Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Strojové učení

Využití umělých neuronových sítí pro předpověď na základě hydrologických dat

Bakalářská práce

Autor: Martin Konvička
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Jan Vaněk, Ph.D.

Odborný konzultant: Ing. Karel Petránek, Ph.D.
KIKM FIM UHK

Hradec Králové Duben 2018

Prohlášení:

Prohlašuji, že jsem bakalářskou/diplomovou práci zpracoval/zpracovala samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové dne 26.4.2018

Poděkování:

Děkuji vedoucímu bakalářské práce Mgr. Janu Vaňkovi, Ph.D. za metodické vedení práce a státnímu podniku Povodí Labe za poskytnutá data

Anotace

Tato bakalářská práce se zabývá využitím umělých neuronových sítí pro predikci průtoku na základě hydrologických dat. Hlavním účelem výzkumné činnosti bylo ověřit využitelnost získaných dat pro strojové učení. Vedlejším cílem bylo také stanovení optimální architektury sítě a hyper-parametrů učení. Pro tyto účely bylo odzkoušeno použití standardní dopředné architektury neuronové sítě s různým počtem skrytých vrstev a regularizační techniky dropoutu. V průběhu výzkumu bylo nutno překonat řadu nepříjemností, např. nedostatek dat zachycující zvýšenou povodňovou aktivitu, nebo obtížnost porovnávat úspěšnost jednotlivých modelů. Bylo použito mnoho způsobů řešení problémů výše uvedených, nepodařilo se však vytvořit model, jehož přesnost by vyhovovala požadavkům státního podniku Povodí Labe.

Annotation

Title: Machine Learning

This bachelor thesis explores the capabilities of feed-forward artificial neural networks for the purposes of predicting from hydrological data. The main objective of the research behind this was to evaluate the data received in terms of its suitableness for machine learning. The secondary goal had to do with comparing various architectures of neural networks and hyper-parameters of learning in order to establish the best combination of both. During the experimental phase we were encountered with several practical problems, such as the lack of data describing increased flooding activity and the difficulty of comparing the accuracy of models due to the use of inadequate metrics. We tried to tackle the problem of not having enough relevant data by using many unconventional methods and techniques, all of which are described in the thesis. Although we are able to conclude that more complicated models generally perform better, our efforts to solve the problems regarding the unevenness of data have not yielded satisfactory results.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Strojové učení je ve 21. století velice rozšířenou a žádanou disciplínou. Příčinu lze spatřit ve vývoji pokročilých učících algoritmů, dostupnosti obrovského množství dat, z nichž je pomocí technik strojového učení možné extrahovat informace a vytvářet predikční modely, a pokroku hardwarových technologií, umožňující paralelizaci výpočtů, které tím výrazně urychlily. V této práci jsou prozkoumány možnosti využití umělých neuronových sítí jakožto modelu predikce na základě hydrologických dat získaných od státního podniku Povodí Labe. Tato data obsahují naměřené údaje ve více méně pravidelných intervalech, která pokrývají období několika let. Hlavním důvodem výzkumné činnosti bylo ověřit využitelnost získaných hydrologických dat pro tvorbu modelů schopných kvalitní predikce povodňové aktivity na základě senzorických údajů naměřených v nejbližší historii.

2 Cíl práce

Smyslem práce bylo ověřit možnosti předpovídání hydrologických údajů s využitím neuronových sítí. Základním předpokladem úspěšné předpovědi je existence funkční závislosti mezi údaji o minulém průběhu (srážky, teploty, průtoky) a budoucím průtokem.

3 Strojové učení

Strojové učení je odnož umělé inteligence zabývající se vývojem obecných algoritmů schopných řešit takové typy úloh, které by byly velmi obtížně explicitně naprogramovatelné. Jednoduchým příkladem takové úlohy může být rozpoznání ručně psaných číslic z rastrového obrazu. Historicky byly úlohy z oblasti umělé inteligence řešeny definováním tzv. pokud-potom pravidel.

Strojové učení je interdisciplinární oblast nacházející aplikace v medicíně, vojenství, počítačovém vidění, zpracováním přirozené řeči, bankovnictví, IT a mnoha dalších. Primárně lze problémy strojového učení dělit na:

1. Učení s učitelem (supervised learning)
2. Učení bez učitele (unsupervised learning)
3. Posilující učení (reinforcement learning)

Při učení s učitelem je na základě poskytnutých vstupů a výstupů snahou naučit model transformovat patřičný vstup na patřičný výstup (perceptrons, deep neural networks). Učení bez učitele je založeno především na hledání zákonitostí a podobností v datech, za účelem jejich lepšího pochopení (k-means, principal component analysis). Principem posilujícího učení je nalézt strategii rozhodovaní, která povede k největšímu užitku (Q-learning).

Problémy, které lze řešit pomocí učení s učitelem, mohou být dvojího typu:

1. Regrese
2. Klasifikace

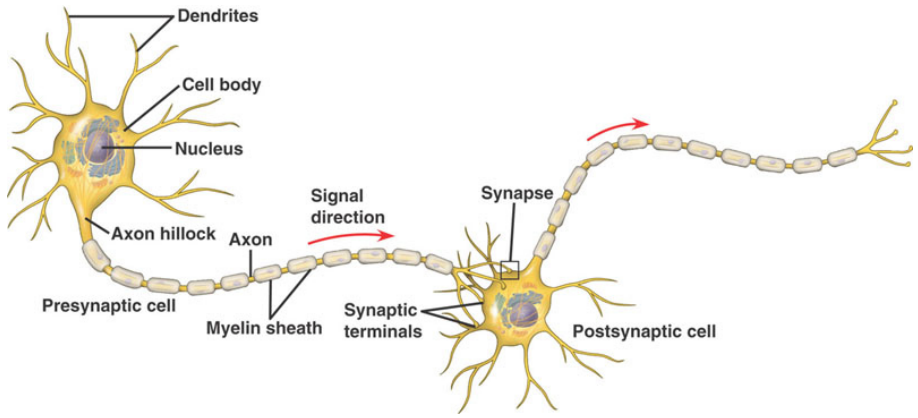
U regresních problémů je výstupem modelu nějaká reálná hodnota. Příkladem může být předpověď ceny pozemku na základě počtu bytů a obsahu pozemku. U klasifikace naopak chceme rozčlenit vstupy do jednotlivých kategorií. Příkladem může být klasifikace pacientů na zdravé a nemocné na základě věku pacienta, krevního tlaku atd.

3.1 Umělé neuronové sítě

Umělé neuronové sítě jsou systémy zpracovávající informace, které mají podobné charakteristiky jako systémy biologických neuronů. Jedním z prvotních účelů studia a rozvoje neuronových sítí byla snaha pochopit kognitivní procesy lidského myšlení. Obvyklými typy úloh, ve kterých neuronové sítě vynikají, jsou mapování vstupu na výstup, klastrování (shlukování dat s podobnými charakteristikami) a hledání omezujících podmínek pro řešení problémů. Následující kapitola popisuje strukturu biologických neuronů, které sloužily jako předloha pro umělé neuronové sítě.

3.1.1 Biologický neuron

Každý biologický neuron obsahuje somu (tělo), axon a dendrity, viz Obrázek 1.



Obrázek 1: Struktura biologického neuronu, převzato z [Torse et al. 2012]

Mezi dendrity a axony jiných neuronů je vzruch přenášen chemickými prostředky a může být v synapsi multiplikován neurotransmitery (inhibujícími nebo excitujícími). V těle neuronu jsou vzruchy kumulovány, a pokud dosáhnou určité intenzity, neuron vyšle vzruch po axonu. Lze sledovat, zda neuron „vystřelil“ či „nevystřelil“, nebo frekvenci výstřelů. Neuron přijímá vzruchy z axonů jiných neuronů svými dendrity a posílá vzruchy přes axon a jeho zakončení do jiných neuronů.

Charakteristikami společnými pro biologické i umělé neuronové sítě jsou paralelismus výpočtu, inhibice/excitace signálu, schopnost neuronu sečíst příchozí signály, ale nejklíčovější je schopnost adaptovat synapse resp. váhy,

kterými jsou jednotlivé vzruchy multiplikovány. Lidský mozek obsahuje několik triliónů synaptických spojení, rychlá adaptace těchto spojení umožňuje učení ze zkušeností. Další důležitou vlastností je schopnost generalizovat resp. schopnost vyřešit takové problémy, které se do určité míry liší od problémů již známých. S tím související je schopnost abstrahovat od detailů, které nejsou potřebné ke správné klasifikaci. V neposlední řadě je podstatná schopnost neuronových sítí tolerovat poškození svých částí. Lidský mozek obsahuje přibližně 19 – 23 miliard neuronů [Pakkenberg a Gundersen 1997] a přibližně $0.15 * 10^{15}$ synapsí a v případě ztráty neuronů podstatných pro výpočet nějaké funkce můžou jiné neurony zaujmout jejich místo právě díky adaptibilitě vah.

První umělé neuronové sítě vynalezli [McCulloch a Pitts 1943]. Byly to sítě s následujícími vlastnostmi:

1. Do každého neuronu mohlo vést více spojení s pozitivními i negativními váhami.
2. Všechna spojení vedoucí do určitého neuronu byla stejné síly, byla-li pozitivní
3. Pokud neuron dostal aspoň jeden negativní signál, nevyslal signál
4. Každý neuron měl mez, a pokud suma příchozích signálů mez přesáhla, neuron vyslal signál

Tyto sítě byly využívány především pro výpočet logických funkcí, jako např. logický součet (AND), logický součin (OR), exkluzivní disjunkce (XOR) atd. Architektura (návrh neuronů a jejich spojení) stejně jako váhy pro jednotlivá spojení a meze pro jednotlivé neurony byly pevně nastaveny. Přechod vzruchu z jednoho neuronu na druhý byl chápán jako jeden diskrétní časový úsek.

3.2 Hebbovy sítě

Jedním z úkolů počátečních neuronových sítí byla klasifikace vstupních dat. Tyto sítě měly jednu vrstvu vstupů a jeden výstupní neuron, jehož výstup rozlišoval dva vzory. Výstupy všech neuronů byly buď binárního [0, 1], nebo bipolárního [-1, 1] charakteru. Výstup výstupního neuronu pro takovou síť byl definován jako:

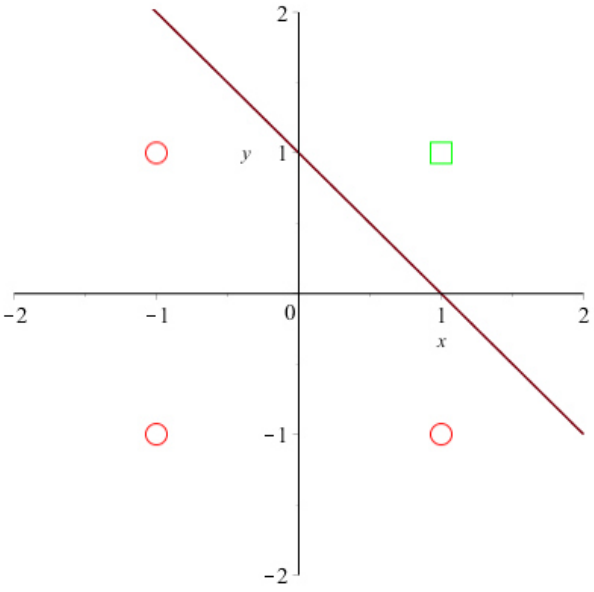
$$z = b + \sum_{i=1}^{n} w_i x_i$$

$$y_{out} = \begin{cases} 1\ pro\ z \geq 0 \\ -1(0)\ pro\ z < 0 \end{cases}$$

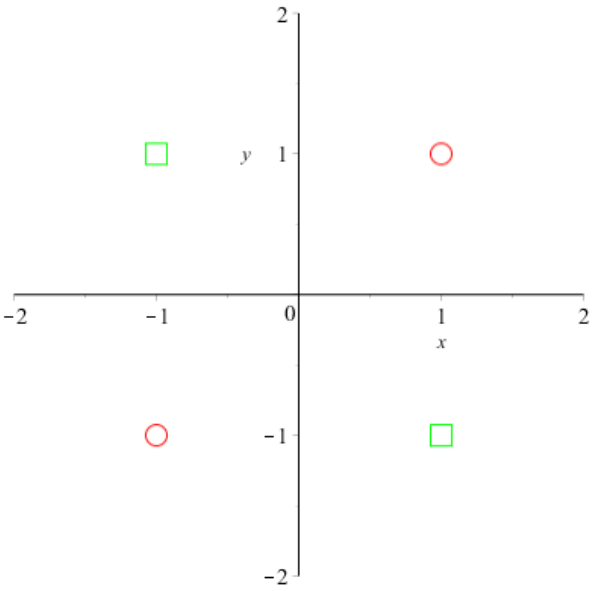
Kde n je počet vstupních neuronů (dimenze vstupních dat), b je bias (číselná hodnota, která vždy ovlivňuje výsledek výpočtu určitým způsobem bez ohledu na vstupní data), w_i je váha pro i-tý neuron a x_i je vstupní hodnota pro i-tý neuron.

Takový výstup dokáže rozlišit dva vzory.

Je třeba podotknout, že Hebbova síť nedokáže vyřešit všechny druhy problémů. Problémy, které nejsou řešitelné, tzn. nelze určit váhy, které by se 100% spolehlivostí dokázaly klasifikovat všechny vstupní vektory, jsou rozebrány v [Minsky et al. 2017] a souvisí především s tzv. linerální separabilitou. Prostor vstupů lze definovat jako n-dimenzionální prostor, kde n je počet vstupních neuronů. Potom takový stav vah, vstupů a biasu, kdy $z = 0$, reprezentuje přímku (pro $n = 2$), rovinu (pro $n = 3$), nebo nadrovinu (pro $n > 3$). Tento stav je tedy diskriminátorem, který dělí prostor vstupů na dvě oblasti. Pokud vstup leží v jedné oblasti, klasifikuje se jako 1, pokud v opačné, tak -1 (0). Potom lze říct, že problém je lineárně separovatelný právě tehdy, když existují váhy a bias, které tvoří diskriminátor, který dokáže správně oddělit (klasifikovat) všechny vstupy. Příklad logické funkce AND (Obrázek 2) splňující lineární separabilitu a logické funkce XOR (Obrázek 3) nesplňující lineární separabilitu.



Obrázek 2: Lineární separabilita logické funkce AND a jeden z možných diskriminátorů



Obrázek 3: Lineární neseparabilita logické funkce XOR

První obrázek znázorňuje diskriminátor oddělující vstupy pro logickou funkci AND. Druhý obrázek znázorňuje logickou funkci XOR, která není lineárně separovatelná.

Hebbovy sítě byly první sítě vyznačující se učícím algoritmem, tzn. algoritmem, který je schopen nalézt takové váhy, které správně mapují (klasifikují) vektory vstupních dat. Myšlenkou bylo zvýšení síly synapse (váhy spojení), pokud

„sousedící“ neurony ve stejný okamžik mají stejný výstup. Algoritmus lze popsat následujícím způsobem:

1. Váhy (včetně biasu) se inicializují na 0.
2. Pro všechny vstupní vektory se vypočítají a sečtou změny vah a biasu

$$\Delta w_i = x_i t$$

$$\Delta b = t$$

   Kde Δw_i je změna váhy i-tého neuronu, x_i je výstup i-tého neuronu, Δb je změna biasu a t je skutečná výstupní hodnota.
3. Upraví se váhy a bias

Z rovnic je zřejmé, že pokud je výstupní hodnota rovna 0, váhy systému se nezmění, což omezuje učení. Bipolární neuron je pro tento typ učení výhodnější. Další výhodou bipolárního vstupního neuronu je možnost reprezentace neznámých dat jakožto 0.

