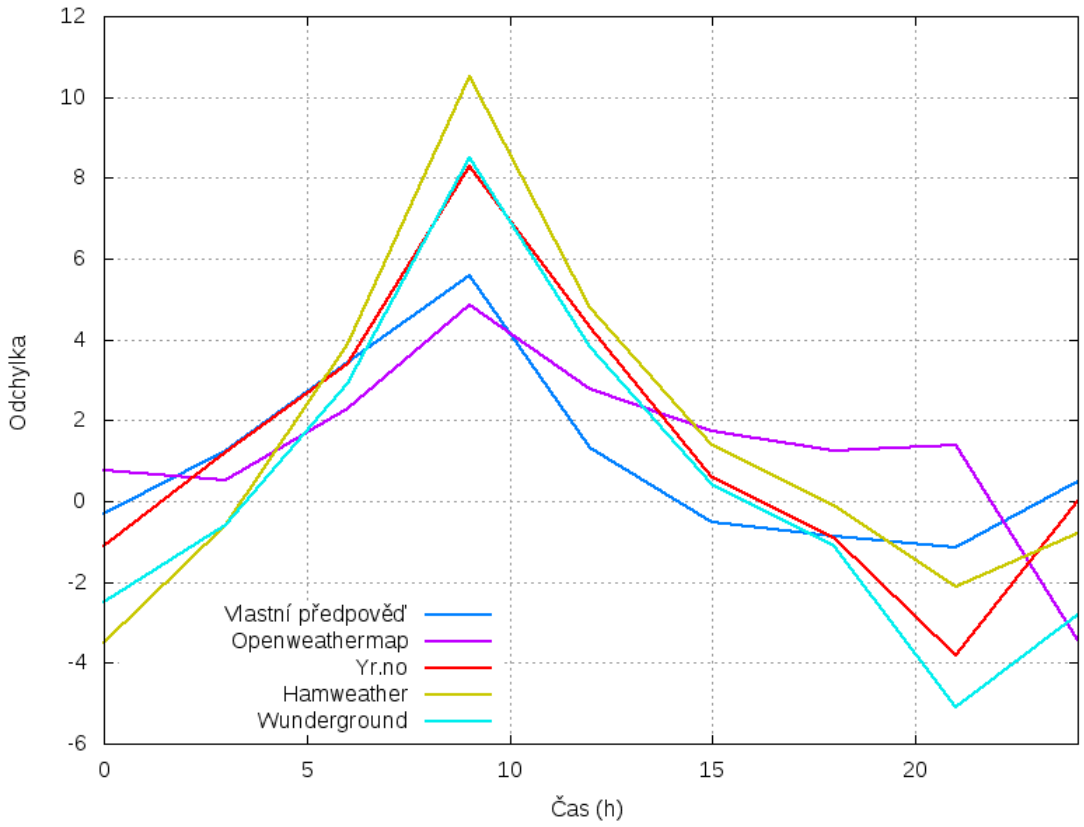


Obrázek 5.2: Srovnání vlastní predikce s realitou během 9 hodin.

Dále byla nově vytvořená předpověď porovnávána se zdroji, ze kterých byla vytvořena. Pro toto zhodnocení byla použita opět metoda RMSE. Byly provedeny čtyři predikce o délkách 12, 24, 84 a 168 hodin, které byly srovnány s původními předpověďmi a současně i s reálně naměřenými daty. Výsledky měření jsou uvedeny v tabulce 5.6. Podle naměřených hodnot lze říci, že nová předpověď je srovnatelná s původními zdroji. Nejlepších výsledků dosahuje při kratší predikci. Na obrázku 5.3 jsou zobrazeny odchylky jednotlivých předpovědí na 24 hodin od reálně naměřených dat.

