VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Modelování molekulových interakcí pomocí neuronových sítí

Modeling of Molecular Interactions Using Neural Networks

2018 David Vojtek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Vojtek**

Studijní program. B2647 Informační a komunikační technologie

Studijní obor· 2612R025 Informatika a výpočetní technika

Téma. Modelování molekulových ınterakcí pomocí neuronových sítí
Modeling of Molecular Interactions Using Neural Networks

Jazyk vypracování. čeština

Zásady pro vypracování.

Hlavním cílem bakalářské práce bude zhodnotit vhodnost použití neuronových sítí pro modelování molekulových interakcí s ohledem na prediktivní schopnost výsledného modelu.

Cíle bakalářské práce lze shrnout v těchto bodech.
1 Nastudujte základní pojmy týkající se neuronových sítí a pojmy kvantové chemie potřebné pro popis modelovaných interakcí.
2 Seznamte se s metodou největšího spádu a jejím použitím při tréninku neuronových sítí.
3. Prozkoumejte existující softwarové knihovny pro implementaci neuronových sítí.
4. S využitím vybraných knihoven implementujte model molekulových interakcí využívající neuronové sítě.
5. Zhodnoťte predikativní schopnosti vytvořeného modelu. V případě dostatečných prediktivní schopnosti vyhodnoťte časovou náročnost tréninku a predikce a navrhněte možná řešení pro zrychlení obou procesů.

Testovací data budou dodána vedoucím práce ve spolupráci s IT4I.

Seznam doporučené odborné literatury·

[1] Behler J., Parrinello M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces, Phys. Rev Lett. 98, 146401 Published 2 April 2007
[2] Skála, L Kvantová teorie molekul, Praha, Universita Karlova, 1995, ISBN 80-7184-007-6

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty

Vedoucí bakalářské práce: **Ing. Marek Běhálek, Ph.D.**

Konzultant bakalářské práce: Ing. Martin Beseda

Datum zadání. 01.09.2017

Datum odevzdání. 30.04.2018

doc. Ing. Jan Platoš, Ph.D
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 24. dubna 2018 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomáhali, zejména Ing. Martinovi Besedovi a Ing. Markovi Běhálkovi, Ph.D., protože bez nich by tato práce nevznikla. Dále bych chtěl poděkovat národnímu superpočítačovému centru IT4Innovations za umožnění provádět u nich výpočty v rámci projektu OPEN-11-4.

Abstrakt

Síly působící mezi částicemi při interakcích lze popsat pomocí nadploch potenciálové energie. Cílem této práce je vyhodnotit, zda by použití správné neuronové sítě dokázalo reprezentovat tyto nadplochy s dostatečnou přesností.

Pro ověření jsem používal dvou analytických modelů a to Lennard-Jonesova potenciálu a Morseho potenciálu. Nadplochy byly reprezentovány pomocí neuronových sítí s dopředným šířením signálu, které byly trénovány metodou zpětné propagace.

V rámci této práce se podařilo vytvořit takové neuronové sítě, které dokáží predikovat nadplochu potenciálové energie z relativně malého souboru vstupních dat. Výsledné neuronové sítě jsou také schopny přibližné extrapolace potenciálových nadploch pro velké vzdálenosti mezi částicemi, mimo rozsah trénovacích dat.

Výsledky získané v rámci této práce poslouží jako základ pro další výzkum, který se bude zabývat reprezentací větších systémů. Díky tomuto výzkumu bude možné efektivně reprezentovat nadplochy a predikovat hodnoty potenciálové energie i v oblastech, kde mají ab initio metody problémy s konvergencí nebo jsou příliš výpočetně náročné.

Klíčová slova: bakalářská práce, strojové učení, neuronová síť, molekulární interakce, Lennard-Jonesův potenciál, Morseho potenciál, síť s dopředným šířením signálu, zpětná propagace, Neuroph, potenciálová nadplocha

Abstract

The forces acting between the particles during their interactions can be described by potential energy surfaces. The aim of this work is to evaluate whether it is possible to represent these surfaces using neural networks with sufficient precision.

For verification, I used two analytical models, namely Lennard-Jones potential and Morse potential. Surfaces were represented by feed-forward neural networks that were trained using back-propagation method.

In this work, such neural networks have been created that can predict potential energy from a relatively small set of input data. The resulting neural networks are also able of an approximate extrapolation of potential surfaces in large distances between particles outside the range of training data.

The results obtained in this thesis will serve as the basis for further research, which will deal with representations of larger systems. Thanks to this research, it will be possible to efficiently represent surfaces and predict potential energy values even in domains where ab initio methods have convergence problems or where their are computationally too demanding.

Key Words: bachelor thesis, machine learning, neural network, molecular interaction, Lennard-Jones potential, Morse potential, feed-forward network, back-propapagation, Neuroph, potential energy surface

Obsah

Seznam použitých zkratek a symbolů

CV	–	Křížová validace
GC	–	Grandmother cell
MAPE	–	střední absolutní procentuální chyba
MSE	–	Střední kvadratická chyba
sMAPE	–	Symetrická střední absolutní procentuální chyba

Seznam obrázků

Seznam tabulek

1 Úvod

Hlavním cílem této práce bylo zjistit, zda by za pomoci neuronových sítí bylo možné reprezentovat nadplochy potenciálové energie. Částice mají tendenci dostat se do nejstabilnějšího stavu, což je vzdálenost, kdy je potenciálová energie mezi nimi nejmenší [1, 2].

Obvykle používané tzv. ab initio metody, se v některých případech ukazují jako velmi výpočetně náročné a výpočty tak mohou vyžadovat mnoho času. Navíc pro některé oblasti potenciálové nadplochy nejsou ab initio metody schopny, kvůli problémům s konvergencí, potenciálovou energii spočítat [3].

Díky reprezentaci nadploch pomocí neuronových sítí, by v budoucnosti stačilo spočítat jen malou část potenciálové nadplochy, která by sloužila k tréninku neuronové sítě a ta by pak byla schopnal predikovat ostatní části potenciálové nadplochy.

Určení potenciálových energií je potřebné v celé řadě vědních oborů. Jeden z aktuálně řešených problémů, kde by zrychlení výpočtu energie potenciálových nadploch bylo užitečné, je použití nízkoteplotního plazmatu při atmosférickém tlaku pro lékařské účely [4]. Toto plazma by například umožnilo přivést aktivní farmaceutické látky přímo do živých buněk, aniž by se poškodily[5].

V rámci této práce jsem se zabýval pouze dvouatomovými molekulami. Pro budoucí použití bude potřeba rozšířit počet atomů v molekule. To značně zvýší výpočetní náročnost, a tak bude potřeba také hledat metody efektivnějšího tréninku takových neuronových sítí a zabývat se možností víceúrovňové paralelizace [6, 7].