V praxi se ukázalo, že tento algoritmus není schopen najít řešení pro všechny lineárně separovatelné typy problémů. Jeden z příkladů uvádí Tabulka 1.

x_1	x_2	x_3	t
1	1	1	1
-1	1	1	-1
1	-1	1	-1
1	1	-1	-1

Tabulka 1: Příklad lineárně separovatelného problému, který nelze vyřešit za pomocí Hebbových sítí, převzato z [Fausett 1994, s. 56]

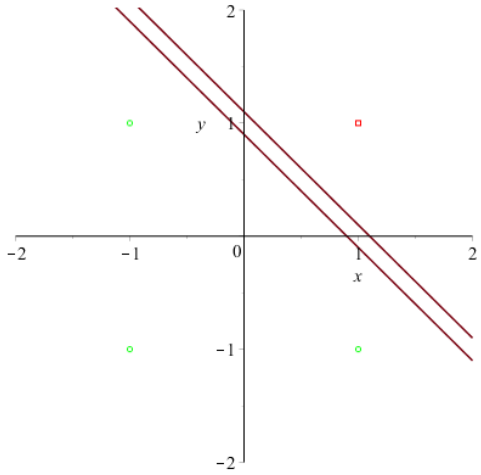
Tento problém je podrobně popsán v [Fausett 1994, s. 56] včetně krokování algoritmu.

3.3 Perceptrony

Učící metoda perceptronů měla největší dopad na strojové učení ze všech jednoduchých sítí. Zde je aktivace výstupního neuronu většinou charakterizovaná:

$$y_{out} = \begin{cases} 1\ pro\ y_{in} \geq \theta \\ 0\ pro - \theta \leq\ y_{in} < \theta \\ -1\ pro\ y_{in} < \theta \end{cases}$$

Kde θ je pevně stanovená mez (kladné číslo). Z výše uvedeného je jasné, že diskriminátor je zde charakterizován dvojící přímek (příp. rovin, nadrovin). Jedna je hranicí mezi prostory jedné kategorie a neurčitým prostorem. Druhá je hranicí mezi prostorem druhé kategorie a neurčitým prostorem, viz Obrázek 4.

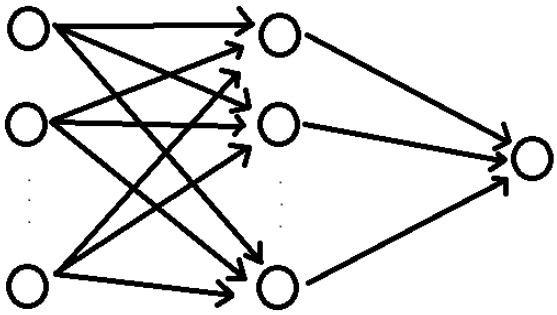


Obrázek 4: Příklad diskriminátoru vymezující neurčitý prostor při klasifikaci logické funkce AND

Parametr θ ovlivňuje „tloušťku“ diskriminátoru. Pokud se vstup nachází v pásmu neurčita, nemůžeme rozhodnout o jeho třídě. Samotná učící metoda se podobá Hebbově metodě. Rozdíl je v tom, že k úpravě vah a biasu dochází pouze po chybné klasifikaci a k učení dochází procházením trénovací sady do doby, kdy už nedojde k chybě. Pravidla pro úpravu jsou stejná jako u Hebbova učení. Věta o konvergenci perceptronů [Novikoff 1963] říká, že existují-li takové váhy, které správně klasifikují všechny vstupy na výstupy, učící metoda perceptronů dokáže tyto váhy najít v konečném čase.

3.4 Vícevrstvé neuronové sítě

Jedním z úskalí perpectronů byl fakt, že perceptrony nedokážou vyřešit problémy, jejichž řešení nejsou lineárně separovatelná. Aby byl tedy schopen perceptron vyřešit takový problém, musí být chytře zvoleny vstupní parametry. Pro komplikovanější problémy (např. rozpoznávání obrazu) to není jednoduché. Myšlenkou tedy byly takové sítě, které by samy dokázaly určit takové reprezentace vstupů, které jsou pro řešení daného problému nejvhodnější. Taková síť by měla jednu vrstvu vstupních parametrů, jednu skrytou vrstvu parametrů určených samotnou sítí a jednu výstupní vrstvu, viz Obrázek 5.



Obrázek 5: Příklad vícevrstvé neuronové sítě

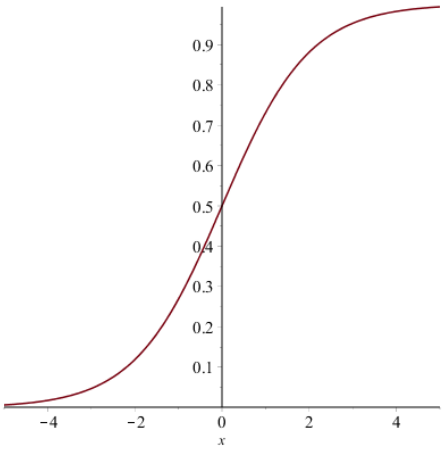
Výpočet výstupu k-tého neuronů skryté a výstupní vrstvy:

$$z_k = b + \sum_{i=1}^{n} w_{i,k} x_i$$

$$x_k = a(z_k)$$

Kde $w_{i,k}$ je váha mezi i-tým neuronem předchozí vrstvy a k-tým neuronem aktuální vrstvy, b je bias předchozí vrstvy, x_i je výstup i-tého neuron předchozí vrstvy a $a(z_k)$ je aktivační funkce neuronu. Taková funkce je často nelineární, což umožňuje vyjádřit nelineární vztah mezi vstupy a výstupy, což řeší problém, se kterým se potýkaly perceptrony. Často se používá logistická sigmoida (Obrázek 6)

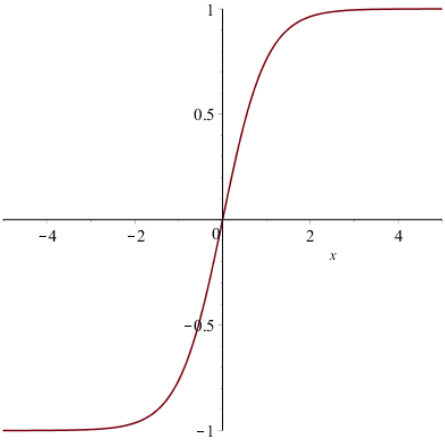
$$a(x) = \frac{1}{1 + e^{(-x)}}$$



Obrázek 6: Logistická sigmoida

nebo hyperbolický tangent (Obrázek 7).

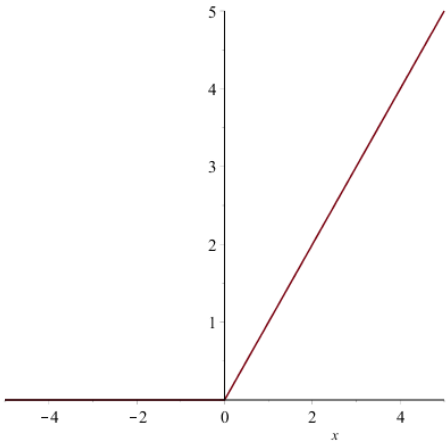
$$a(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$



Obrázek 7: Hyperbolický tangent

V hlubokých neuronových sítí se často používá rektifikovaná lineární aktivační funkce (Obrázek 8), která na základě provedených pokusů [Zeiler et al. 2013] dosahuje například v oblasti rozpoznávání řeči lepších výsledků, než logistická sigmoida.

$$a(x) = \max(0, x)$$



Obrázek 8: Rektifikovaná lineární funkce

Další úlohou aktivační funkce je omezení hodnoty výstupu neuronů. Pokud by byl výstup příliš velký, musely by k patřičnému mapování být váhy velmi malé. To by mohlo působit potíže v přesnosti, vzhledem ke způsobu ukládání čísel s plovoucí desetinou čárkou.

3.5 Zpětná propagace

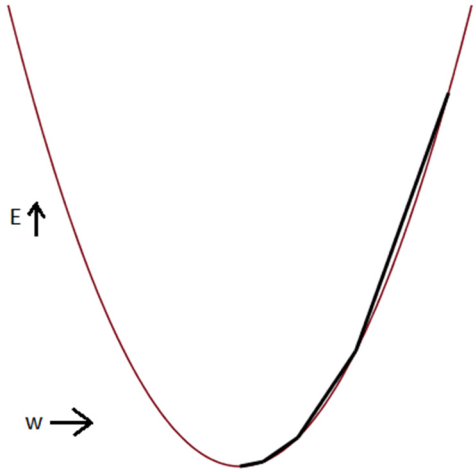
Algoritmus zpětné propagace slouží k nalezení optimálních vah skrytých vrstev minimalizací chyby vzniklé při dopředném výpočtu. Algoritmus byl navržen pro tzv. dopředné neuronové sítě. Pro takové sítě platí, že jejich neurony jsou rozděleny do vrstev a pokud existuje spojení mezi dvěma neurony $x_{O,P}$ a $x_{K,L}$ (kde O a K jsou označení vrstev), pak O > K. Algoritmus začíná inicializací vah malými náhodnými čísly. V praxi se pro rektifikovanou lineární funkci osvědčila inicializace vah neuronu v rozsahu $<-2/\Omega,2/\Omega>$, kde Ω je počet neuronů v předcházející vrstvě propojených s neuronem [DAWSON a WILBY 1998, s. 5]. Poté se provede výpočet pro všechny vstupy. Pro hodnocení modelu musí být zavedena výkonnostní funkce, někdy též označovaná jako chybová funkce. Aby šlo využít zpětné propagace, musí být tato funkce diferencovatelná vzhledem k vahám. Standardně se používá střední kvadratická chyba:

$$E_p = \frac{1}{2}\sum_{i=1}^{n}(t_{p,i} - y_{p,i})^2$$

Kde n je počet výstupních neuronů, p je aktuální tréninkový případ, t je skutečná výstupní hodnota neuronu a y je výstupní hodnota neuronu. Celková chyba je součtem dílčích chyb výstupních neuronů:

$$\mathrm{E} = \sum_{p} \mathrm{E}_{p}$$

Z rovnic je zřejmé, že hodnota chybové funkce roste proporcionálně s odchylkou skutečné hodnoty od vypočítané. Snahou je tedy najít takové váhy, pro které by hodnota chybové funkce byla minimální. Pokud model neobsahuje skryté vrstvy, chybová funkce je konvexní s jedním globálním minimem. Jedním z možných řešení by bylo spočítat globální minimum analytickou cestou (parametry pro které se hodnota první derivace rovná nule). S rostoucím počtem parametrů je tato metoda neefektivní z hlediska výpočetní složitosti. Jinou možností je iterativně činit malé kroky ve směru gradientu (největšího spádu) do doby nalezení minima. Tato metoda se označuje jako gradientní sestup (Obrázek 9).



Obrázek 9: Gradientní sestup u konvexní chybové funkce

Pro jeden skok je třeba upravit váhy proporcionálně hodnotě gradientu (parciální derivace chybové funkce vzhledem k váhám):

$$\Delta_{p} w_{j,i} = \alpha * (-\frac{\partial E_{p}}{\partial w_{j,i}})$$

Kde α je velikost skoku (rychlost učení). Při velmi malém α může učení trvat příliš dlouho. V opačném případě může řešení divergovat. Pro výše definovanou

chybovou funkci by úprava libovolné váhy měla být proporcionální součinu chyby neuronu, do něhož ústí spojení, a výstupu neuronu na opačném konci spojení:

$$\Delta_p w_{j,i} = \alpha \delta_{p,j} o_{p,i}$$

Chyba neuronu ve výstupní vrstvě je rovna součinu rozdílu skutečné hodnoty a vygenerované hodnoty a hodnoty první derivace aktivační funkce pro celkový input neuronu ($net_{p,j}$):

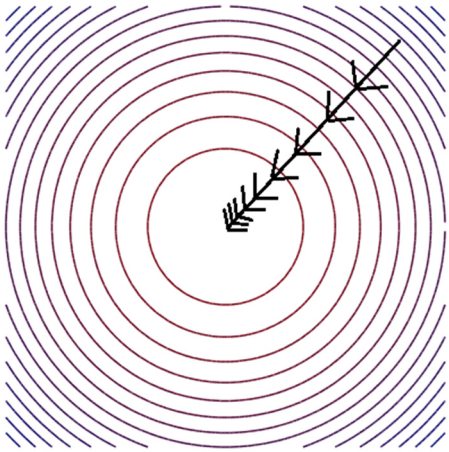
$$\delta_{p,j} = \left(t_{p,j} - o_{p,j}\right) f'(net_{p,j})$$

Po výpočet chyb všech neuronů v síti je nutné spočítat chyby neuronů výstupní vrstvy a rekurzivně pak chyby pro všechny neurony skryté vrstvy (s neurony vstupní vrstvy není spojená žádná chyba). Chyba neuronů skrytých vrstev je rovna součinu derivace aktivační funkce pro celkový input neuronu a sumě součinů chyb všech neuronů spojených s aktuálním neuronem a vah jejich spojení:

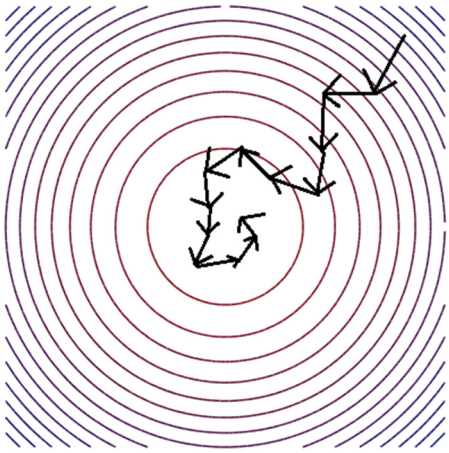
$$\delta_{p,j} = f'(net_{p,j}) \sum_k \delta_{p,k} w_{k,j}$$

Odvození těchto vzorců spolu s experimenty učení pomocí této metody lze nalézt v [Rumelhart et al. 1986]

V klasické variantě zpětné propagace je spočtena chyba pro všechny tréninkové příklady, na jejímž základě jsou vypočteny dílčí chyby pro jednotlivé neurony a upraveny váhy. Tím je garantováno přiblížení se k optimu v každé iteraci. Nevýhodou je nutnost projít všechny tréninkové příklady pro jedinou úpravu vah. Variantou je projít pouze část příkladu, spočítat chyby a upravit váhy (stochastický gradientní sestup). Extrémním příkladem je tzv. online-learning, kde dochází k úpravě vah po spočtení chyby z jednoho tréninkového případu. Obrázek 10 znázorňuje full-batch gradientí sestup a Obrázek 11 jeho stochastickou variantu.



Obrázek 10: Full-batch gradientní sestup



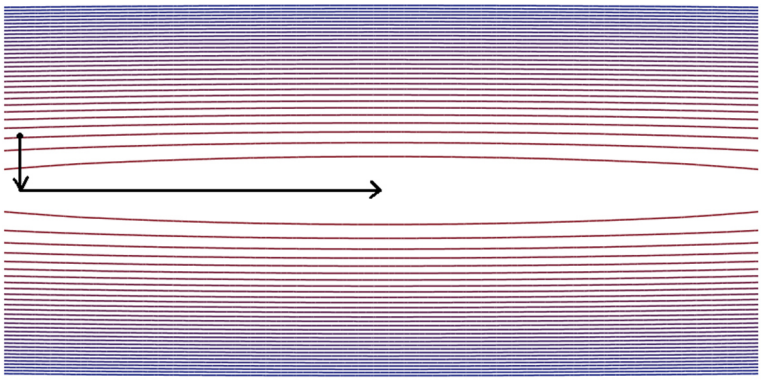
Obrázek 11: Stochastický gradientní sestup

Chybová funkce vícevrstvých sítí se vyznačuje třemi fakty, které uvádí [Zhang et al. 1998, s. 5]. Chybová funkce může mít velké množství (až nekonečno) globálních minim v důsledku permutací vah, které nemění mapovací funkci sítě. Dále se zjistilo, že chybová funkce často obsahuje ploché oblasti s velmi malým gradientem. Ty často nastávají, neboť určité kombinace vah způsobí velké vážené sumy jednoho nebo více neuronů, což po dosazení do aktivační funkce produkuje hodnoty na okrajích aktivační funkce s malým gradientem. Existence lokálních minim byla rovněž dokázána [Mclnerney et al. 1987], provedené experimenty podle [Rumelhart et al. 1986] však ukázaly, že to v mnoha případech nečiní problém. Mezi často používané chybové funkce patří střední absolutní nebo kvadratické chyby.

$$MAE = \frac{1}{n}\sum |y_{pred} - y_{true}|$$

$$MSE = \frac{1}{n}\sum (y_{pred} - y_{true})^2$$

Může se stát, že vzhledem k tvaru chybové funkce a počátečnímu umístění je potřeba činit malé skoky ve směru největšího spádu a velké skoky podél strže viz Obrázek 12



Obrázek 12: Problém zakřivení chybové funkce

V takovém případě může být přínosné použití modifikace zpětné propagace zakomponováním setrvačnostního prvku do výpočtu změny vah:

$$\Delta_p w_{j,i}(n+1) = \alpha \delta_{p,j} o_{p,i} + \beta \Delta w_{j,i}(n)$$

Kde β je konstanta určující vliv předchozích změn vah na aktuální. To pomáhá filtrovat velkou variabilitu změny vah a brání oscilaci.