Zdroj/Délka předpovědi	12 hodiny	24 hodin	84 hodin	168 hodin
OpenWeatherMap	16,28	18,54	29,42	26,30
Yr.no	16,89	20,73	33,43	29,03
Hamweather	32,18	31,51	44,38	40,09
Wunderground	22,03	21,20	33,93	30,61
Vlastní předpověď	16,06	25,38	38,60	30,75

Tabulka 5.6: Porovnání všech předpovědí různých délek s reálně naměřenými daty.



Obrázek 5.3: Srovnání odchylek předpovědí od reality během 24 hodin.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit novou a přesnější předpověď počasí, která vychází z volně dostupných předpovědí. Za tímto účelem vznikla aplikace, která v pravidelných intervalech stahuje předpovědi počasí z různých zdrojů. Z těchto předpovědí vyhodnocuje nejpřesnější zdroj, ze kterého vytváří trénovací množinu pro neuronovou síť. Bylo provedeno testování parametrů, které ovlivňují učení sítě. Byly testovány parametry jako moment učení, rychlostí učení, počet skrytých neuronů, počet skrytých vrstev a velikost časového okna. Po nalezení nejvhodnější konfigurace bylo vytvořeno několik předpovědí různých délek, které byly srovnány s existujícími předpověďmi. Podařilo se vytvořit aplikaci, kterou se svou přesností predikce můžeme částečně srovnávat s existujícími službami.

Tato práce nahlíží na počasí jako na soubor několika parametrů (teplota, vlhkost, ...), které se snaží odděleně predikovat. Avšak dnešní systémy, které předpovídají počasí, zohledňují všechny parametry a vytváří předpovědi, kde jednotlivé parametry na sobě závisí. Možným směrem, kterým by se rozšíření této práce mohlo ubírat, je nalezení modelu, který by byl schopen kombinovat všechny parametry. Nově vzniklá aplikace by se měla při predikci dívat nejen na historická data, ale i na ostatní parametry a dle nich modifikovat novou předpověď. Dalším prvkem, nad kterým je třeba se alespoň zamyslet, je lokace. Aktuální aplikace umožňuje stahovat data pro více míst najednou. Pokud tato místa budou ve stejné oblasti, jsou možná rozšíření, která budou zohledňovat například postup oblačnosti.

Závěrem je třeba říct, že tato práce najde využití v nově vznikajícím projektu na Fakultě informačních technologií VUT v Brně. Jedná se o projekt inteligentní domácnosti. Zjednodušeně jde o inteligentní systém, který je vybaven různými senzory, pomocí kterých je možné vzdáleně pomocí telefonu (tabletu) sledovat aktuální stav budovy nebo nastavovat příslušné zařízení nainstalované v domě jako například otevírání oken nebo nastavování topení dle počasí. Vzhledem k tomu, že tento systém je teprve ve fázi vývoje, byla tato aplikace vyvíjena samostatně. Do budoucna se počítá s jejím rozšířením tak, aby ji bylo možné zapojit do zmíněného projektu.

Literatura

[1] JSON.ORG Introducing JSON. [online], [cit. 2014-04-16].
URL http://www.json.org

[2] Mapping Objects to Relational Databases: O/R Mapping In Detail. [online], [cit. 2014-04-16].
URL http://www.agiledata.org/essays/mappingObjects.html

[3] Neural Network FAQ. [online], [cit. 2014-04-16].
URL ftp://ftp.sas.com/pub/neural/FAQ.html

[4] Neural Networks - OpenCV 2.4.9.0 documentation. [online], [cit. 2014-04-16].
URL http://docs.opencv.org/modules/ml/doc/neural_networks.html

[5] Bouška, J.: *Neuronové sítě pro predikci časových řad.* Diplomová práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2008.

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

[7] Edwards, T.; Tansley, D. S. W.; Frank, R. J.; aj.: Traffic Trends Analysis using Neural Networks. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommuncations*, 1997, s. 157–164.

[8] Kishan Mehrotra, S. R., Chilukuri K. Mohan: *Elements of Artificial Neural Networks.* The MIT Press, 1997, ISBN 0-262-13328-8.