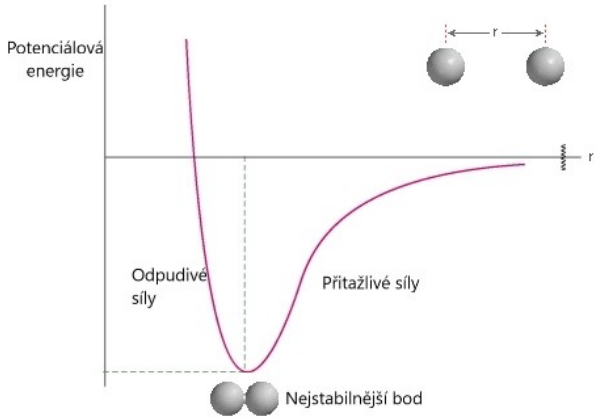
Jako vstup do neuronové sítě sloužila v této práci vzdálenost mezi atomy. Toto řešení umožnilo vyhnout se problémům se souřadnicovými systémy. U některých souřadnicových systémů, jako je například kartézský systém souřadnic, by otočení nebo posunutí částic způsobilo změny výstupů neuronové sítě díky odlišným vstupům, zatímco reálně zůstává vzdálenost mezi částicemi stejná a potenciálová energie se tak nemění [8].

Tato práce se ve druhé kapitole zabývá dvěma použitými analytickými modely, které sloužily k tréninku a testování vhodnosti neuronových sítí. Ve třetí kapitole pak přibližuje typy neuronových sítí, metody učení těchto sítí a také metody výpočtů chyb, které byly použity k vyhodnocení výsledků. Čtvrtá kapitola srovnává softwarové knihovny neuronových sítí a je zde také popsána zvolená knihovna i hardware, který sloužil k provádění výpočtů. Poslední, pátá kapitola se zabývá implementovanými neuronovými sítěmi, proběhlými měřeními a výsledky, které přinesly.

2 Modely molekul

Potenciál energie působící mezi atomy v molekulách lze popsat několika modely. Tyto modely popisují potenciální energii pomocí tzv. *potenciálové nadplochy*. V případě dvouatomových molekul pak nadplocha degeneruje v *potenciálovou křivku* (viz obrázek 1) a v této práci se tak budeme dále setkávat pouze s tímto případem.

Vzhledem k tomu, že popisujeme chování pouze dvou atomů, můžeme nahradit zadávaní vstupů jako polohových určení atomů pouze jejich vzdáleností, které je translačně i rotačně invariantní, což nám umožňuje vyhnout se problémům při posunu nebo rotaci systému.



Obrázek 1: Potenciálová křivka (zdroj:[9])

2.1 Lennard-Jonesův potenciál

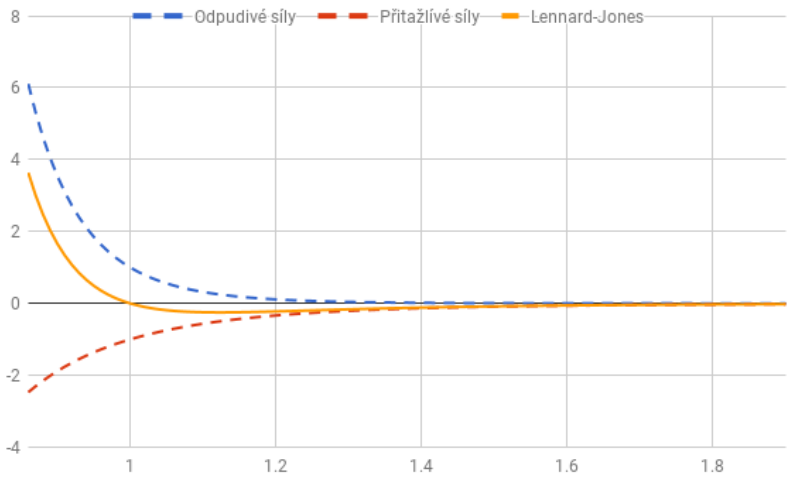
Lennard-Jonesův potenciál je zjednodušený model interakce párů atomů nebo molekul. Mezi molekulami působí zároveň přitažlivé a odpudivé síly. Přitažlivé síly jsou mezi dipóly molekul a odpudivé síly nastanou, když se elektronové mraky obou molekul připlíží natolik aby se odpuzovaly. Tyto síly udávají potenciální energii těchto dvou molekul. Lennard-Jonesova rovnice nám udává vývoj této potenciální energie v závislosti na měnící se vzdálenosti mezi molekulami.

$$V_{LJ} = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] = \varepsilon \left[\left(\frac{r_m}{r} \right)^{12} - 2 \left(\frac{r_m}{r} \right)^{6} \right] \tag{1}$$

ε nám udává hloubku potenciálové jámy, σ je konstanta označující vzdálenost, při které je potenciální energie mezi částicemi 0. r pak udává vzdálenost mezi molekulami a $r_m = 2^{1/6}\sigma$ je vzdálenost, kdy je potenciální energie nejmenší, tj. kdy nabývá hodnoty $-\epsilon$.

Přitažlivé síly odpovídají energiím pro vzdálenosti větší než r_m a odpudivé síly pak vzdálenostem menším, než r_m. Molekuly mají tendenci dostat se do vzdálenosti, kde je jejich potenciální energie nejmenší, podobně jako kulička kutálející se ze svahu. Přitažlivé síly odpovídají $-\left(\frac{\sigma}{r}\right)^6$ a jsou zastoupeny mimo jiné *Londonovou disperzí*, která je způsobena silami vyplývajícími z kvantově indukované okamžité polarizace multipólů molekul. Odpudivé síly reprezentované $\left(\frac{\sigma}{r}\right)^{12}$ jsou pak

zastoupeny *Pauliho repulzí*, která je způsobena překrývajícími se orbitaly elektronů. Tyto síly se rychle zmenšují se zvětšující vzdálenosti mezi atomy jak je zobrazeno na obrázku 2.



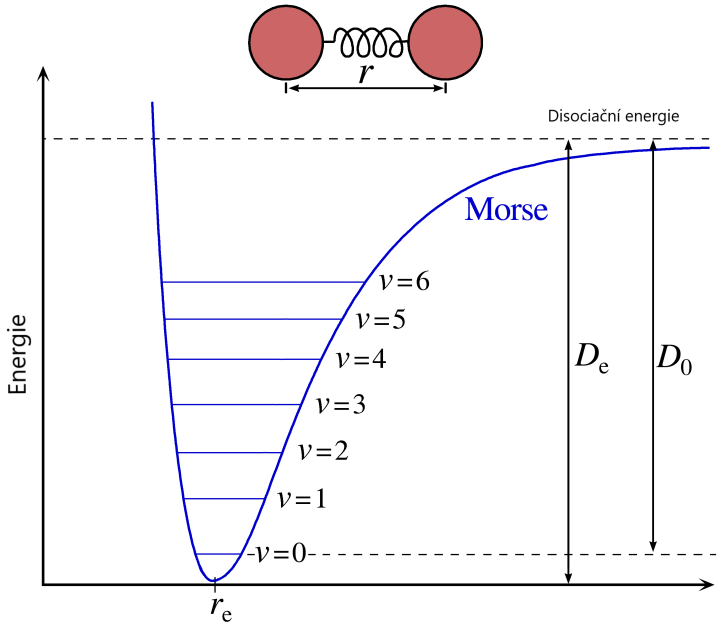
Obrázek 2: Lennard-Jones a rozložení sil

2.2 Morseho potenciál

Morseho potenciál, popsaný rovnicí (2), je zjednodušený model potenciálové energie vibrací dvouatomových molekul. Tento model reprezentuje vibrace molekul lépe, než *harmonický oscilátor* [10].

$$V_{(r)} = D_e \left(1 - e^{-a(r-r_e)}\right)^2 \tag{2}$$

Zde r je vzdálenost mezi atomy v molekule, r_e je vzdálenost rovnovážné vazby - vzdálenost, kde je potenciál nejmenší. D_e je hloubka potenciálové jámy. a pak určuje šířku potenciálové jámy čím menší a tím širší. Morseho potenciál je znázorněn na obrázku 3.