Jiné optimalizační algoritmy (Adam, RMSProp) jsou popsány v následujících kapitolách.

3.6 Adam (adaptive moment estimation)

Gradientní optimalizátor má široké využití v mnoha oblastech vědy a výzkumu. Nejčastěji je zapotřebí optimalizovat cílovou funkci s parametry, která vyžaduje maximalizovat nebo minimalizovat s ohledem k těmto parametrům. V těchto případech je gradientní sestup velice výhodný, neboť výpočet hodnot první derivace funkce má stejnou výpočetní složitost jako výpočet hodnoty této funkce. Často jsou cílové funkce složené z několika podfunkcí. V takových případech je výhodnější vypočítat gradienty vzhledem k jednotlivým podfunkcím. V případech velkého množství proměnných parametrů modelu a stochastické povahy cílové funkce lze využít optimalizační algoritmus Adam, který adaptuje rychlost učení jednotlivých vah v závislosti na odhadu prvního a druhého momentu vypočtených gradientů. Metoda Adam vznikla kombinací metod AdaGrad [Duchi et al. 2011] a RMSProp [Tieleman a Hinton 2012], z nichž první upravuje rychlost učení v závislosti na frekvenci parametrů, kde pro vzácné parametry je rychlost učení

akcelerována, zatímco ve druhém případě dělíme gradient odmocninou pohybujícího se průměru druhých mocnin úpravy vah z předchozích iterací.

Algoritmus Adam, jak je uveden v [Kingma a Ba 2014]:

α … velikost kroku

$\beta_1\beta_2 \in <0,1)$ … exponenciální rychlost rozkladu pro odhady momentů

$\varepsilon := 10^{-8}$ … konstanta zabraňující dělení nulou

$f(\Theta)$ … chybová funkce vzhledem k parametrům Θ

Θ_0 … počáteční vektor parametrů

$m_0 := 0$ … inicializace vektoru prvních momentů

$v_0 := 0$ … inicializace vektoru druhých momentů

$t := 0$ … inicializace počátku

Do konvergence Θ_t {

$$t \coloneqq t + 1$$

$$g_t := \nabla_\Theta f(\Theta_{(t-1)})$$

$$m_t := \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$

$$v_t \coloneqq \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$

$$\hat{m}_t \coloneqq m_t / (1 - \beta_1^t)$$

$$\hat{v}_t \coloneqq v_t / (1 - \beta_2^t)$$

$$\Theta_t \coloneqq \Theta_{(t-1)} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$$

}

vrať Θ_t

3.7 Normalizace vstupů

Před započetím tréninku je často žádoucí normalizovat vstupní hodnoty na jednotný rozsah. Obzvlášť pokud jsou velké rozdíly mezi řády jednotlivých hodnot. „Normalizace dat na jednotný rozsah (např. 0 – 1) je nezbytná jednak pro zamezení převážení malých čísel velkými, ale také pro zamezení předčasné saturace skrytých neuronů, což by mohlo ztížit proces učení“ [Basheer a Hajmeer 2000, s. 17]. Normalizaci lze provést nejrůznějšími způsoby. [DAWSON a WILBY 1998] ve svých experimentech pro předpověď srážek použili normalizace vzhledem k rozsahu:

$$N_i = \frac{R_i - Min_i}{Max_i - Min_i}$$

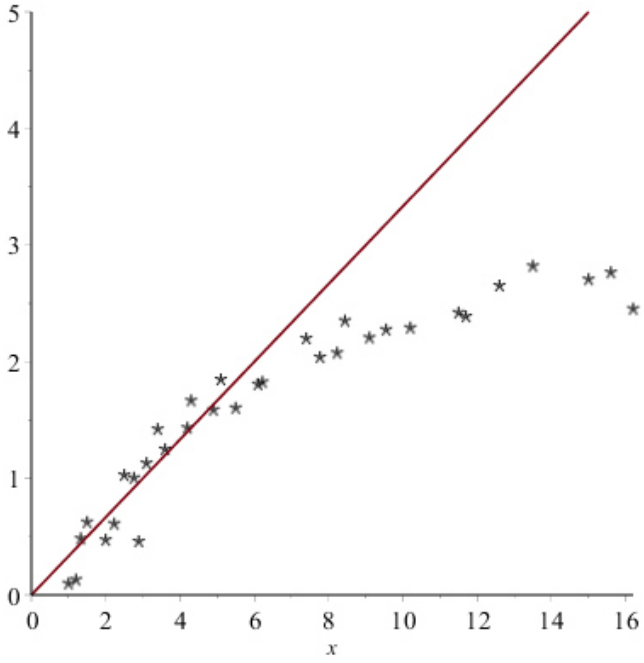
Kde N_i je normalizovaná hodnota i-tého neuronu, R_i je skutečná hodnota i-tého neuronu, Max_i a Min_i jsou maximální a minimální hodnoty i-tého neuronu. Jako druhou metodu použili normalizaci vzhledem k sumě druhých mocnin:

$$N_i = \frac{R_i}{\sqrt{SS_i}}$$

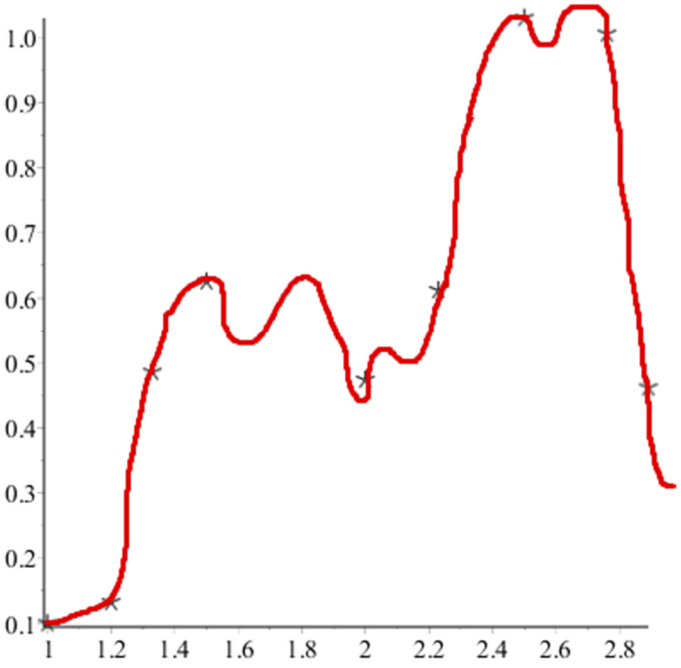
Kde SS_i je suma druhých mocnin.

3.8 Problémy učení a jejich možná řešení

Při učení modelu mohou nastat problémy, které lze rozčlenit do dvou základních typů. V praxi je kladen důraz na přesnost výsledků modelu na doposud neviděných příkladech. Věrohodným modelem je takový model, který vykazuje vysokou úspěšnost při tréninku a zároveň do jisté míry ignoruje šum v datech. Prvnímu z problémů se říká velký bias (high bias) a vypovídá o přílišné jednoduchosti modelu. Opačná situace nastává v případě, kdy je model natolik složitý, že vykazuje velkou úspěšnost při tréninku, při praktickém nasazení však má úspěšnost výrazně menší (high variance). Obě tyto situace znázorňují Obrázek 13 a Obrázek 14.

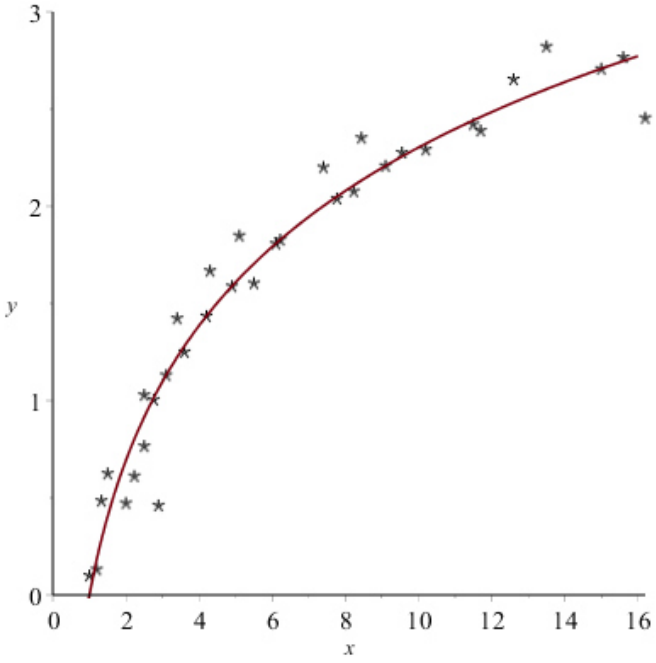


Obrázek 13: High bias



Obrázek 14: High variance

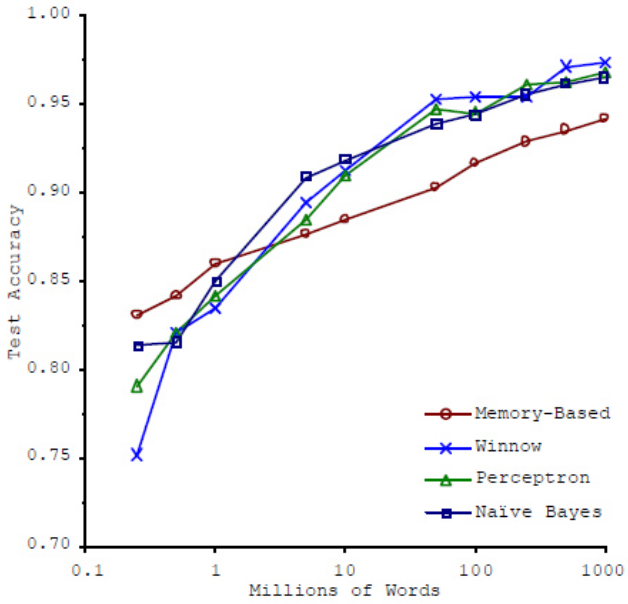
Jeden z věrohodných modelů postihuje Obrázek 15



Obrázek 15: Příklad věrohodného modelu

Aby bylo možno posoudit věrohodnost modelu, je třeba jej testovat na nové, předtím neviděné sérii dat. Při trénování toho lze dosáhnout rozdělením kompletní sady dat do trénovací, validační a testovací sady. Trénování probíhá výhradně na trénovací sadě a testování na validační a testovací sadě. Způsob alokace dat do těchto sad závisí na konkrétní problémové oblasti. Při náhodném rozdělení by mohlo dojít k nerovnoměrnému zastoupení vstupních dat v jednotlivých sadách. To lze vyřešit opakováním experimentů pro různé validační a trénovací sady. Úspěšnost modelu lze chápat jako průměrnou úspěšnost všech pokusů. Testovací sada je v rámci jednoho experimentu stejná a slouží pro posouzení kvality. Pokud model vykazuje malou chybovost na trénovací sadě a velkou na validační sadě, problémem je pravděpodobně vysoká variabilita. Pokud model vykazuje velkou chybovost i na trénovací sadě, problémem je pravděpodobně vysoký bias. Řešením problému vysokého biasu je obvykle zesložitění modelu (přidání skryté vrstvy nebo neuronů ve vrstvě, přidání vstupního parametru). Ideálním řešením vysoké variability je trénink na větším množství dat. Důležitost velkého množství dat se potvrdila ve výzkumné práci [Banko a Brill 2001], kde byl porovnán vliv množství dat a použité algoritmy na přesnost klasifikace správného slova z množiny slov často zaměňovaných. Z výsledků (Obrázek 16) lze usoudit, že pro tento typ

problému mělo množství dat mnohem větší vliv na kvalitu modelu než použitý algoritmus.



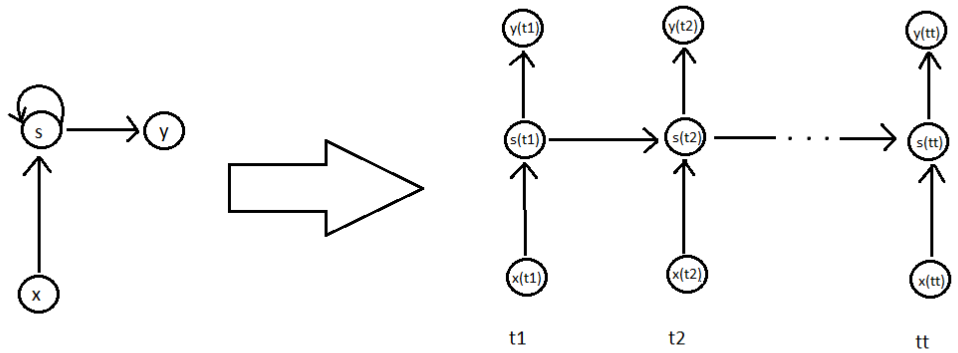
Obrázek 16: Porovnání učících algoritmů pro NLP, převzato z [Banko a Brill 2001, s. 2]

Dalším možným řešením vysoké variability je použití regularizace (L1/L2, Dropout). Myšlenkou L1/L2 regularizace je zakomponování násobku vah modelu do výpočtu chybové funkce. V případě L1 regularizace (Lasso) se jedná o sumu absolutních hodnot vah modelu. To vede k vynulování určitých vah [Tibshirani 1996] a tak v podstatě k selekci vstupních parametrů modelu. U L2 regularizace (Ringe) se k chybové funkci přičte suma druhých mocnin vah. Tendence modelu minimalizovat hodnotu určité váhy je tedy úměrná druhé mocnině hodnoty této váhy. To má za následek konvergenci k malým nenulovým vahám. [Demir-Kavuk et al. 2011] ve svém výzkumu predikce vlastností molekul na základě malého vzorku dat použili L1 regularizace k selekci parametrů a následná L2 regularizace přispěla k lepším výsledkům. Dropout je regularizační technika spočívající v náhodném vypnutí určitého zastoupení neuronů při tréninku. Každý tréninkový příklad je tak trénován na jiné „řidší“ architektuře, která sdílí váhy s ostatními. Testování probíhá standardním způsobem za využití všech neuronů, což je ekvivalentní průměrování výsledků mnoha různých modelů a provedené experimenty

[Srivastava et al. 2014] vykazují lepší výsledky z hlediska generalizace pro různé typy úloh.

3.9 Rekurentní neuronové sítě

Ačkoliv se ukázalo, že dopředné neuronové sítě dokážou reprezentovat jakoukoli nelineární funkci, nedokážou ve svém výpočtu zohlednit předchozí vstupy. Tato vlastnost je klíčová pro vyřešení úloh, kde je třeba kromě aktuálního vstupu znát i kontext. Příkladem může být předpověď časových řad, nebo zpracování přirozeného jazyka (NLP). Generování textu je problém, kdy při generaci slova je nutné znát kontext textu (rod, číslo,…). To dalo vzniknout myšlence vytvořit sítě se spojeními mezi neurony stejné vrstvy a neurony „se smyčkou“ (výstup jednoho neuronu slouží jako vstup do stejného neuronu). Všechny rekurentní architektury lze rozložit v čase na dopředné síťě. Jednoduchý příklad s jedním neuronem ve vstupní, skryté a výstupní vrstvě je znázorněn (Obrázek 17).