[9] Sviták, J.: *Neuronové sítě a predikce časových řad.* Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2010.

[10] Tomáš Hruška, J. T., Jan Kroulík: *Internetové aplikace (WAP) III.* FIT VUT v Brně, 2006-2012.

[11] V. Mařík, J. L., O. Štěpánková: *Umělá inteligence (4).* ACADEMIA, 2003, ISBN 80-200-1044-0.

[12] Zbořil, F. V.: Soft Computing. [online], [cit. 2014-04-16].
URL http://www.fit.vutbr.cz/study/courses/SFC

Příloha A

Obsah CD

A.1 Seznam adresářů a souborů

wfbods/api_tests testy webových služeb

wfbods/docs programová dokumentace aplikace

wfbods/docs/report zdrojové soubory technické zprávy

wfbods/wthr zdrojové soubory

wfbods/wthr/conf/settings.py nastavení projektu

wfbods/wthr/data inicializační soubory

wfbods/wthr/database/models.py definice schéma databáze

wfbods/wthr/mlp/model.py definice tříd neuronové sítě

wfbods/wthr/out výstupy aplikace

wfbods/wthr/scripts spustitelné skripty

wfbods/wthr/scripts/temperature_predict.py ukázka použití aplikace

wfbods/wthr/tools/errors.py seznam programových vyjímek

wfbods/wthr/tools/utils.py pomocné funkce

wfbods/wthr/tools/deserializers.py zpracování dat

wfbods/wthr/db.sqlite3 databázový soubor

wfbods/wthr/manage.py řídící soubor projektu

wfbods/wthr/scripts.log logovací soubor

Příloha B

Manuál

Aplikace byla vytvořena pro operační systém Linux Ubuntu/Debian.

B.1 Instalace

Pro instalaci projektu je potřeba, kromě samotného interpretu jazyku Python verze 2.7.5, také dvě aplikace. Správne balíčků `pip` a tvůrce virtuálního prostředí `virtualenv`. Instalace na systém Debian/Ubuntu:

```
$ sudo apt-get install python-pip python-virtualenv
```

Dále je postup instalace následovný:

```
# vytvoření virtualního prostředí
$ virtualenv bpenv
$ cd bpenv
$ source bin/activate

# instalace OpenCV
$ sudo apt-get install python-opencv

# instalace knihoven pro XML
$ sudo apt-get install libxml2-dev libxslt-dev
$ sudo apt-get install -y libxml2-dev libxslt1-dev zlib1g-dev

# instalace nástroje Gnuplot
$ sudo apt-get install gnuplot-nox

# nakopírování knihovny OpenCV do virtuálního prostředí
$ cp /usr/lib/pymodules/python2.7/cv*  lib/python2.7/site-packages/

# nakopírování zdrojových souborů z CD
$ unzip /media/cdrom/src.zip

# instalace zbývajících knihoven
$ cd wfbods
$ pip install -r requirements.txt
```

B.2 Použití aplikace

Pro použití aplikace je třeba vytvořit jednoduchý skript například `test.py` ve složce `scripts`. Tento soubor bude v hlavičce obsahovat načtení tříd `DataSet`, `BackpropNetwork` a funkce `plot`.

```
from mlp.model import DataSet, BackpropNetwork
from tools.utils import plot
```

Dále bude soubor obsahovat jedinou funkci `run`, ve které budou umístěny všechny příkazy. Nejdříve vytvoříme název programu a délku požadované predikce.

```
title = "test1"
predict_len = 4
```

Poté je třeba vytvořit datové množiny pro neuronovou síť. Nejprve vytvoříme instanci třídy `DataSet` s identifikátorem lokace 1, parametrem teploty a velikost časového okna o hodnotě 8.

```
ds = DataSet(1, "temperature", 8)
```

Následuje vymezení rozsahů pro jednotlivé množiny.

```
ds.make_train_set(0, 400)
ds.make_test_set(400,450)
ds.make_predict_set(441, 449, predict_len)
```