Obrázek 3: Vizualizace Morseho potenciálu (Zdroj: [11])

Kromě Morseho potenciálu se používá *Morse/Long-range potenciál* představený roku 2009. Tento potenciál je přesnější vzhledem k reálným výsledkům zvláště při dlouhých vzdálenostech, kdy se potenciálová křivka blíží k asymptotě [12].

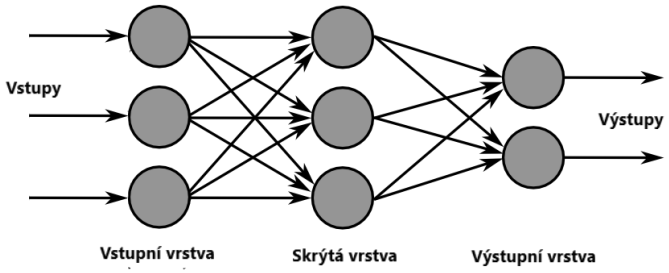
3 Neuronové sítě a metody hodnocení modelu

Neuronové sítě jsou inspirovány fungováním lidského mozku. Snaží se alespoň částečně simulovat některé jeho funkce, protože simulování celého mozku je kvůli jeho komplexitě mimo možnosti dnešních počítačů. V neuronových sítích jsou informace zpracovávány paralelně, a to pomocí celé neuronové sítě. Proto mají zpracovávaná data globální charakter.

Propojení našich simulovaných neuronů je podobné jako propojení biologických neuronů, a to pomocí vazeb. Význam vazby pro výsledek je dán její *vahou*. Takové vazby, které vedou ke správnému výsledku jsou při učení posilovány a ostatní oslabovány. Učení je nejdůležitější rys neuronových sítí, který odlišuje konvenční zpracování úloh na počítačích a používání neuronových sítí pro zpracování dat. Zatímco u běžných počítačových úloh byla nejdůležitější a nejtěžší tvorba algoritmu, který přemění vstupní data na data výstupní, u neuronových sítí tento proces zcela odpadá a je nahrazen učením pomocí již známého vzorku dat, tzv. trénovací množiny.

3.1 Typy sítí

Feed-forward síť dopředného šíření signálu je jedna ze základních typů neuronových sítí [13]. Tato síť obsahuje jednu vrstvu vstupních neuronů, jednu vrstvu výstupních neuronů a libovolný počet tzv. *skrytých vrstev* neuronů mezi nimi. Každý neuron má vazby na všechny neurony předchozí a následující vrstvy. Jedná se o takzvané *úplné propojení neuronů*. Signály se v síti šíří pouze jedním směrem a to od vstupní vrstvy k výstupní.



Obrázek 4: Šíření signálu v síti dopředného šíření signálu

Pro každou síť jsou na začátku synaptické váhy mezi neurony nastaveny na malé náhodné hodnoty. Následuje proces učení sítě, kdy se tyto hodnoty upravují, aby bylo dosaženo požadovaného výsledku. Toto učení probíhá pomocí některého pravidla učení sítě, například Hebbova pravidla nebo metody zpětného šíření signálu, podrobněji popsaného později. Proces výpočtu v neuronové síti probíhá následovně:

1. Dojde k přivedení vstupních hodnot do neuronů vstupní vrstvy.

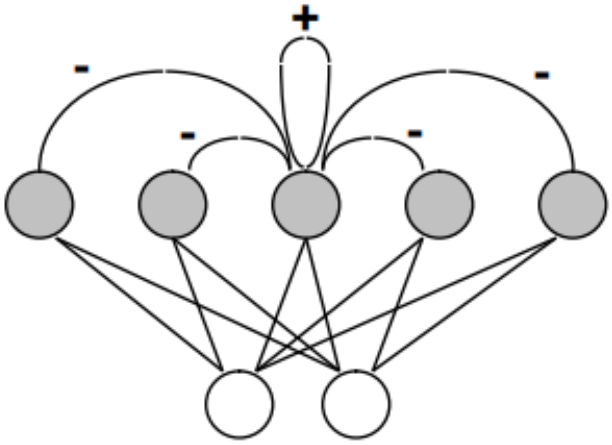
2. Tyto hodnoty jsou pomocí synoptických vazeb přivedeny k následující vrstvě a upraveny (zesíleny či zeslabeny) pomocí synoptických vah mezi neurony.

3. Každý neuron provede sumaci všech hodnot do něho přivedených a vypočítá výstupní hodnotu pomocí své aktivační funkce.

4. Tento proces probíhá přes všechny vnitřní vrstvy až k vrstvě výstupní, kde pak získáme výsledné hodnoty všech jejích neuronů.

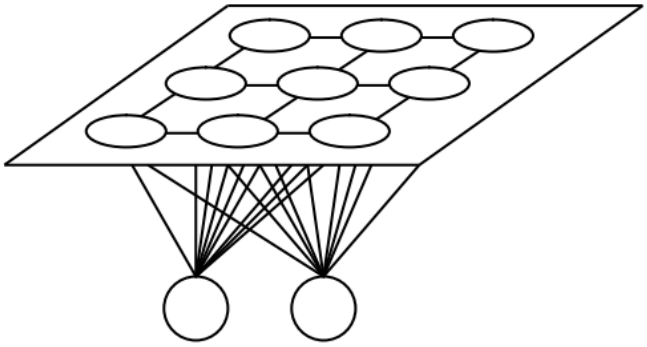
Samoorganizující se neuronové sítě jsou schopny samy zorganizovat data do kategorií, aniž by potřebovaly trénovací množinu dat. Toto se obzvláště hodí, když je potřeba najít spojitosti mezi daty, kterých si nejsme vědomi. Síť se snaží rozdělit data a jim přiřazené neurony tak, že čím „podobnější“ data jsou, tím blíže u sebe pak budou.