Obrázek 17: Rekurentní neuronová síť s jedním neuronem ve skryté vrstvě

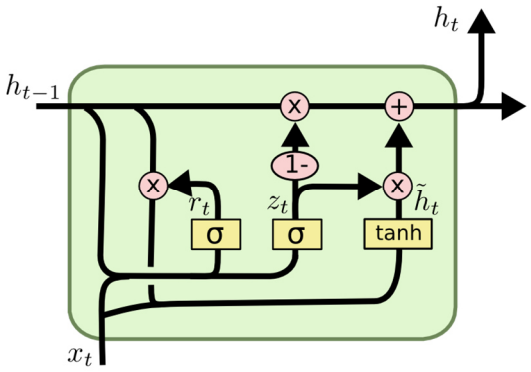
Lze je tedy chápat jako velmi hluboké dopředné sítě, kde váhy v jednotlivých časových okamžicích jsou fixní. Mechanismem pro nalezení vah je generalizace zpětné propagace, zpětná propagace v čase.

3.10 Zpětná propagace v čase

Pro úpravu vah rekurentní neuronové sítě je nejprve proveden výpočet výsledku modelu pro několik časových kroků. Poté je srovnáním výsledku výpočtu se skutečnou hodnotou vypočtena chyba. Vzorec pro výpočet chyb neuronů je stejný jako u klasické zpětné propagace. Následně je chyba zpětně propagována v čase do doby, kdy započal dopředný výpočet. Některé neurony však budou přijímat chybu ze sebe samých z předchozího časového kroku, proto je nutné ukládat si hodnoty aktivačních funkcí neuronů v čase. Na konci je změna vah rovna sumě změň vypočítaných vah v jednotlivých krocích. Metoda má však několik úskalí. Vzhledem k nutnosti uchovávat hodnoty aktivačních funkcí a změn vah v jednotlivých krocích, je tato metoda velmi náročná na paměť. Závažnějším problémem je tvz. problém vybuchujícího/mizejícího gradientu [Bengio et al. 1994], ke kterému dochází z důvodu lineárního charakteru zpětné propagace.

3.11 LSTM (Long Short-Term Memory)

Řešení komplikací vzniklých při použití zpětné propagace v čase bylo navrženo v roce 1997 [Hochreiter a Schmidhuber 1997]. Revoluční myšlenkou bylo použití sítí skládající se z tzv. paměťových buněk, viz Obrázek 18.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Obrázek 18: Long-Short Term Memory struktura paměťové buňky, převzato z [Olah 2015]

Paměťové buňky se vyznačují stavem a obsahují brány (neuronové sítě), které spolu se vstupními daty rozhodují o informačních tocích v buňce, to znamená které informace uchovat, které vstupní informace využít a do jaké míry, a jaký má být výstup.

4 Úvod do knihovny pro strojové učení - Keras

Keras [Chollet a others 2015] je open-source knihovna napsaná v programovacím jazyce Python, sloužící jakožto vysokoúrovňové rozhraní pro návrh a implementaci hlubokých neuronových sítí. Původně byl Keras vyvinut v rámci projektu ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Keras umožňuje rychlý, intuitivní vývoj a testování architektur konvenčních typů neuronových sítí (včetně konvolučních a rekurentních). Jako backend pro výpočetní činnost je možno využít TensorFlow (https://www.tensorflow.org/), CNTK (https://www.microsoft.com/en-us/cognitive-toolkit/) nebo Theano (http://deeplearning.net/software/theano/). K výpočtu lze využít CPU i GPU.
Keras poskytuje dva druhy API:

- Sekvenční model
- Funkcionální API

4.1 Sekvenční model

Sekvenční model je založen na postupném přidávání vrstev sítě a to buď v konstruktoru modelu, nebo pomocí metody add. Keras nabízí spoustu předdefinovaných vrstev (Core layers - Dense, Activation, Dropout…, Convolution layers, Reccurent layers,…), umožňuje však i definici vlastních. Na příkladu je uveden možný způsob implementace predikce sekvencí s využitím LSTM.

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.layers import Embedding
from keras.layers import LSTM

model = Sequential()
model.add(Embedding(max_features, output_dim=256))
model.add(LSTM(128))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=16, epochs=10)
score = model.evaluate(x_test, y_test, batch_size=16)
```

Obrázek 19: Příklad implementace neuronové sítě pro predikci sekvencí s využitím sekvenčního modelu knihovny Keras, převzato z [Chollet a others 2015]

4.2 Funkcionální API

Funkcionální API umožňuje tvorbu složitých modelů s více výstupy, orientované acyklické grafy a modely se sdílenými vrstvami. Model je definován na základě Input a Output tensorů a trénink lze zahájit voláním metody fit do které lze poslat list vstupních a výstupních dat (při implementaci modelu s více vstupními nebo výstupními daty). Následující příklad postihuje možnou implementaci modelu se sdílenými vrstvami.

```
import keras
from keras.layers import Input, LSTM, Dense
from keras.models import Model

tweet_a = Input(shape=(280, 256))
tweet_b = Input(shape=(280, 256))

# This layer can take as input a matrix
# and will return a vector of size 64
shared_lstm = LSTM(64)


# When we reuse the same layer instance
# multiple times, the weights of the layer
# are also being reused
# (it is effectively *the same* layer)
encoded_a = shared_lstm(tweet_a)
encoded_b = shared_lstm(tweet_b)

# We can then concatenate the two vectors:
merged_vector = keras.layers.concatenate([encoded_a, encoded_b], axis=-1)

# And add a logistic regression on top
predictions = Dense(1, activation='sigmoid')(merged_vector)

# We define a trainable model linking the
# tweet inputs to the predictions
model = Model(inputs=[tweet_a, tweet_b], outputs=predictions)