Nyní je potřeba vytvořit neuronovou síť. Vytvoříme instanci třídy `BackpropNetwork` s 8 vstupními, 55 skrytými a 1 výstupním neuronem, objektem datové sady `ds` a názvem programu `title`.

```
nn = BackpropNetwork(8, [55], 1, ds, title)
```

Poté je možné nastavit parametry trénování a sputit trénování.

```
nn.set_train_params(0.1, 0.5, 1000, 0.01)
nn.train()
```

Po natrénování sítě lze provést testování, které bude zakresleno do grafu pomocí funkce `plot`.

```
nn.test()
plot(title+"-test")
```

A posledním možným krokem je samotná predikce, kterou lze rovněž zakreslit do grafu.

```
nn.predict_and_test(predict_len)
plot(title+"-forecast")
```

Výsledný skript se spouští příkazem:

```
python manage.py runscript test
```

Příloha C

Ukázky odpovědí ze serverů

C.1 Hamweather


```
{
    "response": [
        {
            "profile": {
                "tz": "Europe/Prague"
            },
            "loc": {
                "lat": 49.088,
                "long": 17.876
            },
            "interval": "3hr",
            "periods": [
                {
                    "precipIN": 0,
                    "windSpeedMPH": 5,
                    "dateTimeISO": "2014-02-24T01:00:00+01:00",
                    "windDirDEG": 54,
                    "timestamp": 1393200000,
                    "tempC": 0,
                    "sky": 0,
                    "humidity": 84
                }
            ]
        }
    ]
}
```


C.2 Yr.no


```
<weatherdata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://api.met.no
/weatherapi/locationforecast/1.8/schema"
```

```
created="2013-09-22T20:01:33Z">
   <meta>
      <model name="EC.GEO.0.25" termin="2013-09-22T12:00:00Z"
      runended="2013-09-22T19:07:33Z" nextrun="2013-09-23T08:00:00Z"
      from="2013-09-22T21:00:00Z" to="2013-10-02T12:00:00Z" />
   </meta>
   <product class="pointData">
      <time datatype="forecast" from="2013-09-22T21:00:00Z"
       to="2013-09-22T21:00:00Z">
         <location altitude="350" latitude="49.0883" longitude="17.8755">
            <temperature id="TTT" unit="celcius" value="10.4"/>
            <windDirection id="dd" deg="289.3" name="W"/>
            <windSpeed id="ff" mps="2.0" beaufort="1" name="Flau vind"/>
            <humidity value="78.3" unit="percent"/>
            <cloudiness id="NN" percent="97.7"/>
         </location>
      </time>
    </product>
</weatherdata>
```

C.3 OpenWeatherMap

```
{
    "cod": "200",
    "message": 0.0015,
    "city": {
        "id": 3065843,
        "coord": {
            "lon": 17.873489,
            "lat": 49.08799
        },
    },
    "list": [
        {
            "dt": 1397800800,
            "main": {
                "temp": 283.71,
                "humidity": 82,
                "temp_kf": 5.46
            },
            "weather": [
                {
                    "main": "Clear",
                }
            ],
            "clouds": {
                "all": 0
            },
```

```
            "wind": {
                "speed": 1.3,
                "deg": 38.5042
            },
            "dt_txt": "2014-04-18 06:00:00"
        }
    ]
}
```


C.4 Wunderground


```
{
    "hourly_forecast": [
        {
            "mslp": {
                "metric": "1026",
                "english": "30.31"
            },
            "temp": {
                "metric": "2",
                "english": "36"
            },
            "wdir": {
                "degrees": "48",
                "dir": "NE"
            },
            "qpf": {
                "metric": "",
                "english": ""
            },
            "humidity": "72",
            "sky": "1",
            "FCTTIME": {
                "epoch": "1393192800",
                "pretty": "11:00 PM CET on February 23, 2014"
            }
        }
    ]
}
```