Tento druh sítě používá kompetitivní síť, která má pouze dvě vrstvy neuronů – vstupní a výstupní. Vstupní je úplně propojená s vrstvou výstupní. Každý výstupní neuron reprezentuje nějaký objekt ze vstupu. Takovým neuronům se říká *Grandmother cell (GC)*, protože tento neuron dokáže rozeznat nejen jeden objekt, ale celou třídu mu podobných objektů. Všechny výstupní neurony jsou pak ještě spojeny do jednoho z nich pomocí vazeb. Neuron je sám se sebou propojen pomocí sebeexcitujicí vazby. S ostatními neurony ve vrstvě je pak propojen pomocí inhibiční vazby. To vede k posílení neuronu, který měl na začátku největší hodnotu excitace a k oslabování ostatních. Tento neuron se nazývá *vítězný*. Postupně se vytvoří skupiny vstupních neuronů pro každou GC. Problém může nastat při nevhodné náhodné inicializaci vah, když jedna GC opakovaně vyhrává na úkor ostatních, které zůstávají nevyužité. Tento problém je možné řešit pomocí tzv. *svědomí neuronů*. V případě častých výher se vítězný neuron z procesu odpojí, aby dal šanci ostatním neuronům.



Obrázek 5: Samoorganizující se neuronové sítě (Zdroj: [14])

Kohonenovy mapy jsou neuronové sítě založené na principu samoorganizujících se neuronových sítí, jejichž smyslem je nalézt prostorovou reprezentaci datových struktur. Nejčastější forma Kohonenových map je ve formě dvoudimenzionální implementace, která je znázorněna na obrázku 6.



Obrázek 6: Dvoudimenzionální implementace Kohonenových map (Zdroj: [14])

Proces fungování Kohonenových map je založen na Kohonenově algoritmu, který je následující:

1. Inicializace sítě - nastavení vah, kde $w_{ij}(t)$ je váha vazby mezi vstupem i a neuronem j v čase t. Tato váha se nastaví na malou náhodnou hodnotu.

2. Poskytnutí vstupů

3. Stanovení vítěze kompetice - Vypočtení vzdálenosti d_j mezi vstupem a každým neuronem j. Tento výpočet se provádí pomocí rovnice (3).

$$d_j = \sum_{i=0}^{n-1} \left(x_i\left(t\right) - w_{ij}\left(t\right)\right)^2 \tag{3}$$

   kde x(t) je vstup v čase t

4. Výběr vítěze pomocí výběru minimální vzdálenosti

5. Úprava vah pro vítězný neuron a jeho sousedy. Nové váhy jsou spočítány pomocí rovnice (4).

$$w_{ij}\left(t+1\right) = w_{ij}\left(t\right) + \eta\left(t\right)\left(x_i\left(t\right) - w_{ij}\left(t\right)\right) \tag{4}$$

   kde $\eta \in \langle 0, 1\rangle$ je koeficient učení, který se časem snižuje

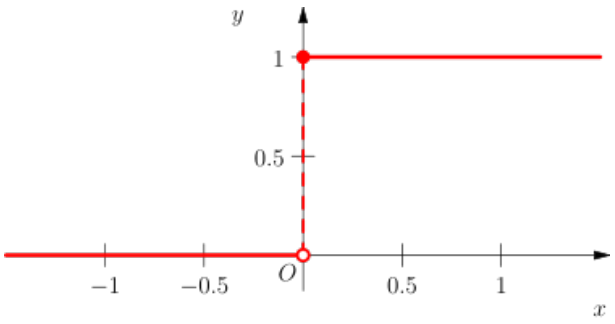
6. Návrat ke kroku 2

3.2 Aktivační funkce

Základní vlastnost rozlišující různé neurony je jejich aktivační funkce. Každý neuron přijme vážený součet vstupů a na tuto hodnotu aplikuje svou aktivační funkci. Její výstup je pak výstupem neuronu.

Jednotkový krok (Unit step) je po částech spojitá funkce, která má záporné vstupy hodnotu 0 a pro kladné hodnotu 1. Hodnota v bodě 0 se liší podle konkrétní definice, většinou však nabývá hodnot 0, 1, $\frac{1}{2}$ nebo není vůbec definována. Můžeme se setkat s variantami, které se liší jak v hodnotách, které funkce nabývá, tak i v rozdílném „bodu změny“.

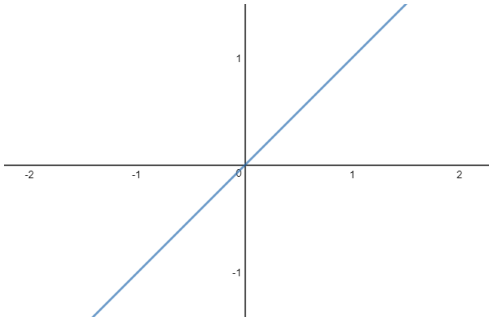
$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{5}$$



Obrázek 7: Graf jednotkového kroku

Lineární funkce popsaná rovnicí (6) je spojitá funkce, která má tvar přímky. Pomocí parametru a můžeme měnit sklon přímky a pomocí parametru b můžeme přímku posunout.

$$f(x) = ax + b \quad a, b \in \mathbb{R} \tag{6}$$



Obrázek 8: Graf lineární funkce

Sigmoidální funkce jsou množina různých funkcí. Jedna z nejpoužívanějších je *logistická funkce* popsaná rovnicí (7), někdy nazývaná také jako *soft step*. Je spojitou obdobou jednotkového kroku. Rozsah je od 0 do 1. Další funkce mívají rozsah obvykle od -1 do 1. Takovou funkcí je například *hyperbolický tangens* (8) nebo *softsign* (9). Další sigmoidální funkcí je *arkus tangens* (10) s rozsahem od $-\frac{\pi}{2}$ do $\frac{\pi}{2}$ nebo *ISRU* (11) s rozsahem od $-\frac{1}{\sqrt{a}}$ do $\frac{1}{\sqrt{a}}$.

$$f_{(x)} = \frac{1}{1+e^{-x}} \tag{7}$$

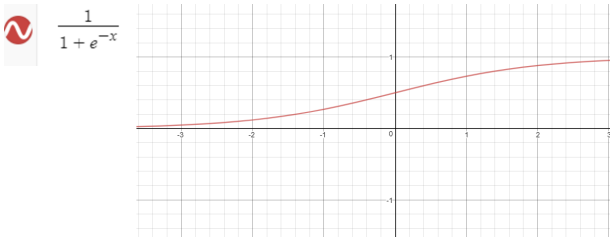
$$f_{(x)} = tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{8}$$

$$f_{(x)} = \frac{x}{1+|x|} \tag{9}$$

$$f_{(x)} = tan^{-1}(x) \tag{10}$$

$$f_{(x)} = \frac{x}{\sqrt{1+ax^2}} \tag{11}$$

$$\tag{12}$$

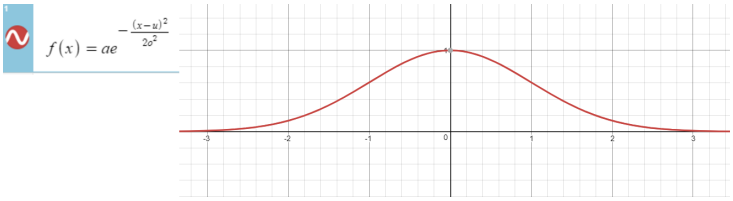


Obrázek 9: Graf logistické funkce

Obrázek 10: Skupina sigmoidálních funkcí

Gaussova funkce je další z často používaných aktivačních funkcí (13). Má 3 parametry a udává výšku vrcholu a musí být kladné, μ udává posunutí vrcholu do bodu 0 a σ udává šířku "základny".