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit([data_a, data_b], labels, epochs=10)
```

Obrázek 20: Příklad implementace modelu se sdílenými vrstvami s využitím funkcionálního API knihovny Keras, převzato z [Chollet a others 2015]

5 Metodika výzkumu

V rámci výzkumné činnosti bylo provedeno několik sérií experimentů. V každé ze sérií bylo provedeno několik experimentů a každý z experimentů zahrnoval 3 pokusy. Jeden experiment byl proveden pro jedno nastavení hyper-parametrů s jednou architekturou. Pro lepší porovnání přesnosti jednotlivých experimentů byly 3 krát generovány validační a trénovací sady pro jednotlivé pokusy. Testovací sady byly totožné v rámci jedné série experimentů. Jednotlivé experimenty byly porovnány na základě průměrné hodnoty RMSE a korelačního koeficientu na validační sadě, případně na testovací sadě. Pro vybrané experimenty byly vypočítány Nash-Sutcliffe koeficienty [Nash a Sutcliffe 1970], běžně používané k vyjádření úspěšnosti hydrologických modelů.

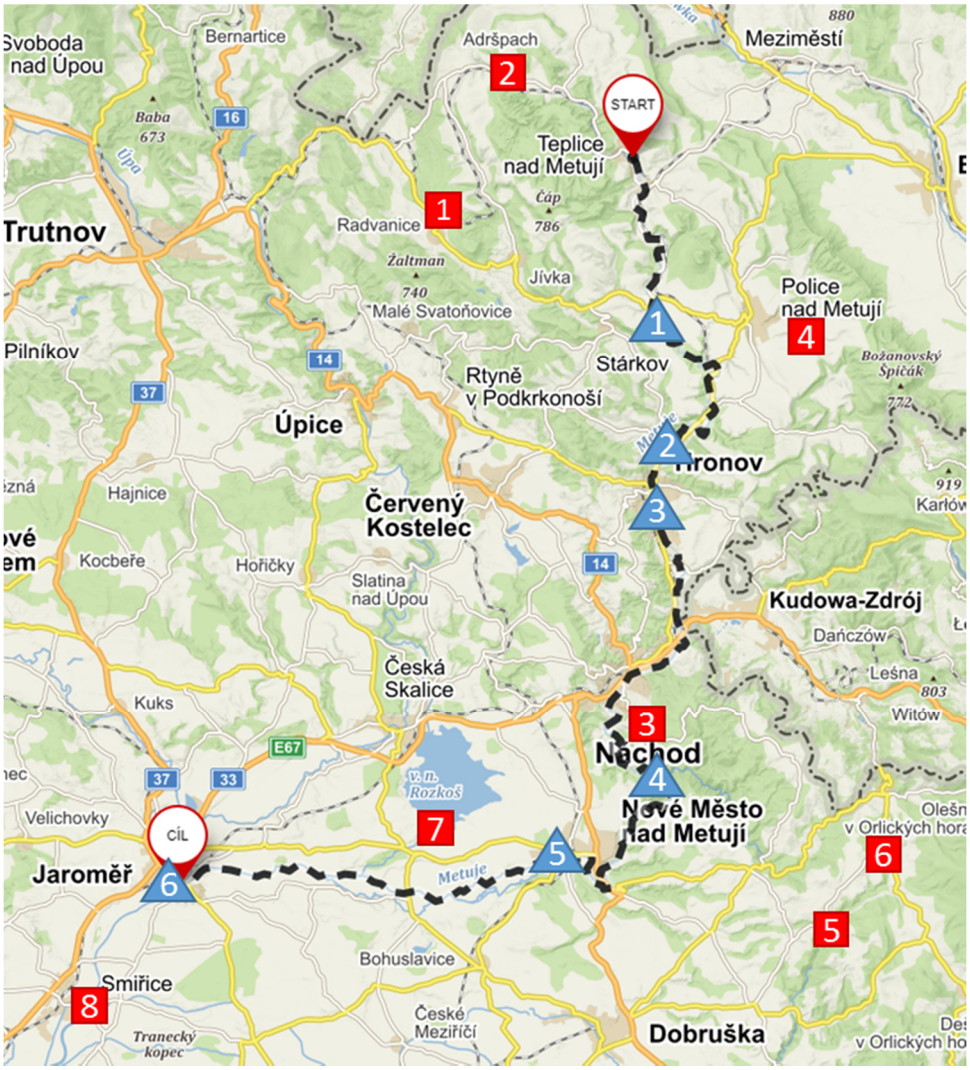
$$RMSE = \sqrt{\frac{1}{n}\sum(y_{pred} - y_{true})^2}$$

$$Korelační\ koeficient = \frac{\sum((y_{true} - \bar{y}_{true}) * (y_{pred} - \bar{y}_{pred}))}{\sqrt{\sum(y_{true} - \bar{y}_{true})^2} * \sqrt{\sum(y_{pred} - \bar{y}_{pred})^2}}$$

$$Nash - Sutcliffe\ koeficient = 1 - \frac{\sum(y_{true} - y_{pred})^2}{\sum(y_{true} - \bar{y}_{true})^2}$$

5.1 Vstupní data

Pro analýzu byla využita data z říčního toku řeky Metuje poskytnutá státním podnikem Povodí Labe a obsahující data ze senzorů výšky hladiny, teplot a srážek (Obrázek 21). K výšce hladiny byly k dispozici i odpovídající průtoky. Kompletní seznam senzorů ukazují Tabulka 2 a Tabulka 3.



Obrázek 21: Umístění senzorů pro měření výšky hladiny (modré) a senzorů srážek a teploty (červené)

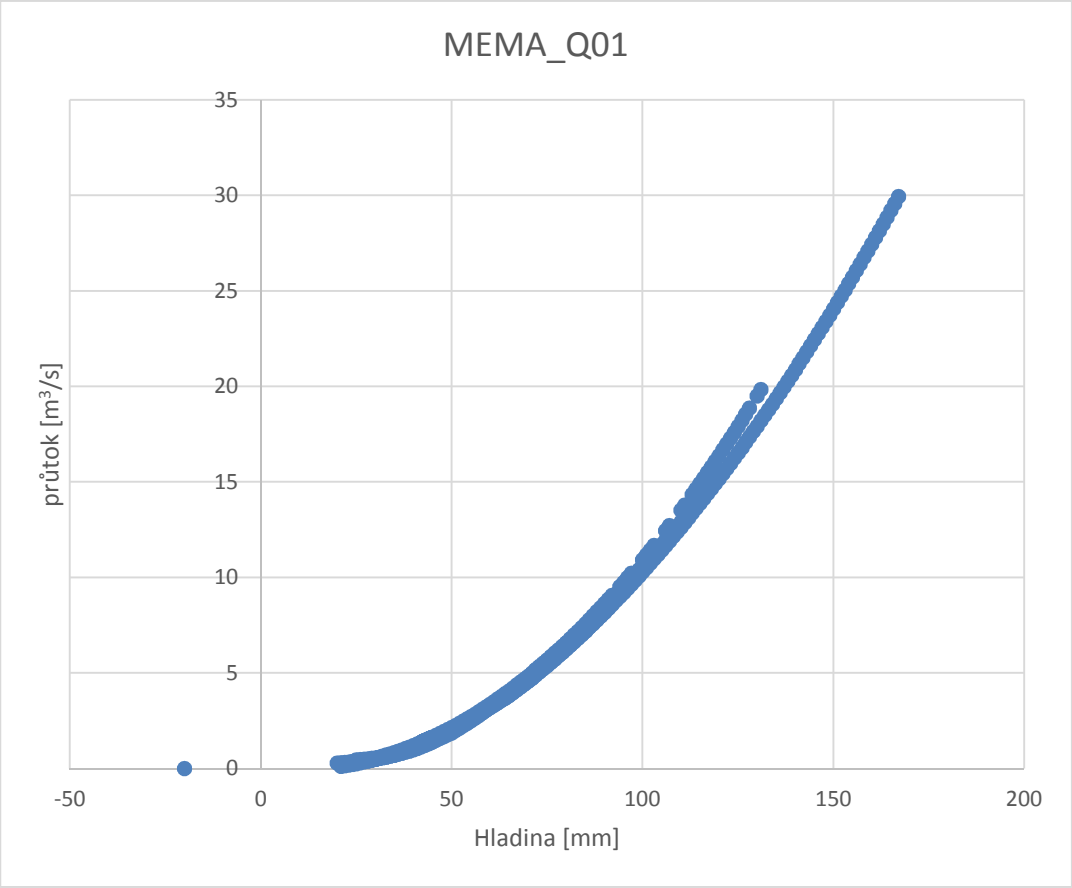
Umístění senzoru	GPS	Označení	Poznámka
Maršov	N 50,5299807 E16,1898291	MEMA_Q01	1
Dřevíč - Velký Dřevíč	N50°29'28.96 E16°10'59.622	DRVD_Q01	2 – není na Metuji, ale na přítoku
Metuje - Hronov	16.1808339 v.d. 50.4831738 s.š.	MEHR_Q01	3
Metuje – Krčín	16.1300204 v.d. 50.3516282 s.š.	MEKR_Q01	4
Metuje - Jaroměř	N50°20'28.706 E15°55'33.203	MEJA_Q01	6
Olešenka - Peklo	N50°22'44,8 E16°11'9,1	OSPE_Q01	5 – není na Metuji, ale na přítoku

Tabulka 2: Seznam senzorů průtoku a výšky hladiny

Senzor	GPS	Označení	Perioda měření	Měřeno od	Pozn.
Radvanice - Srážková intenzita (mm)	N50°34'0 E16°3'27,8	SSRDSR01	15 min	22.02.2001 10:15:00	1
Radvanice - Teplota vzduchu (°C)		SSRDTO01			
Adršpach - Srážková intenzita (mm)	Stanice ČHMÚ	SSDASR01	60 min	14.03.2008 00:00:00	2
Adršpach - Teplota vzduchu (°C)		SSDATO01			
Náchod Bražec - Srážková intenzita (mm)	N50°23'42.167 E16°9'22.807	SSBRSR01	15 min	19.07.2001 12:30:00	3
Náchod Bražec - Teplota vzduchu (°C)		SSBRTO01			
Police nad Metují - Srážková intenzita (mm)	Stanice ČHMÚ	SSPMSR01	60 min	14.03.2008 00:00:00	4
Dobřany - Srážková intenzita (mm)	N50°19'23.403 E16°17'5.5	SSDOSR01	15 min		5
Dobřany - Teplota vzduchu (°C)		SSDOTO01			
Sedloňov - Srážková intenzita (mm)	N50°21'.489 E16°19'20.681	SSSESR01	15 min	01.01.2000 00:00:00	6
Sedloňov - Teplota vzduchu (°C)		SSSETO01			
Přehrada Rozkoš - Srážková intenzita (mm)	N50°21'46.104 E16°3'43.703	PRROSR01	15 min	05.01.2001 12:45:00	7
Přehrada Rozkoš - Teplota vzduchu (°C)		PRROTO01			
Jez Smiřice - Srážková intenzita	N50°18'4.227 E15°52'39.7	LASCSR01	15 min	19.10.2010 10:00:00	8
Jez Smiřice - Teplota vzduchu (°C)		LASCTO01			

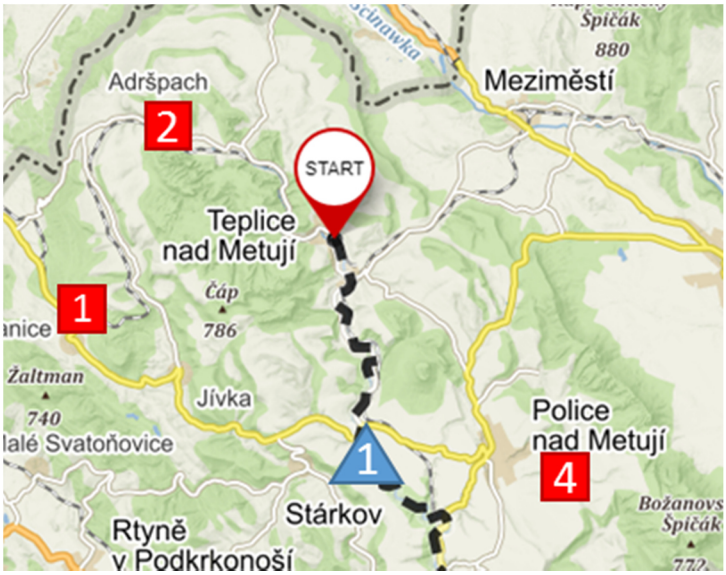
Tabulka 3: Seznam senzorů pro měření srážek a teploty

Pro další analýzu byl s primárními daty proveden základní rozbor a pročištění. Na základě analýzy vztahu průtok-vodní stav (Obrázek 22) byly zjištěny změny v charakteristikách způsobené možnou úpravou profilu nebo metodiky měření. Protože na ostatních senzorech byly detekovány i významnější odlišnosti, bylo po konzultaci s pracovníky Povodí rozhodnuto, že pro další analýzu budou použita pouze data o průtoku, která lépe charakterizující aktuální situaci na toku a neměla by být zatížena případnými změnami na profilu toku.



Obrázek 22: Graf závislosti průtok-hladina v Maršově nad Metují. Dvě charakteristiky ukazují změnu závislosti způsobenou možnou úpravou profilu nebo metodiky měření.

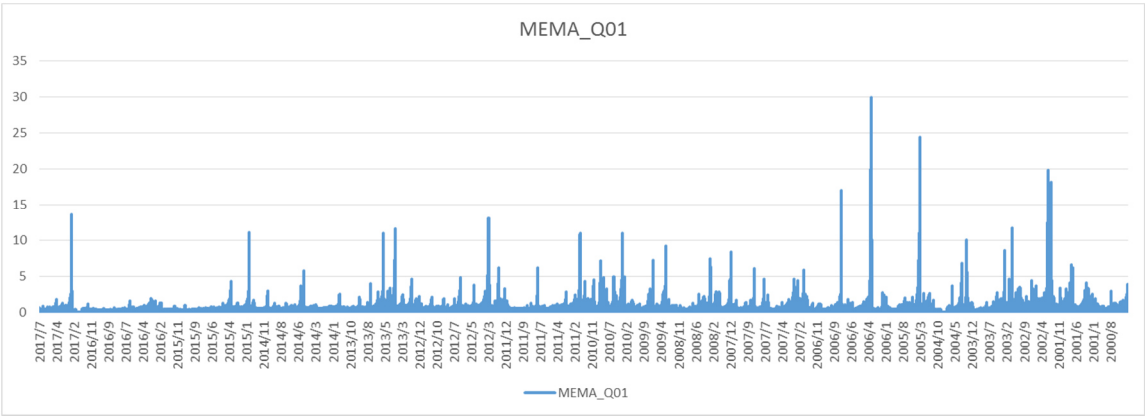
Pro pilotní analýzu byla vybrána oblast horního toku, konkrétně senzor výšky hladiny a průtoku v Maršově nad Metují. Tomuto senzoru nepředchází žádný další senzor hladiny, a proto jsou naměřené údaje o průtoku ovlivněny především srážkami v dané oblasti, nikoli přítokem z horních částí toku. Údaje o srážkách a teplotách byly zahrnuty pouze ze senzorů umístěný v nejbližším okolí Maršova, tj. Radvanice, Adršpach a Police nad Metují (Obrázek 23). Byly určeny základní číselné charakteristiky těchto senzorů (Tabulka 4).



Obrázek 23: Umístění senzorů v okolí Maršova nad Metují

Označení	Umístění senzoru, měřená veličina	Min	Max	Průměr	Počet měření
MEMA_Q01	Metuje - Maršov nad Metují - vodní stav (cm)	-20	167	35,005	517613
MEMAVS01	Metuje - Maršov nad Metují - průtok (m^3/s)	0	29,9	0,896	517637
SSRDSR01	Radvanice - Srážková intenzita (mm)	0	22,3	0,023	569195
SSRDTO01	Radvanice - Teplota vzduchu (°C)	-27,2	35,1	6,635	564455
SSDASR01	Adršpach - Srážková intenzita (mm)	0	51,6	0,073	74880
SSDATO01	Adršpach - Teplota vzduchu (°C)	-31,1	34,1	6,897	74098
SSPMSR01	Police nad Metují - Srážková intenzita (mm)	0	35,3	0,061	80304

Tabulka 4: Základní charakteristiky vstupních dat ve vybrané lokalitě horního toku



Obrázek 24: Průběh průtoků ve sledovaném období

5.1.1 Výběr a příprava dat

Pro další zpracování byla použita data o průtoku (senzor MEMA_Q01), teplotní (sensory SSRDTO01, SSDATO01) a srážkové (senzory SSRDSR01, SSDASR01, SSPMSR01) údaje naměřené v okolí. Použitá metoda strojového učení probíhá na jednotlivých učících vzorcích. Každý vzorek představuje jeden časový okamžik, ke kterému jsou vztažena všechna měření. Proto bylo nutné všechny údaje přepočítat na jednu pravidelnou periodu a vybrat intervaly, v kterých byly dostupné všechny hodnoty. Vzhledem k četnosti dostupných dat byla zvolena hodinová perioda a všechny údaje byl k tomuto časovému intervalu přepočítány. Pro učení neuronové sítě je třeba znát nejen všechna data aktuálního vzorku, ale zároveň i historii průběhu jednotlivých veličin na základě následujících pravidel. Proto byly hodnoty zároveň agregovány do intervalů ½, 1, 2, 3, 7, 30, 90, 180, 360 dní. Pravidla pro čištění dat byla následující:

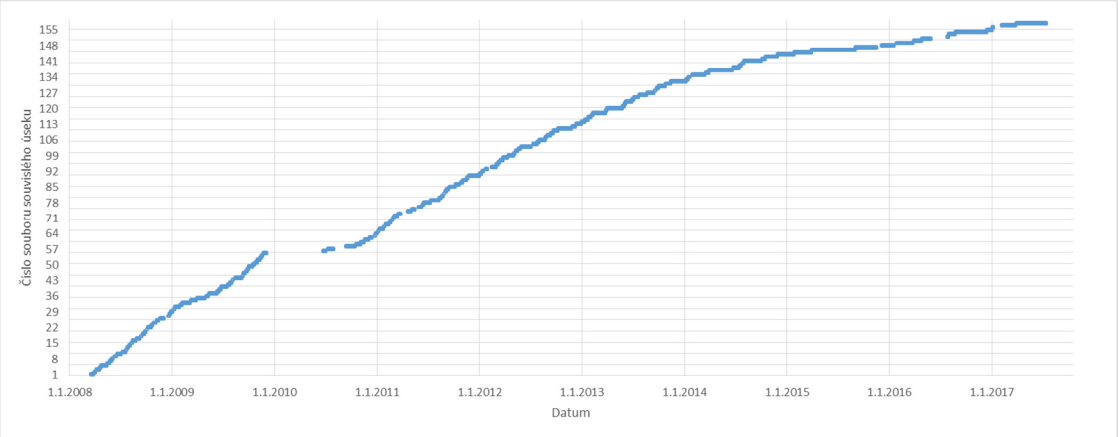
- Srážková intenzita měřená v kratších než 60 minutových intervalech byla sumarizována
- Teplotní údaj měřený v kratších než 60 minutových intervalech byl vzorkován v celou hodinu
- Výpadek měření kratší než 3 hodiny byl lineárně interpolován z okolních hodnot
- V případě chybějících dat alespoň z jednoho senzoru byl daný vzorek ze vstupního souboru odstraněn
- Souvislé úseky kratší než 3 dnů (72 hodin) byly vypuštěny

Takto bylo získáno 158 souvislých úseků (celkem s 66675 vzorky v hodinových intervalech) s kompletními měřenými i agregovanými údaji v následující struktuře (Tabulka 5). Jednotlivé úseky představují různě dlouhé souvislé posloupnosti hodinových dat z období od 18. 3. 2008 do 13. 7. 2017. V tomto sledovaném období se nachází několik delších výpadků dat, např. v první polovině roku 2010 (Obrázek 25). Navíc bylo nutno zcela vypustit řadu měření, která předcházela roku 2008 a zahrnovala několik stavů odpovídající stupňům povodňové aktivity pohotovosti (průtok nad 15,4 m^3s^{-1}) a ohrožení (průtok nad 24,2 m^3s^{-1}). Je to způsobeno tím, že

senzory srážek a teploty umístěné v Adršpachu a Polici nad Metují poskytují data až od roku 2008.

Název sloupce	**Popis**
index	původní index vzorku
tt	časový údaj přepočtený na jednu hodnotu
ROK, MESIC, DEN, HODINA, MINUTA, SEKUNDA	aktuální čas vzorku
MEMA_Q01, MEMA_Q01_0.5, MEMA_Q01_1, MEMA_Q01_2, MEMA_Q01_3	okamžitý aktuální průtok a průměry za předchozí ½, 1, 2, 3 dny
MEMAVS01	aktuální výška hladiny