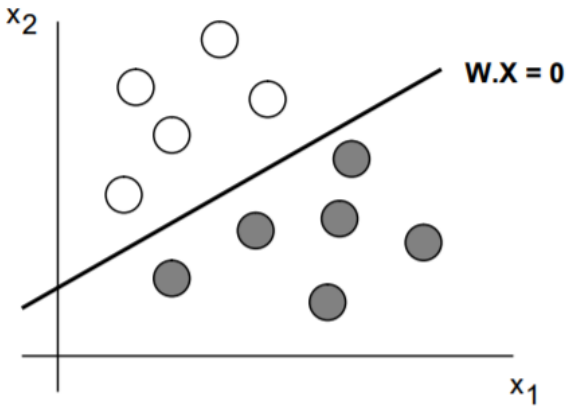
$$f_{(x)} = ae^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad a,\sigma \in \mathbb{R}+ \quad \mu \in \mathbb{R} \tag{13}$$



Obrázek 11: Gaussova funkce pro a=1, $\mu = 0$, $\sigma = 1$

3.3 Typy neuronů

Perceptron je jeden ze základních dnes používaných modelů neuronu. U tohoto modelu, stejně jako v případě neuronu první generace, musí pro aktivaci (excitaci) neuronu potenciál překonat jeho vnitřní prahovou hodnotu a tím dojde k excitaci neuronu na hodnotu 1 [14, 15]. V opačném případě je hodnota vždy 0. Díky této vlastnosti lze snadno rozdělit vstupy do dvou kategorií - lineárně separovaných množin. Jako aktivační funkce se používá jednotkový krok (5), nemusí však nabývat hodnot 0 a 1, lze zvolit i jiné dvě hodnoty. Rovněž „bod změny“ nemusí být vždy 0.



Obrázek 12: Rozdělení vstupů perceptronu do skupin (Zdroj: [14])

Lineární neuron je dalším z běžně používaných neuronů. Jeho aktivační funkcí je lineární funkce, popsaná rovnicí (6). Tento neuron tak nemá jen dva možné stavy, ale celou škálu. Vrací vážený součet vstupů a rozsahem není omezen. Díky těmto vlastnostem reaguje na malé rozdíly v datech citlivěji, než perceptron. Také se často používá ve vstupní a výstupní vrstvě.

Sigmoidální neuron používá jako aktivační funkci některou ze sigmoidálních funkcí, jejichž příklady jsou popsány rovnicemi (7)–(11). Jako v případě perceptronu je rozsah nejčastěji mezi 0 a 1, ale může být i mezi -1 a 1, podle zvolené funkce. Tyto funkce jsou spojitou obdobou jednotkového kroku popsaného rovnicí (5). Tento druh neuronů patří mezi nejpožívanější díky velkému množství použitelných aktivačních funkcí.

3.4 Metoda největšího spádu

Je metoda, která nám umožňuje najít lokální minimum funkce [16], tj. řeší úlohu danou rovnicí (14).

$$\min_{x \in \mathbb{R}^n} f(x) \tag{14}$$

Tato metoda opakuje sérii kroků dokud není výsledek dostatečně přesný, výsledek tedy průběžně zpřesňuje v tzv. *iteracích*. Algoritmus začíná určením počátečního bodu $x_{(0)}$ a další body získává tak, aby klesala hodnota *cenové funkce* $f(x)$. Využívá k tomu znalosti směru ve kterém klesá funkce nejrychleji, tj. směru největšího spádu. Ten je dán záporným gradientem funkce $f(x)$. V případě, že je funkce $f(x)$ kvadratická, tj. ve tvaru daném rovnicí (15), pak je směr největšího spádu, popsán rovnicí (16), kde $A \in \mathbb{R}^{n \times n}$ je symetrickou, pozitivně definitní maticí a $b, c \in \mathbb{R}^n$.

$$f(x) = \frac{1}{2} x^T A x - b^T x + c \tag{15}$$

$$-f'\left(x_{(i)}\right) = b - A x_{(i)} \tag{16}$$

Dále zavedeme *vektor vzdálenosti od řešení* $e_{(i)}$ daný rovnicí (17) a *residuum* $r_{(i)}$, které udává směr největšího spádu a je popsáno rovnicemi (16) a (18).

$$e_{(i)} = x_{(i)} - x \tag{17}$$

$$r_{(i)} = b - A x_{(i)} = -A e_{(i)} = -f'\left(x_{(i)}\right) \tag{18}$$

Po určení směru musíme určit nový bod x_{i+1}, ležící na této směrnici, viz. rovnice (19).

$$x_{(i+1)} = x_{(i)} + \alpha r_{(i)} \tag{19}$$

Vzdálenost obou bodů se dá spočítat jako směrová derivace (20). Pro její spočítaní si pomůžeme řetízkovým pravidlem daným rovnicí (21). Z něj můžeme vyvodit, že musíme zvolit α tak, aby $r_{(i)}$ a $f'\left(x_{(i+1)}\right)$ byly ortogonální.

$$\frac{d}{d\alpha} f\left(x_{(i+1)}\right) = 0 \tag{20}$$

$$\frac{d}{d\alpha} f\left(x_{(i+1)}\right) = f'\left(x_{(i+1)}\right)^T \frac{d}{d\alpha} x_{(i+1)} = f'\left(x_{(i+1)}\right)^T r_{(i)} \tag{21}$$

α je pak možné spočítat pomocí rovnice (22).

$$\alpha = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}} \tag{22}$$

S novým bodem celou sérií kroků zopakujeme dokud není splněno konvergenční kritérium (např. dostatečně malá vzdálenost $x_i - x_{i-1}$).

3.5 Učení neuronových sítí

Učení neuronových sítí je proces jehož cílem je nastavit synaptické váhy tak, aby celá síť dávala přesné výsledky. Učení může být s učitelem, kdy je předložen vzor, v podobě trénovací množiny, z které se neuronová síť učí. Při učení bez učitele není žádná trénovací množina a síť sama třídí vstupy do skupin.

Hebbovo pravidlo je jedna z metod pro získaní požadovaného výsledku neuronové sítě, kdy je třeba přizpůsobit váhy jednotlivých spojení mezi neurony[17]. Toto pravidlo je popsáno rovnicí (23). Učení začíná inicializací vah mezi neurony a jejich vnitřní prahovou hodnotu na náhodná malá čísla. Následně se podle předložených vstupů vypočítají výstupy. Pokud je neuron excitován správně, spoje, které tuto excitaci vyvolaly, se posilují, nebo liší-li se výsledek při excitaci od požadovaného výsledku pak se tyto spoje oslabují. Pokud není neuron excitován, váhy se nemění.

$$\Delta w_i(t) = \eta x_i(t) \cdot y_i(t) \tag{23}$$

kde $\Delta w_i(t)$ je změna váhy vazby pro krok t
η je kladný koeficient učení
$x_i(t)$ je vstupní hodnota do neuronu v kroku t
$y_i(t)$ je výstupní hodnota do neuronu v kroku t

Pravidlo učení perceptronu je metoda učení pro neuronové feed-forward sítě bez skryté vrstvy tvořené perceptronem [18]. Při tomto učení se porovnává skutečná výstupní hodnota s předpokládanou hodnotou. Liší-li se, musí dojít ke změně vah. Výstup lze pro každý neuron spočítat pomocí (24).

$$y = f(y_{vstup}) = \begin{cases} 1, & y_{vstup} > \theta \\ 0, & y_{vstup} < \theta \end{cases} \tag{24}$$

kde $\theta \in \mathbb{R}$ je vnitřní prahová hodnota perceptronu.

Aby bylo možné provést efektivní učení neuronové sítě je potřeba zvolit vhodnou trénovací množinu a metodu učení. Trénovací množina se skládá ze dvou objektů – množiny vstupů a množiny požadovaných výstupů. Úkolem metody je pak přizpůsobit váhu spojení mezi neurony, aby následně dokázala tato síť předvídat výstupy podle dat z trénovací množiny. Síť také musí mít schopnost generalizace (zobecnění), tak aby dokázala předvídat výstupy i z dat mimo trénovací množinu.

Backpropagation (*metoda zpětného šíření*) je jedna z metod učení vícevrstvé neuronové sítě pomocí trénovací množiny[19]. Jedná se o metodu největšího spádu popsanou v sekci 3.4 aplikovanou na problém minimalizace cenové, resp. *chybové funkce*. Cílem je optimalizovat váhy spojení tak, aby se síť naučila, jak správně mapovat vstupy na výstupy. Celý proces začíná dopřednou fází, stejně jako u feed-forward, kdy se na vzorku z trénovací množiny excituje celá síť až k výstupní vrstvě. Následně se provede výpočet celkové chyby sítě pomocí rovnice (25)

$$E = \frac{1}{2n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2, \tag{25}$$

kde n je počet neuronů ve výstupní vrstvě,
$\hat{Y}$ je požadovaná hodnota a
Y je predikovaná hodnota.

Následuje zpětné šíření signálu, kdy chceme zjistit, jak moc každá vazba zasáhla do celkové chyby. Spočítat to můžeme pomocí řetízkového pravidla rovnice (26). Jednotlivé části této rovnice můžeme vypočítat pomocí vztahů (27) a (28) což je derivace aktivační funkce neuronu. Tato vlastnost umožňuje použití zpětné propagace pouze s diferencovatelnými aktivačními funkcemi, jako jsou například logistické aktivační funkce popsané rovnicí (9).

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_i}, \tag{26}$$

kde y je hodnota excitace neuronu dána jeho aktivační funkcí a z je součet vah vstupů do neuronu.

$$\frac{\partial z}{\partial w_i} = x_i \tag{27}$$

$$\frac{\partial y}{\partial z} = y(1-y) \tag{28}$$

Hodnota $\frac{\partial E}{\partial y}$ se liší v závislosti na tom, zda se sledovaná vazba nachází mezi výstupním neuronem a skrytým neuronem nebo mezi skrytým neuronem a jiným skrytým, popř. vstupním neuronem. Rovnici (29) použijeme v prvním zmíněném případě. Pro vazbu nacházející se mezi skrytými vrstvami, popř. skrytou a vstupní vrstvou, použijeme k výpočtu rovnici (30).

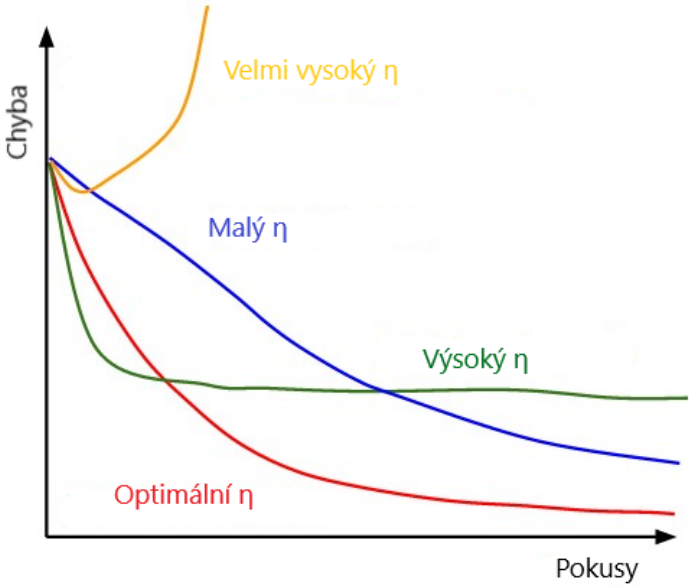
$$\frac{\partial E}{\partial y} = (y - o_j)_k \tag{29}$$

$$\frac{\partial E}{\partial y} = \sum_{i=1}^{m} \frac{\partial E}{\partial z^i}\frac{\partial z^i}{\partial y} = \sum_{i=1}^{m} \frac{\partial E}{\partial z^i} w^i \tag{30}$$

Po spočítání chyb všech vazeb v síti, můžeme upravit jejich hodnoty. Novou hodnotu vazby vypočítáme pomocí rovnice (31). Důležitým prvkem při učení neuronové sítě je *koeficient učení*

(learning rate) [20]. Ten nám určuje, jak moc budeme upravovat váhy jednotlivých vazeb. Příliš vysoký koeficient může způsobit, že se síť nebude zlepšovat nebo se bude i zhoršovat, jelikož díky velkým změnám vah po každém průchodu nepůjde zjistit, zda byly změny prospěšné. U příliš malého koeficientu sice máme větší šanci zjistit, že jsme se vydali „správným směrem“, ale celý proces bude výpočetně i časově náročnější, protože vyžaduje více kroků [21]. Je potřeba najít optimální hodnotu koeficientu učení tak, aby byla úprava vah dostatečně vysoká, abychom zbytečně neprodlužovali dobu trénování, ale zároveň dostatečně nízká, aby bylo možné rozpoznat správnou cestu.

$$w_i^+ = w_i - \eta \frac{\partial E}{\partial w_i} \tag{31}$$



Obrázek 13: Vývoj chyby neuronové sítě v závislosti na koeficientu učení (Zdroj: [20])

3.6 Přeučení

Při učení neuronové sítě hrozí také možnost *přeučení* (*overfitting*), kdy je síť příliš přizpůsobena trénovací množině a selhává při predikci jiných hodnot [22]. Toto může být způsobeno např. příliš malým počtem neuronů, malým množstvím trénovacích dat nebo přílišnou složitostí sítě (velký počet skrytých vrstev neuronů).