SSDASR01, SSDASR01_0.5, SSDASR01_1, SSDASR01_2, SSDASR01_3, SSDASR01_7, SSDASR01_30, SSDASR01_90, SSDASR01_180, SSDASR01_360	okamžité aktuální srážky a úhrny za předchozí ½, 1, 2, 3, 7, 30, 90, 180, 360 dní
SSPMSR01, SSPMSR01_0.5, SSPMSR01_1, SSPMSR01_2, SSPMSR01_3, SSPMSR01_7, SSPMSR01_30, SSPMSR01_90, SSPMSR01_180, SSPMSR01_360	
SSRDSR01_H, SSRDSR01_H_0.5, SSRDSR01_H_1, SSRDSR01_H_2, SSRDSR01_H_3, SSRDSR01_H_7, SSRDSR01_H_30, SSRDSR01_H_90, SSRDSR01_H_180, SSRDSR01_H_360	
SSDATO01, SSDATO01_0.5, SSDATO01_1, SSDATO01_2, SSDATO01_3, SSDATO01_30, SSDATO01_90, SSDATO01_180, SSDATO01_360	okamžitá aktuální teplota a průměry za předchozí ½, 1, 2, 3, 30, 90, 180, 360 dní
SSRDTO01, SSRDTO01_0.5, SSRDTO01_1, SSRDTO01_2, SSRDTO01_3, SSRDTO01_30, SSRDTO01_90, SSRDTO01_180, SSRDTO01_360	

Tabulka 5: Struktura dat po předzpracování



Obrázek 25: Souvislé úseky ve sledovaném období

Pro učení neuronové sítě byla data ještě rozšířena i o okamžité hodnoty jednotlivých veličin v blízké historii (-1, -2, -3, -4, -5 hodin) a také budoucí stavy u veličin, které budou vstupem (teplota, srážky) i výstupem (průtok) předpovědi. Budoucí hodnoty (při reálném nasazení získané z meteorologických modelů) byly reprezentovány srážkami a teplotami (okamžitými v budoucnosti +1, +2, +3, +4, +5, +6 hodin a souhrnnými za následujících 12 a 24 hodin). Celkem bylo použito při učení neuronové sítě 125 vstupních hodnot. Tato data sloužila pro predikci průtoku za 24 hodin, který mohl být srovnáván se skutečným průtokem. Seznam všech vstupních neuronů je následující:

- Aktuální čas: rok, měsíc
- Průtok okamžitý: 0h, -1h, -2h, -3h, -4h, -5h (MEMA_Q01)
- Průtok průměr za -½d, -1d, -2d, -3d
- Senzory teploty (SSRDTO01, SSDATO01)
  - Okamžité 0h, -1h, -2h, -3h, -4h, -5h
  - Průměrné -½d, -1d, -2d, -3d, -30d, -90d, -180d, -360d
  - Predikované +1h, +2h, +3h, +4h, +5h, +6h, průměr za následujících 12h, 24h
- Senzory srážek (SSRDSR01, SSDASR01, SSPMSR01)
  - Okamžité 0h, -1h, -2h, -3h, -4h, -5h
  - Kumulativní -½d, -1d, -2d, -3d, -7d, -30d, -90d, -180d, -360d
  - Predikované +1h, +2h, +3h, +4h, +5h, +6h, suma za následujících 12h, 24h

Před vstupem do neuronové sítě byla data normalizována odečtením střední hodnoty a dělením směrodatnou odchylkou.

Výstupním neuronem (předpovídanou hodnotou) byl okamžitý průtok za 24 hodin. Byly provedeny i experimenty se sítěmi s výstupem za 1, 6, 12 a 24 hodin, blízké predikční časy však příliš odpovídaly aktuálnímu průtoku, proto byly sítě zjednodušeny na jediný výstupní neuron. Snaha predikovat více budoucích stavů by mohla negativně ovlivnit přesnost jednotlivých výstupů.

5.1.2 Rozdělení dat

Po pilotní sérii experimentů bylo pro hlavní série experimentů použito rozdělení na trénovací, validační a testovací sadu podle následujícího algoritmu. Z celkové množiny sekvencí (souvislých úseků) byly náhodně vybírány přilehlé sekvence do testovací sady, dokud délka nepřesáhla 1 rok. Ze zbylých sekvencí byla stejným způsobem vybrána validační sada. Zbytku bylo využito k trénování. Počet pozorování takto získaných byl 62 093, z čehož přibližně 44 800 připadlo na trénování, 8 600 na validování a testování.

Ve snaze zpřesnit předpovědi povodňových epizod byla v některých experimentech trénovací data vyfiltrována tak, aby rozdíl mezi aktuálním průtokem a předpovídaným byl větší nebo roven 0.06 (10% pozorování s menším rozdílem bylo zachováno). Tím došlo k redukci trénovacích dat na cca 10 tisíc.

5.2 Hledání optimální architektury a hyper-parametrů

Porovnáván byl vliv použité architektury, chybové funkce, inicializace vah a vážení chybové funkce na RMSE. Přehled architektur uvádí Tabulka 6.

	Počet skrytých vrstev	Počet neuronů ve vrstvách
Architektura 1	1	32
Architektura 2	1	64
Architektura 3	2	64
Architektura 4	2	256
Architektura 5	3	128
Architektura 6	3	256
Architektura 7	4	128
Architektura 8	4	256

Tabulka 6: Porovnávané architektury neuronových sítí

V pilotních experimentech byly použity standardní chybové funkce (MAE, MSE). Hodnoty těchto funkcí však můžou dosahovat malých hodnot i při velkých chybách na epizodách, které se dají považovat za vzácné jevy v datech. To vedlo k myšlence regularizovat chybovou funkci různými způsoby, např. korelačním koeficientem. Byly porovnávány následující funkce:

$$Correlation\ loss = 1 - \left|\frac{\sum((y_{true} - \bar{y}_{true}) * (y_{pred} - \bar{y}_{pred}))}{\sqrt{\sum(y_{true} - \bar{y}_{true})^2} * \sqrt{\sum(y_{pred} - \bar{y}_{pred})^2}}\right|$$

$$MAE^3 = \frac{1}{n}\sum |y_{pred} - y_{true}|^3$$

$$MAE^5 = \frac{1}{n}\sum |y_{pred} - y_{true}|^5$$

$$MAE\ Reg\ Corr\ 1 = MAE * (1 + Correlation\ loss)$$

$$MAE\ Reg\ Corr\ 2 = \frac{1}{n}\sum \frac{|y_{pred} - y_{true}|}{(1.5 * \left|\frac{\sum((y_{true} - \bar{y}_{true}) * (y_{pred} - \bar{y}_{pred}))}{\sqrt{\sum(y_{true} - \bar{y}_{true})^2} * \sqrt{\sum(y_{pred} - \bar{y}_{pred})^2}}\right|)}$$

Porovnány byly následující inicializace:

I1 = váhy pro určitý neuron jsou generovány z rovnoměrného rozdělení $\left[\frac{-2}{\sigma}, \frac{2}{\sigma}\right]$, kde σ je počet neuronů z předchozí vrstvy

I2 = váhy pro určitý neuron jsou generovány z normálního rozdělení s průměrem 0 a směrodatnou odchylkou $\frac{1}{\sqrt{\sigma}}$.

I3 = váhy pro určitý neuron jsou generovány z normálního rozdělení s průměrem 0 a směrodatnou odchylkou 0.05, kde hodnoty vzdálené od průměru o více než dvojnásobek směrodatné odchylky jsou zahozeny a znovu vybrány.

Porovnáván byl rovněž i účinek vážení chybové funkce hodnotou skutečného průtoku navýšeného o jedničku. Regularizace dropoutem byla aplikována na všechny skryté vrstvy a aktivační funkcí skrytých vrstev byla rektifikovaná lineární funkce.

6 Výsledky

V následujících kapitolách jsou uvedeny výsledky porovnávající jednotlivé hyper-parametry a architektury.

6.1 Porovnání architektur sítí

Pro porovnání architektur byla použita filtrovaná data, počet epoch činil 500 a chybovou funkcí byla MSE. Byla použita inicializace vah I1.
Tabulka 7 až Tabulka 14 ukazují výsledky jednotlivých architektur:

6.1.1 Výsledky pro architekturu 1

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0769	0.0547	0.0522	0.0613
RMSE (cross)	0.3291	0.2844	0.246	0.2865
RMSE (test)	0.3955	0.3086	0.1199	0.2747
Korelační koeficient (test)	0.1247	0.0936	0.1543	0.1242

Tabulka 7: Vliv hodnoty dropoutu na RMSE pro architekturu 1

Na architekturu 1 má aplikace dropoutu jednoznačně pozitivní vliv, což lze vypozorovat z klesající hodnoty RMSE na validační a testovací sadě. S růstem hodnoty dropoutu lze vypozorovat klesající hodnotu RMSE na trénovací sadě. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.16, 0.23 a 0.19.

6.1.2 Výsledky pro architekturu 2

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0671	0.0486	0.0482	0.0546
RMSE (cross)	0.3542	0.1973	0.1705	0.2406
RMSE (test)	0.3386	0.1365	0.3875	0.2876
Korelační koeficient (test)	0.0691	0.1174	0.1069	0.0978

Tabulka 8: Vliv hodnoty dropoutu na RMSE pro architekturu 2

Pro architekturu 2 hodnota RMSE klesá s růstem hodnoty dropoutu na validační i trénovací sadě. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.29, 0.15 a 0.12.

6.1.3 Výsledky pro architekturu 3

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0297	0.0392	0.0482	0.0390
RMSE (cross)	0.3041	0.1866	0.2001	0.2302
RMSE (test)	0.1318	0.2288	0.296	0.2189
Korelační koeficient (test)	0.0618	0.1177	0.1319	0.1038

Tabulka 9: Vliv hodnoty dropoutu na RMSE pro architekturu 3

Pro architekturu 3 byl experiment bez dropoutu tím nejhorším z hlediska hodnoty RMSE na validační sadě. S růstem hodnoty dropoutu lze vypozorovat růst hodnoty RMSE na trénovací a testovací sadě. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.27, 0.15 a 0.15.

6.1.4 Výsledky pro architekturu 4

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0238	0.0303	0.0325	0.0288
RMSE (cross)	0.1459	0.1474	0.166	0.1531
RMSE (test)	0.1277	0.225	0.2534	0.2021
Korelační koeficient (test)	0.1924	0.1978	0.2202	0.2035

Tabulka 10: Vliv hodnoty dropoutu na RMSE pro architekturu 4

Z výsledků pro architekturu 4 nelze vypozorovat zlepšení předpovědi s růstem parametru dropoutu. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.12, 0.12 a 0.1.