3.7 Vyhodnocení neuronových sítí

Pro zhodnocení, nakolik je neuronová síť správným modelem, je třeba zhodnotit dvě hlavní hlediska – souhlas s tréninkovými daty a prediktivní schopnost sítě. Existuje několik metod k výpočtu chyb, jichž se síť dopouští.

MSE (Mean Squared Error) - Střední kvadratická chyba počítá střední chybu nebo odchylku mezi vypočtenou hodnotou a předpokládanou hodnotou. MSE nám udává, že čím menší je jeho hodnota, tím blíže jsou si hodnoty vypočtené a předpokládané. Toho se využívá při trénování neuronových sítí, kdy hodnota MSE slouží jako kritérium zastavení učení. Rovnice MSE (32) nám říká, že hodnota MSE je součet všech druhých mocnin rozdílu mezi vypočtenou hodnotou (Y) a předpokládanou hodnotou ($\hat{Y}$) vydělený počtem hodnot.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{32}$$

sMAPE (Symetric mean absolute percentage error) - Symetrická střední absolutní procentuální chyba měří procentuální chybu mezi vypočtenou a předpokládanou hodnotou [23]. Podobně jako u MSE platí, že čím menší je hodnota chyby, tím je neuronová síť lepší. sMAPE je založena na střední absolutní procentuální chybě (MAPE) (33), kde n je počet hodnot, Y je vypočtená hodnota a ($\hat{Y}$) je předpokládaná hodnota. Hlavní nevýhodou MAPE je nesymetričnost, kdy u větších vypočtených hodnot jsou odchylky postihovány více než u malých vypočtených hodnot [24].

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|}\,[\%] \tag{33}$$

Problém s nesymetričností řeší právě sMAPE, které má více řešení. Základní verzí je (34), která má rozsah od 0% do 200%. Existuje i varianta (35) , která má rozsah od 0% do 100%.

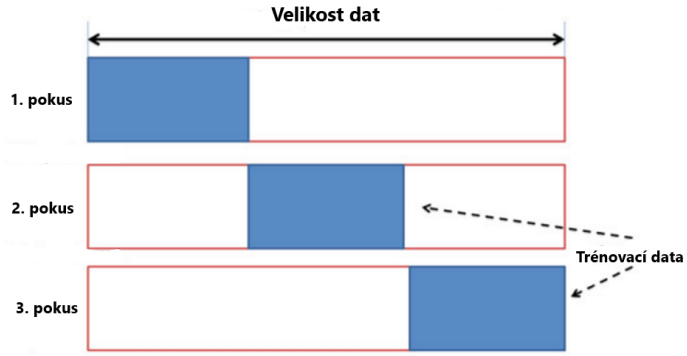
$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|\hat{Y}_i - Y_i\right|}{(Y_i + \hat{Y}_i)/2} \tag{34}$$

$$sMAPE = 2\frac{100\%}{n}\sum_{i=1}^{n}\frac{\left|\hat{Y}_i - Y_i\right|}{|Y_i| + \left|\hat{Y}_i\right|} \tag{35}$$

Problém u použité sMAPE spočívá v tom, že když se hodnota vypočtená nebo předpokládaná rovná 0, chyba se velmi přiblíží maximální velikosti (100% nebo 200%) a to může výrazně zkreslit celkový výsledek. Tento neduh se týká všech verzí, proto je vhodné kombinovat sMAPE i s jinými výpočty chyb.

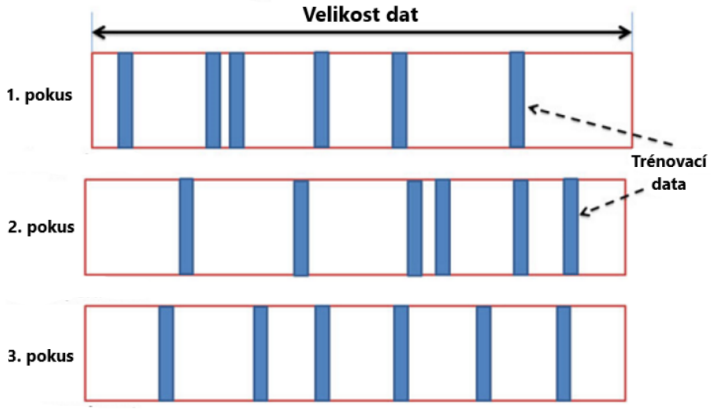
Křížová validace (Cross-validation - CV) je metoda posouzení prediktivních schopností neuronových sítí. Základní princip spočívá v rozdělení vstupní množiny dat na dvě podmnožiny - trénovací podmnožinu a testovací podmnožinu. Neuronová síť je naučena pomocí trénovací podmnožiny a následně otestována pomocí testovací podmnožiny. Tento proces se několikrát opakuje. Vyhodnocení pak probíhá pomocí některého řešení výpočtu chyb.

Existuje několik odlišných přístupů jak data rozdělit do dvou podmnožin. Jedním z nich je k-fold, kdy se vstupní množina dat rozdělí na k podmnožin, které jsou postupně použity jako trénovací množiny a ostatní pak jako testovací, jak je ukázáno na obrázku 14. Toto rozdělení nám zajistí, že každá část vstupní množiny je otestována na prediktivní schopnosti. Při rozumně zvoleném k nemusí být výpočet náročný, avšak pro k blížící se počtu prvků v množině můžou být potřebné výpočetní prostředky neadekvátně vysoké. Rovněž se předpokládá, že každá část bude mít prediktivní schopnosti.



Obrázek 14: Rozdělení dat do množin podle k-fold (Zdroj: [25])

Další možností použití CV je volba náhodného vzorku dat. Trénovací množina je náhodně vybrána jako určené procento prvků ze vstupní množiny. Nevybrané prvky tvoří testovací množinu, jak je ukázáno na obrázku 15.



Obrázek 15: Rozdělení dat do množin podle náhodného vzorku dat (Zdroj: [25])

Směrodatná odchylka běžně označována písmenem σ je metoda měření statistické variability. Směrodatná odchylka vypovídá o tom, nakolik se od sebe navzájem typicky liší jednotlivé

případy v souboru zkoumaných hodnot [26]. Čím menší je směrodatná odchylka tím méně se jednotlivé hodnoty od sebe liší. Směrodatná odchylka je dána rovnicí (36).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{36}$$

4 Výběr softwaru, použitý hardware a návrh vlastní sítě

4.1 Požadavky na software

Důležitou částí této práce byl výběr vhodné softwarové knihovny nebo frameworku pro tvorbu neuronových sítí. Knihovna měla být dostatečně jednoduchá, aby umožnila rychlou tvorbu a úpravu neuronových sítí bez rozsáhlých předchozích znalostí a zkušeností. Dále měla obsahovat API pro některý programovací jazyk s kterým jsem měl předchozí zkušenosti, nejlépe pro C# nebo Javu. Zcela optimálním řešením by pak byla volba open-source knihovny. Další požadavky nehrály při výběru tak důležitou roli.