6.1.5 Výsledky pro architekturu 5

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.02	0.0309	0.0511	0.0340
RMSE (cross)	0.1626	0.1642	0.1823	0.1700
RMSE (test)	0.0797	0.2291	0.1467	0.1518
Korelační koeficient (test)	0.1639	0.1935	0.1789	0.1788

Tabulka 11: Vliv hodnoty dropoutu na RMSE pro architekturu 5

Pro architekturu 5 rovněž platí, že hodnota RMSE na validační sadě roste s růstem parametru dropoutu. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.14, 0.13 a 0.13.

6.1.6 Výsledky pro architekturu 6

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0197	0.0347	0.0460	0.0335
RMSE (cross)	0.1414	0.1856	0.1847	0.1706
RMSE (test)	0.1603	0.0741	0.0967	0.1104
Korelační koeficient (test)	0.1789	0.1970	0.1962	0.1788

Tabulka 12: Vliv hodnoty dropoutu na RMSE pro architekturu 6

Na architektuře 8 lze pozorovat růst hodnoty RMSE na validační sadě s růstem hodnoty dropoutu. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.12, 0.15 a 0.14.

6.1.7 Výsledky pro architekturu 7

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0206	0.0367	0.0532	0.0368
RMSE (cross)	0.1824	0.1631	0.2174	0.1876
RMSE (test)	0.0927	0.1195	0.1385	0.1169
Korelační koeficient (test)	0.1438	0.2009	0.2121	0.1856

Tabulka 13: Vliv hodnoty dropoutu na RMSE pro architekturu 7

Z výsledků pro architekturu 7 nelze vypozorovat jednoznačný trend hodnoty RMSE na validační sadě, na testovací a trénovací sadě však dochází k nárůstu. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.16, 0.13 a 0.16.

6.1.8 Výsledky pro architekturu 8

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0175	0.0301	0.0417	0.0298
RMSE (cross)	0.149	0.1555	0.1897	0.1647
RMSE (test)	0.1153	0.1012	0.1391	0.1186
Korelační koeficient (test)	0.1507	0.1614	0.2013	0.1712

Tabulka 14: Vliv hodnoty dropoutu na RMSE pro architekturu 8

Na architektuře 8 s růstem hodnoty dropoutu roste i hodnota RMSE na trénovací a validační sadě. Pro parametry 0, 0.05 a 0.1 jsou rozdíly mezi hodnotami RMSE na trénovací a validační sadě 0.13, 0.13 a 0.15.

Z průměrných výsledků pro předvedené architektury lze vypozorovat, že hodnota RMSE pro validační sadu se pohybovala v rozmezí 0.15 – 0.29. Průměrné hodnoty korelačního koeficientu se pohybovaly v rozmezí 0.1 – 0.28.

6.2 Porovnání chybových funkcí

Následující pokusy porovnávající chybové funkce byly provedeny na architektuře 8, použita byla nefiltrovaná data. Regularizace dropoutem nebyla použita. Počet epoch činil 200.

Tabulka 15 ukazuje vliv jednotlivých chybových funkcí:

	RMSE (train)	RMSE (cross)	RMSE (test)	Korelační koeficient (test)
MAE	0.0127	0.1638	0.2145	0.2988
MSE	0.0152	0.1670	0.2646	0.2469
Correlation loss	0.0313	0.7177	0.7448	0.8264
MAE^3	0.0251	0.1438	0.2108	0.2443
MAE^5	0.0927	0.2369	0.234	0.1739
MAE Reg Corr 1	0.2202	0.2783	0.193	0.3411
MAE Reg Corr 2	0.1237	0.1456	0.1802	0.2955

Tabulka 15: Vliv chybové funkce na RMSE

Z hlediska hodnoty RMSE lze za nejlepší funkci považovat funkce MAE^3 a MAE Reg Corr 2. Pro funkci Correlation loss byly hodnoty RMSE vypočteny po přepočtu výsledků modelu lineární regresí s koeficienty stanovenými na trénovací sadě.

6.3 Porovnávání inicializací vah

V následujících pokusech jsou srovnány různé inicializace vah. Použita byla architektura 8 a chybová funkce MAE. Data nebyla filtrována a počet epoch činil 200.

Tabulka 16 až Tabulka 18 ukazují výsledky jednotlivých inicializací vah:

6.3.1 Výsledky pro I1

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.015	0.0202	0.0289	0.0214
RMSE (cross)	0.2008	0.143	0.186	0.1766
RMSE (test)	0.1911	0.2384	0.1806	0.2034
Korelační koeficient (test)	0.3285	0.356	0.3067	0.3304

Tabulka 16: Vliv hodnoty dropoutu na RMSE pro inicializaci vah I1

6.3.2 Výsledky pro I2

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0123	0.0256	0.048	0.0286
RMSE (cross)	0.1374	0.1872	0.1691	0.1646
RMSE (test)	0.1912	0.1547	0.1653	0.1704
Korelační koeficient (test)	0.3306	0.3383	0.3531	0.3406

Tabulka 17: Vliv hodnoty dropoutu na RMSE pro inicializaci vah I2

6.3.3 Výsledky pro I3

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0118	0.0269	0.0313	0.0233
RMSE (cross)	0.1296	0.147	0.1329	0.1365
RMSE (test)	0.2305	0.1747	0.1615	0.1889
Korelační koeficient (test)	0.3397	0.3334	0.3439	0.3390

Tabulka 18: Vliv hodnoty dropoutu na RMSE pro inicializaci vah I3

Z výsledků pro jednotlivé inicializace nelze vypozorovat jednoznačný trend ve změnách hodnoty RMSE v závislosti na změnách hodnoty regularizace dropoutem.

Inicializace I3 má však výrazně menší průměrnou RMSE na validační sadě v porovnání s inicializacemi I1 a I2.

6.4 Porovnání vážení chybové funkce

Následující experimenty proběhly za použití architektury 8, chybové funkce MAE a počet epoch činil 200. Data nebyla filtrována. Byla použita inicializace vah I1.
Tabulka 19 až Tabulka 20 porovnávají hodnoty RMSE pro pokusy bez vážení chybové funkce a pokusy s vážením:

6.4.1 Výsledky bez vážení

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.015	0.0202	0.0289	0.0214
RMSE (cross)	0.2008	0.143	0.186	0.1766
RMSE (test)	0.1911	0.2384	0.1806	0.2034
Korelační koeficient (test)	0.3285	0.356	0.3067	0.3304

Tabulka 19: Vliv hodnoty dropoutu na RMSE pro experiment bez vážení chybové funkce

6.4.2 Výsledky s vážením

	1. Experiment	2. Experiment	3. Experiment	Průměr
Dropout	0	0.05	0.1	
RMSE (train)	0.0157	0.0209	0.0302	0.0223
RMSE (cross)	0.1883	0.171	0.1625	0.1739
RMSE (test)	0.2169	0.2102	0.2164	0.2145
Korelační koeficient (test)	0.3192	0.2961	0.3241	0.3131

Tabulka 20: Vliv hodnoty dropoutu na RMSE pro experiment s vážením chybové funkce

Nelze vypozorovat výrazný vliv použitého vážení na hodnotu RMSE pro jednotlivé sady.

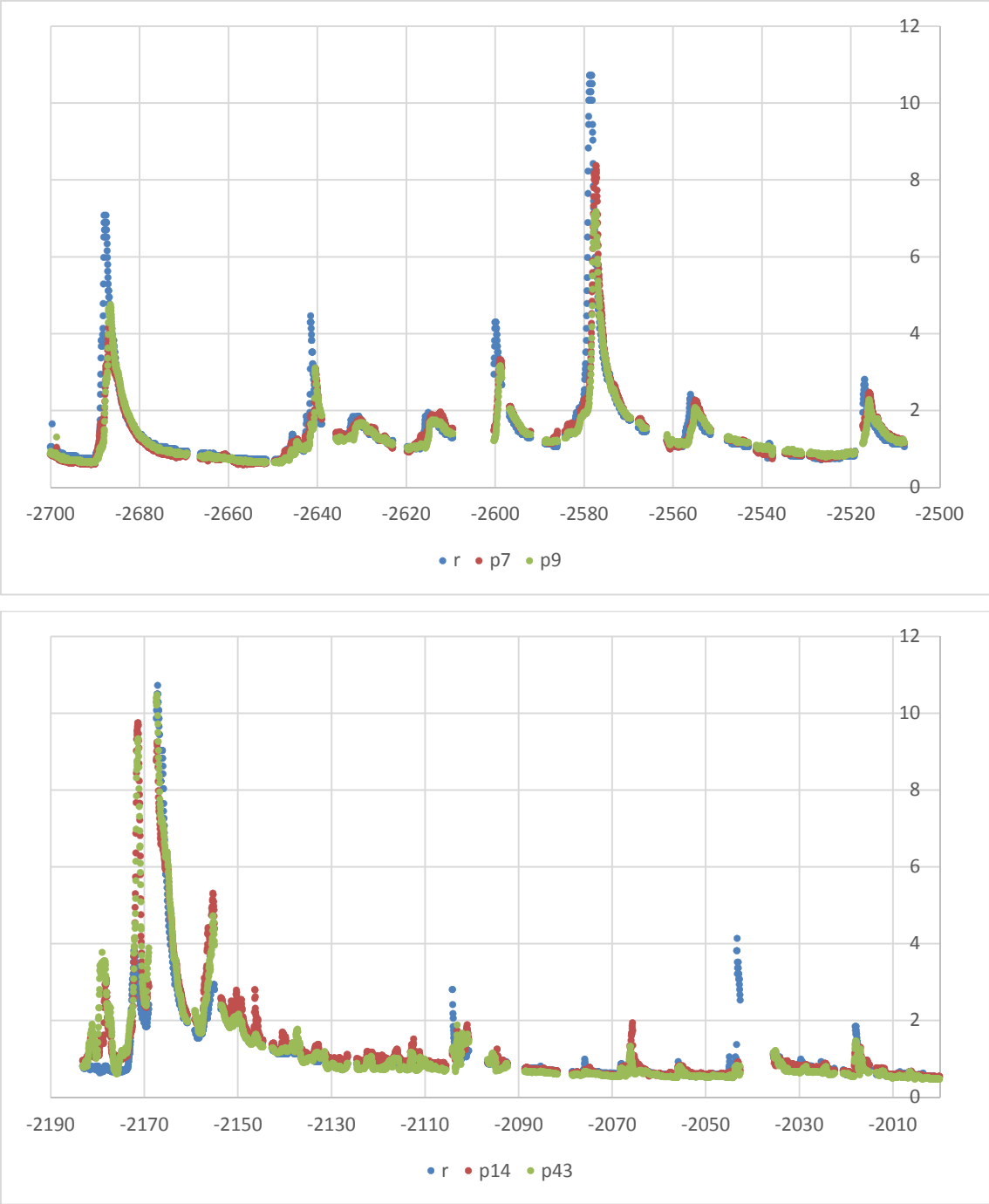
6.5 Vyhodnocení grafů vybraných experimentů

Pro hodnocení výsledků bylo použito také vizuálního porovnání grafu predikovaného a skutečného průtoku. Na grafech je možné pozorovat, zda hodnota predikce dosáhla úrovně skutečného průtoku, případně zda byla úroveň podhodnocena nebo nadhodnocena. Zároveň je možné hodnotit, zda nárůst či snížení průtoku nastává u predikovaného průběhu ve stejném okamžiku jako u skutečného průběhu, případně zda nedošlo ke zpoždění. Obrázek 26 a Obrázek 27 ukazují výsledky čtyř vybraných experimentů s kódovými označeními p7, p9, p14 a p43, které z provedených experimentů dosahovaly nejlepších kombinací RMSE a korelačních koeficientů. Na vodorovných osách v grafech jsou vyneseny počty dní uplynulé od referenčního dne na konci roku 2017, záporné hodnoty znamenají minulost.

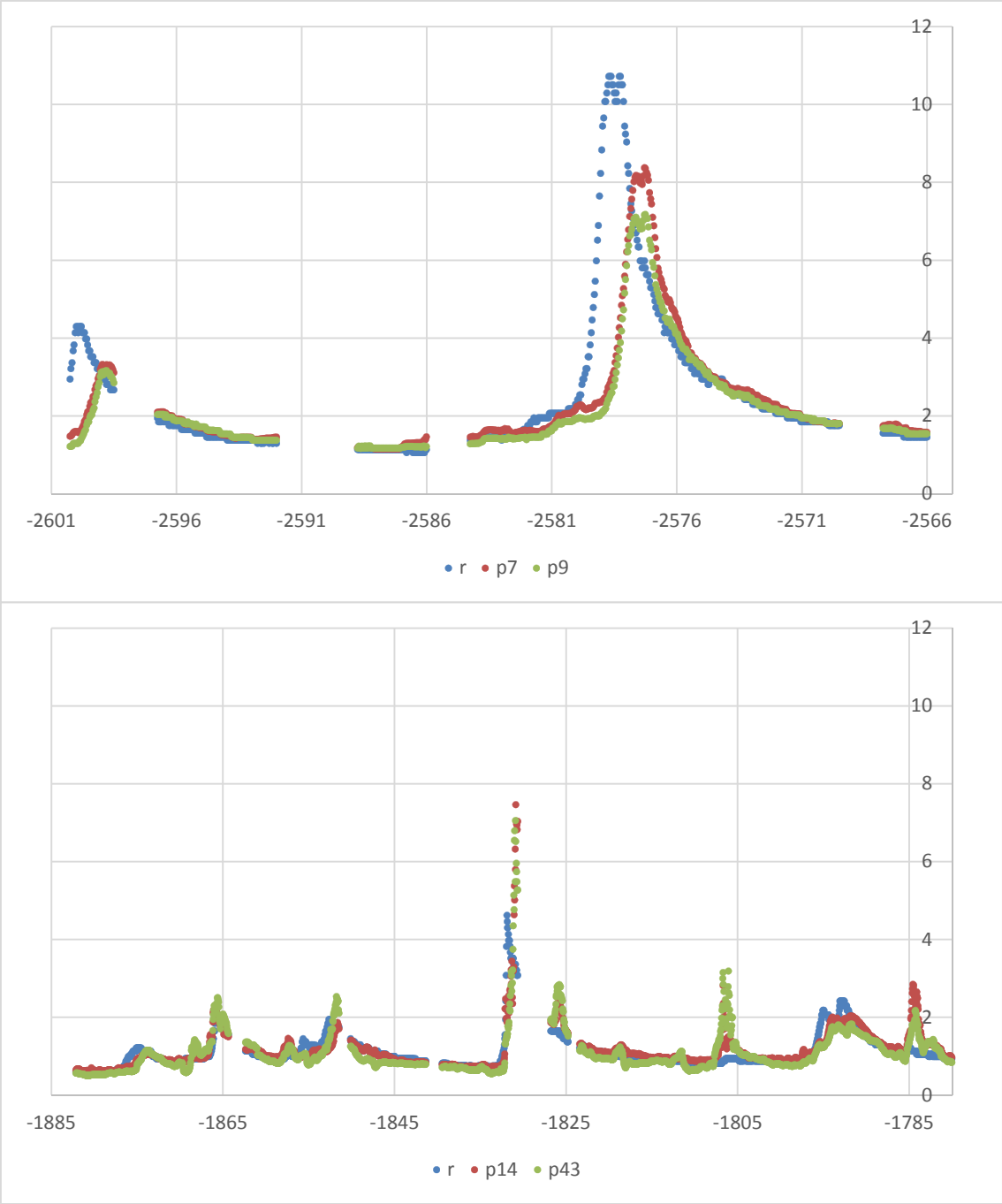
V následující tabulce jsou Nash-Sutcliffovy koeficienty pro vybrané experimenty:

Experiment	Nash-Sutcliffe E
P7	0,68
P9	0,63
P14	0,66
P43	0,66

Tabulka 21 - Nash-Sutcliffe koeficienty pro vybrané experimenty



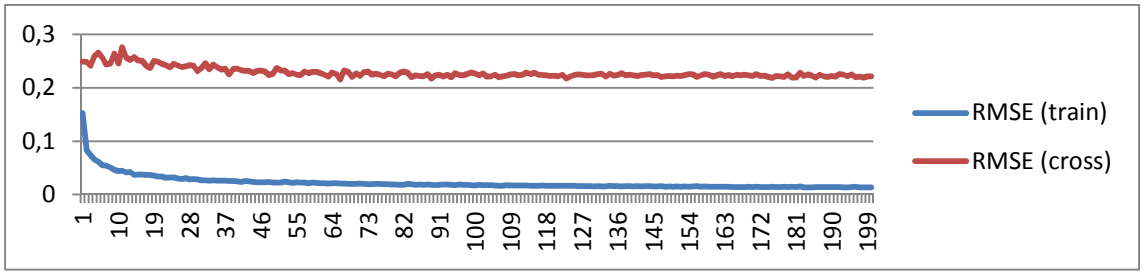
Obrázek 26. Srovnání naměřených a predikovaných hodnot hodinových průtoků o 24 hodin později na testovací sadě. Modré značky (r) představují skutečný naměřený průtok, červené a zelené značky odpovídají predikovaným hodnotám v jednotlivých vybraných experimentech.



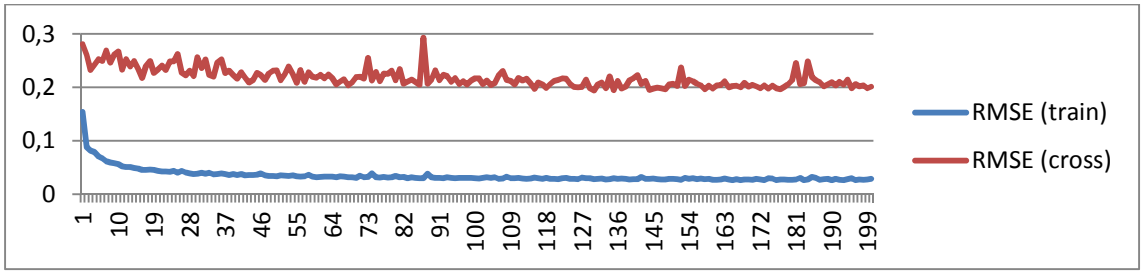
Obrázek 27. Detaily výsledků na vybraných experimentech

7 Shrnutí výsledků

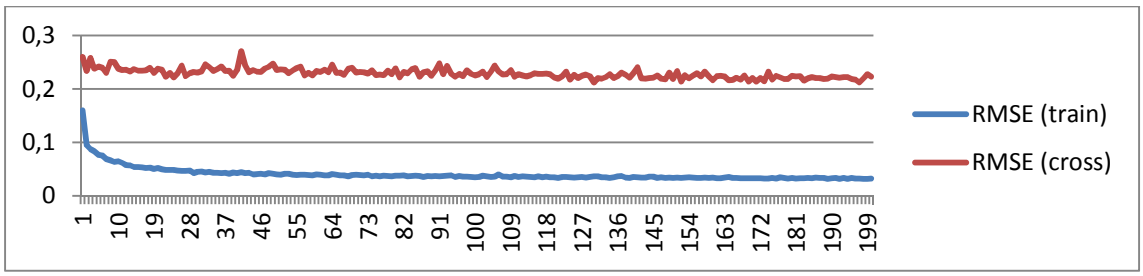
Po pozorném prozkoumání je nutno dojít k závěru, že jedinou proměnnou, která měla výrazný vliv na hodnotu RMSE, byla zvolená architektura. Dále bylo možné vypozorovat výrazný rozdíl mezi hodnotou RMSE na validační sadě při inicializaci I3 a ostatními způsoby inicializace. Rozdíly RMSE při porovnávání ostatních proměnných byly minimální a dají se přičíst náhodnému rozdělení trénovacích a validačních sad. S růstem parametru dropoutu rostla i RMSE na trénovací sadě, na chybu na validační sadě to však nemělo jednoznačně pozitivní vliv, jak je patrné z následujících grafů (Obrázek 28 až Obrázek 30).



Obrázek 28: Průběžná změna RMSE bez použití regularizace dropoutem v průběhu učení

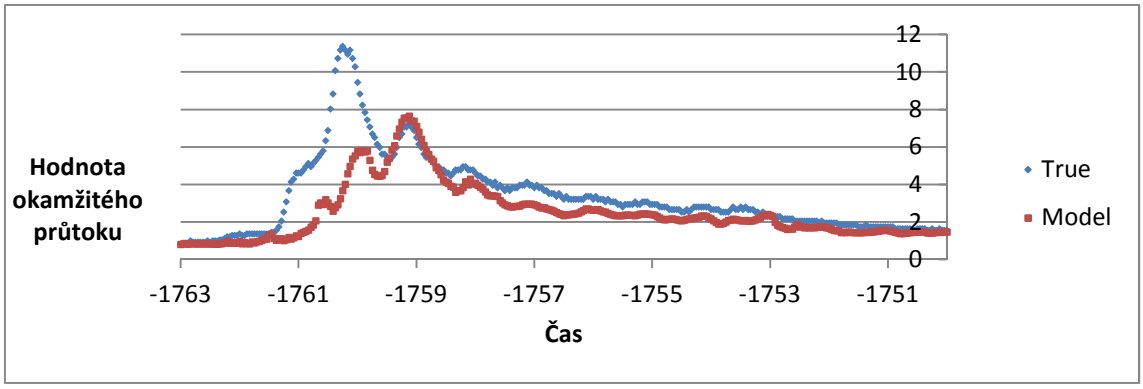


Obrázek 29: Průběžná změna RMSE s použitím regularizace dropoutech (parametr 0.05) v průběhu učení



Obrázek 30: Průběžná změna RMSE s použitím regularizace dropoutech (parametr 0.1) v průběhu učení

Po prozkoumání grafů předpovědí jednotlivých modelů bylo vidět systematické opožďování a podhodnocování předpovědi zejména při nástupu povodňové epizody, viz Obrázek 31.



Obrázek 31: Zpoždění a podhodnocení předpovědi průtoku modelem

8 Závěry a doporučení

Po prozkoumání výsledků šetření je nutné konstatovat, že se neprokázala schopnost dopředných neuronových sítí předpovídat s přesností potřebnou pro praktické využití. Tato přesnost je obzvláště malá při nástupech epizod povodní, jejichž předpověď je hlavním praktickým využitím modelů v této oblasti. Důvodem může být nízká kvalita dat, která byla zapříčiněna chybějícími údaji ze senzorů. Závažnějším problémem se však jeví velmi malé zastoupení povodní a s tím související rozličný charakter jednotlivých povodní komplikující zachycení funkční závislosti. Nerovnoměrnost v datech se nepodařila dostatečně vykompenzovat vážením chybové funkce, použitím netradičních chybových funkcí ani filtrací dat.

Jednou z velkých komplikací bylo stanovení vhodné metriky pro samotné porovnávání. Nízká hodnota RMSE na testovací sadě totiž v závislosti na náhodném rozdělení do příslušných sad ještě nemusí signalizovat kvalitní předpovědi. Doplnění dat by mohlo výrazně posunout kvalitu modelu.

Zlepšení předpovědi by mohlo být dosaženo i využitím rekurentních neuronových sítí (LSTM, GRU), které mohou být účinné, pokud existují závislosti mezi aktuálním stavem a dávnou minulostí. Tyto závislosti by nemusely být zachyceny dopřednou sítí. Ke zlepšení předpovědi by mohlo vést přeformulování problému na problém klasifikace stupně povodňové aktivity, stejně tak i jako využití technik učení bez učitele pro klasifikaci vstupních údajů a trénink samostatných sítí pro jednotlivé skupiny.

9 Seznam použité literatury

BANKO, Michele a Eric BRILL, 2001. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* [online]. Stroudsburg, PA, USA: Association for Computational Linguistics, s. 26–33 [vid. 2018-04-10]. ACL '01. Dostupné z: doi:10.3115/1073012.1073017

BASHEER, I. A a M HAJMEER, 2000. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods* [online]. **43**(1), Neural Computting in Micrbiology, 3–31. ISSN 0167-7012. Dostupné z: doi:10.1016/S0167-7012(00)00201-3

BENGIO, Y., P. SIMARD a P. FRASCONI, 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* [online]. **5**(2), 157–166. ISSN 1045-9227. Dostupné z: doi:10.1109/72.279181

DAWSON, CHRISTIAN W. a ROBERT WILBY, 1998. An artificial neural network approach to rainfall-runoff modelling. *Hydrological Sciences Journal* [online]. **43**(1), 47–66. ISSN 0262-6667. Dostupné z: doi:10.1080/02626669809492102

DEMIR-KAVUK, Ozgur, Mayumi KAMADA, Tatsuya AKUTSU a Ernst-Walter KNAPP, 2011. Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features. *BMC Bioinformatics* [online]. **12**(1), 412. ISSN 1471-2105. Dostupné z: doi:10.1186/1471-2105-12-412

DUCHI, John, Elad HAZAN a Yoram SINGER, 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*. **12**(Jul), 2121–2159.

FAUSETT, Laurene, 1994. *Fundamentals of neural networks: architectures, algorithms, and applications*. B.m.: Prentice-Hall, Inc. ISBN 0-13-334186-0.

HOCHREITER, Sepp a Jürgen SCHMIDHUBER, 1997. Long short-term memory. *Neural computation*. **9**(8), 1735–1780. ISSN 0899-7667.

CHOLLET, François a OTHERS, 2015. *Keras* [online]. B.m.: GitHub. Dostupné z: https://github.com/keras-team/keras

KINGMA, Diederik P. a Jimmy BA, 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* [online]. [vid. 2018-01-25]. Dostupné z: http://arxiv.org/abs/1412.6980

MCCULLOCH, Warren S. a Walter PITTS, 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* [online]. **5**(4), 115–133. ISSN 0007-4985, 1522-9602. Dostupné z: doi:10.1007/BF02478259

MCLNERNEY, John M, Karen G HAINES, Steve BIAFORE a Robert HECHT-NIELSEN, 1987. Can Backpropagation error surfaces have non-global minima? In: *Proc. of the IEEE-IJCNN89*. s. 11–627.

MINSKY, Marvin, Seymour A. PAPERT a Léon BOTTOU, 2017. *Perceptrons: An Introduction to Computational Geometry*. B.m.: MIT Press. ISBN 978-0-262-53477-2.

NASH, J. E. a J. V. SUTCLIFFE, 1970. River flow forecasting through conceptual models part I — A discussion of principles. *Journal of Hydrology* [online]. **10**(3), 282–290. ISSN 0022-1694. Dostupné z: doi:10.1016/0022-1694(70)90255-6

NOVIKOFF, Albert BJ, 1963. *On convergence proofs for perceptrons*. B.m.: STANFORD RESEARCH INST MENLO PARK CALIF.

OLAH, Christopher, 2015. *Understanding LSTM Networks -- colah's blog* [online] [vid. 2018-04-10]. Dostupné z: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

PAKKENBERG, Bente a Hans Jørgen G GUNDERSEN, 1997. Neocortical neuron number in humans: effect of sex and age. *Journal of Comparative Neurology*. **384**(2), 312–320. ISSN 1096-9861.

RUMELHART, David E., Geoffrey E. HINTON a Ronald J. WILLIAMS, 1986. Learning representations by back-propagating errors. *Nature* [online]. **323**(6088), 533. ISSN 1476-4687. Dostupné z: doi:10.1038/323533a0

SRIVASTAVA, Nitish, Geoffrey HINTON, Alex KRIZHEVSKY, Ilya SUTSKEVER a Ruslan SALAKHUTDINOV, 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*. **15**(1), 1929–1958. ISSN 1532-4435.

TIBSHIRANI, Robert, 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*. **58**(1), 267–288. ISSN 0035-9246.

TIELEMAN, Tijmen a Geoffrey HINTON, 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*. **4**(2), 26–31.

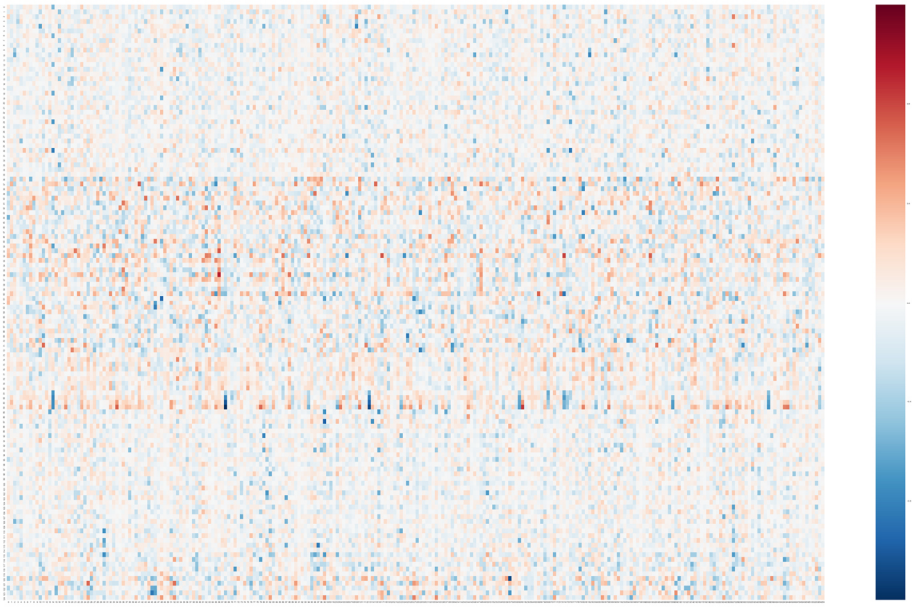
TORSE, DA, RR MAGGAVI a SA PUJARI, 2012. Nonlinear blind source separation for EEG signal pre-processing in brain-computer interface system for Epilepsy. *International Journal of Computer Applications*. **50**(14).

ZEILER, Matthew D, M RANZATO, Rajat MONGA, Min MAO, Kun YANG, Quoc Viet LE, Patrick NGUYEN, Alan SENIOR, Vincent VANHOUCKE, Jeffrey DEAN a OTHERS, 2013. On rectified linear units for speech processing. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. B.m.: IEEE, s. 3517–3521.

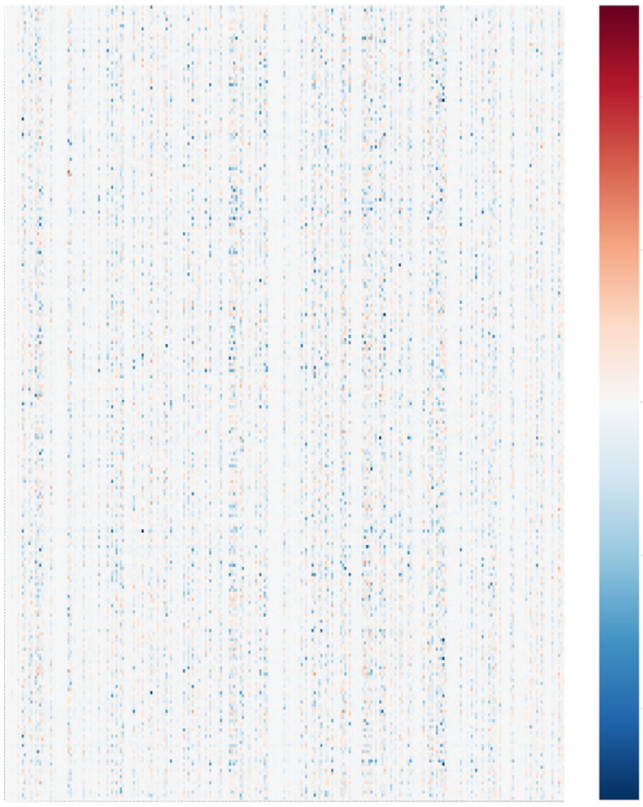
ZHANG, Guoqiang, B. EDDY PATUWO a Michael Y. HU, 1998. Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting* [online]. **14**(1), 35–62. ISSN 0169-2070. Dostupné z: doi:10.1016/S0169-2070(97)00044-7

10 Přílohy

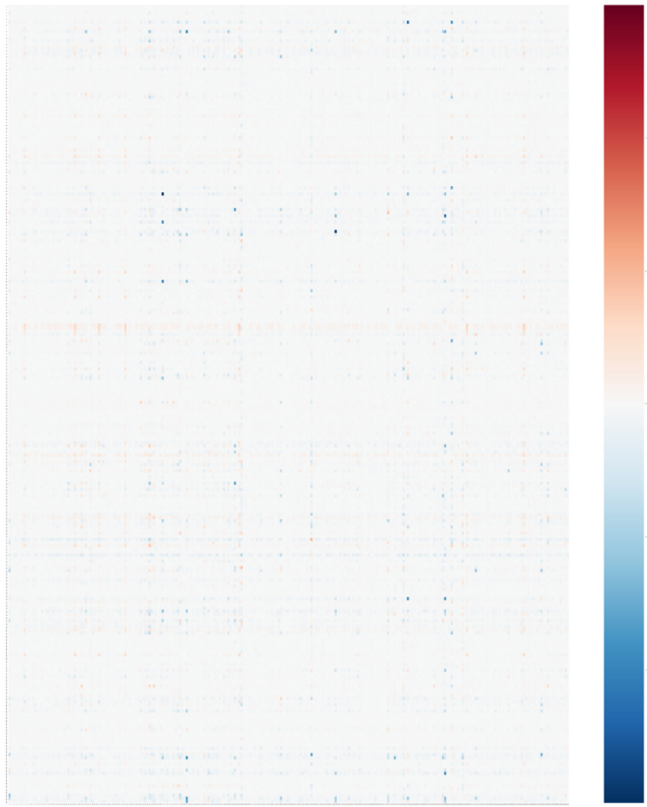
1) Ukázky heatmap zobrazující stav vah v neuronové síti



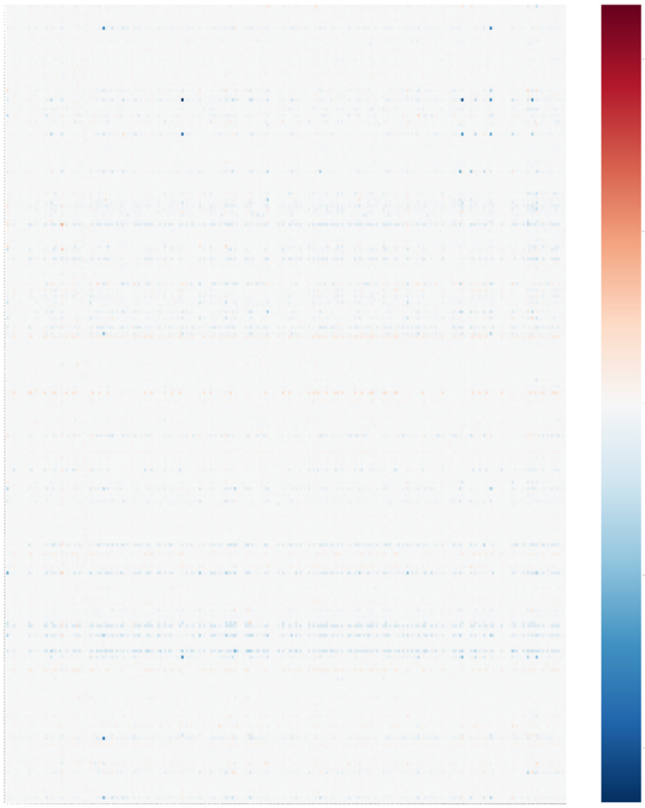
Obrázek 32 - Heatmapa zobrazující stav vah v první vrstvě po učení



Obrázek 33 - Heatmapa zobrazující stav vah ve druhé vrstvě po učení

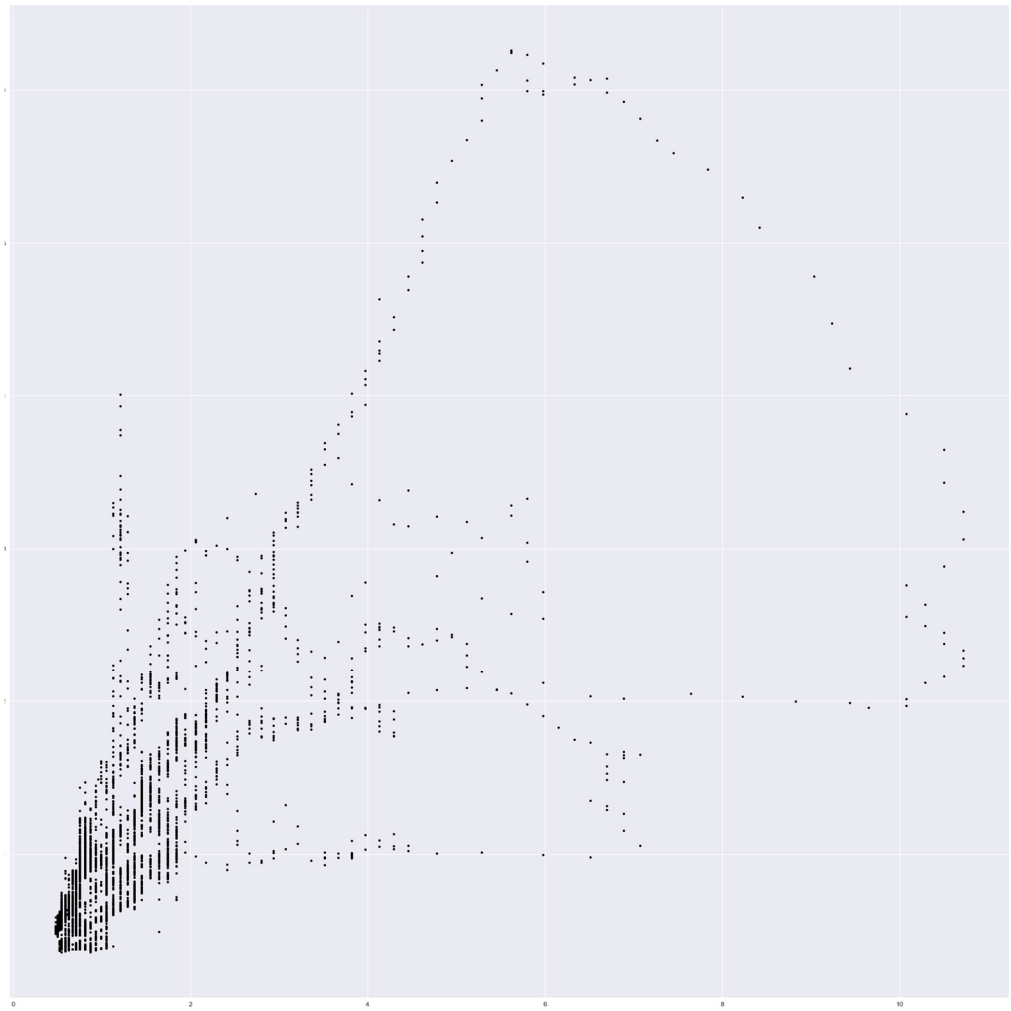


Obrázek 34 - Heatmapa zobrazující stav vah ve třetí vrstvě po učení

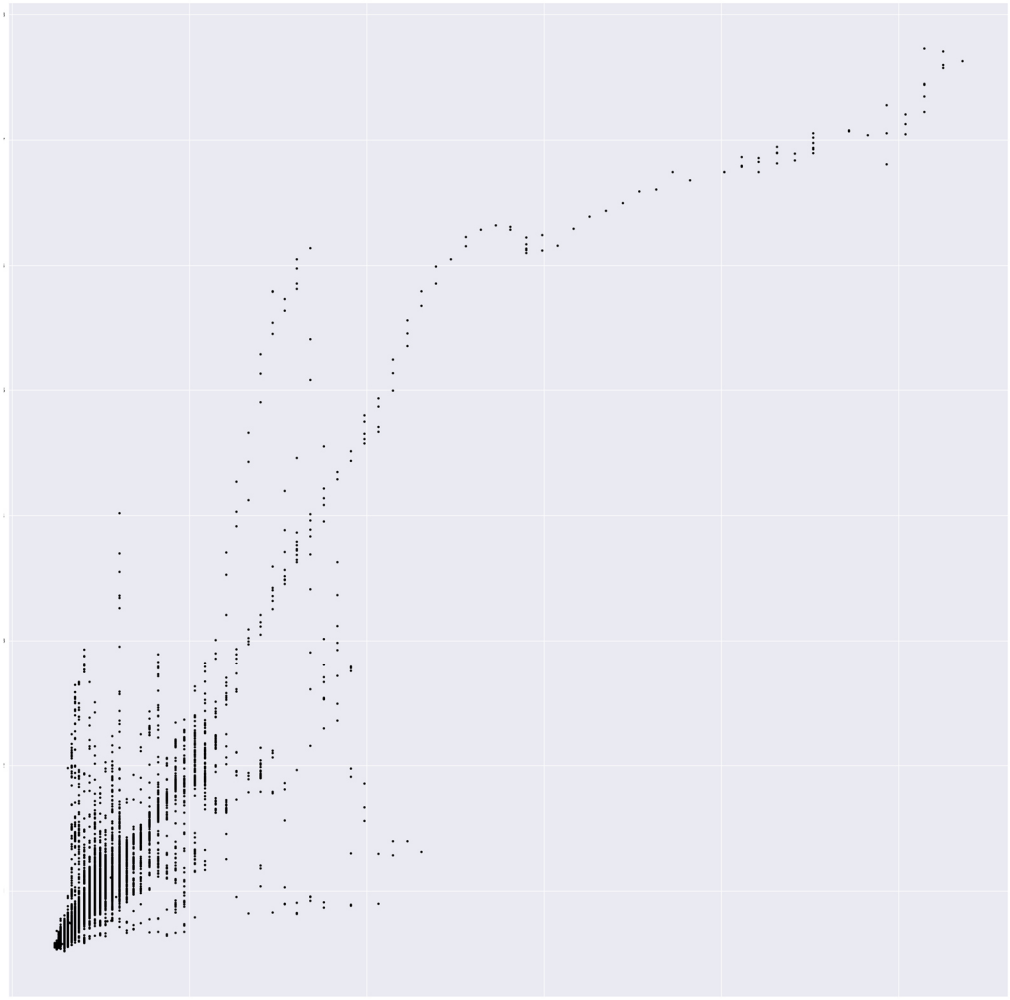


Obrázek 35 - Heatmapa zobrazující stav vah ve čtvrté vrstvě po učení

2) Ukázky scatterplotů znázorňující souvislosti mezi předpovídanými (svislá osa) a naměřenými (vodorovná osa) průtoky



Obrázek 36 - Znázornění souvislosti mezi předpovídanými a naměřenými průtoky na validační sadě



Obrázek 37- Znázornění souvislosti mezi předpovídanými a naměřenými průtoky na testovací sadě

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Konvička Martin	Boženy Němcové 925, Rožnov pod Radhoštěm	I1500380

TÉMA ČESKY:

Strojové učení

TÉMA ANGLICKY:

Machine Learning

VEDOUCÍ PRÁCE:

Mgr. Jan Vaněk, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce:
Prostudovat metody strojového učení, navrhnout systém pro predikci na základě hydrologických dat, systém implementovat a otestovat.

Postup prací:
Rešerše literatury
Návrh systému
Implementace
Testování

SEZNAM DOPORUČENÉ LITERATURY:

C. W. Dawson and R. Wilby, "An artificial neural network approach to rainfall-runoff modelling," Hydrological sciences journal, vol. 43, no. 1, pp. 4766, 1998.
F.-J. Chang and Y.-C. Chen, "A counterpropagation fuzzy-neural network modeling approach to real time streamflow prediction," Journal of Hydrology, vol. 245, no. 1, pp. 153164, May 2001.

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....