Schopnost paralelizace nebyla vzhledem k relativně malému množství výpočtu důležitá. Při výpočtech pro více atomů, kde by byly výpočty mnohem komplikovanější, by možnost paralelizace hrála mnohem větší roli při určení platformy. Tehdy by bylo vhodné zvolit knihovnu vhodnou pro systém na kterém budou výpočtu probíhat. Například výběr knihovny s podporou technologie CUDA by byl důležitý v případě, že výpočty budou probíhat na GPU.

Ne všechny knihovny umožňovaly úplnou kontrolu nad tvorbou neuronových sítí. Například kontrola nad určením vazeb mezi neurony nebyla u všech knihoven přítomna.

4.2 Srovnání dostupného softwaru

Tensorflow je open-source framework pro tvorbu neuronových sítí vyvíjen společnosti Google v jazyce Python [27]. API je dostupné i pro řadu jiných programovací jazyků jako C++ nebo JAVA. Je používány některými velkými společnostmi například Nvidia, SAP nebo Intel. Se svými schopnosti a možnostmi však velice přesahuje potřeby řešeného problému. Množství nastavení jednotlivých aspektů tvorby, trénovaní a vyhodnocení neuronových síti, vzhledem k mým dosavadním malým zkušenostem s neuronovými sítěmi, by zbytečně komplikovalo a zdržovalo vývoj.

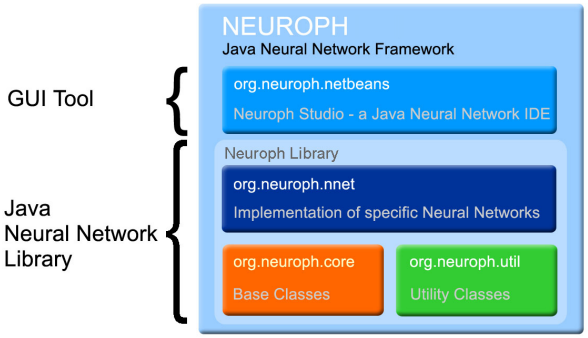
Caffe je další open-source framework pro tvorbu neuronových sítí napsaný v jazyce C++ [28]. Tento framework vznikl původně na univerzitě v americkém Berkeley pro akademické účely. Umožňuje pokročilé metody rozpoznávání obrazu a obrazovou segmentaci. Společností Facebook byl vyvinutý Caffe2, který dále rozšiřuje metody prací s obrazy. Jelikož při řešení této práce nepracuji s žádnými obrázky, byli lepší alternativy, které jsem zvolit.

Torch je open-source framework postavený na skriptovacím jazyku Lua [29]. Používá knihovny C/C++ a technologii CUDA pro výpočty na GPU. Důraz je kladen zejména na výpočty pomocí GPU. Podporu použití v iOS a Android systému.

Java Neural Network Framework Neuroph je open-source Java knihovna doplněné jednoduchým GUI umožňující tvorbu a trénování neuronových sítí v programech napsaných v

jazyce Java. Třídy v knihovně odpovídají základním konceptům neuronových sítí jako neuron, vrstva neuronů, synoptická vazba, aktivační funkce, pravidlo učení atd. V knihově jsou zakomponovány možnosti tvorby feed-forward sítí s učením pomocí zpětné propagace, Kohonenových map a Hopfieldových sítí. Všechny třídy mohou být rozšířeny a upraveny pro tvorbu vlastní sítě, nebo metody učení.

Právě tento framework jsem vybral jako nejlepší možnost. Celý framework je sestavený z několika na sobě nezávislých části, jak je ukázáno na obrázku 16, což umožňovalo během vývoje jednoduchý neuronových sítí použít pouze specifickou část celého frameworku, kterou jsem potřeboval.



Obrázek 16: Diagram Neuroph Frameworku (Zdroj: [30])

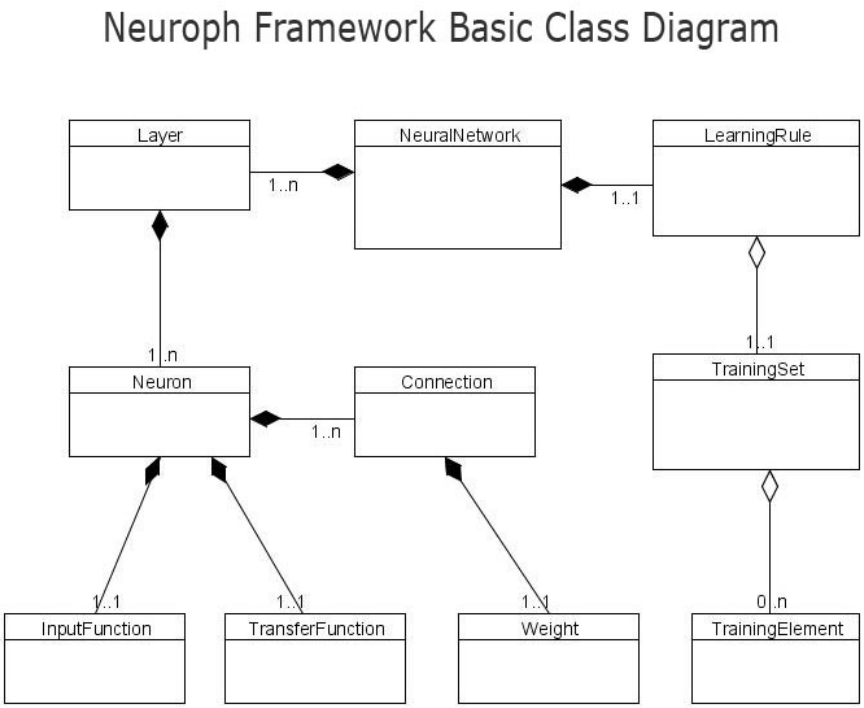
Nejdůležitější část je Java knihovna org.neuroph.core, která obsahu vše co jsem potřeboval pro tvorbu neuronové sítě.

4.3 Popis použitého hardwaru

Část výpočtů, z důvodu časově náročného tréninku neuronových sítí, probíhala na superpočítači Salomon národního superpočítačového centra IT4Innovations. Tento nejvýkonnější superpočítač v České republice a 87. nejvýkonnější superpočítač na světě [31] je složen z 1008 výpočetních uzlů každý obsahující dva procesory Intel Xeon E5-2680v3, 2.5 GHz s celkem 24 jádry a 128 GB RAM. 432 z těchto uzlů je navíc vybaven akcelerací pomocí dvou procesorů Intel Xeon Phi 7120P s 61 jádry. Celý superpočítač má teoretický výkon přes 2 Pflop/s. Superpočítač běží na operačním systému CentOS Linux.[32]

4.4 Návrh sítě

Při návrhu neuronové sítě typu feed-forward, která se obvykle používá u tohoto typu problémů, jsou nejdůležitějšími faktory počet neuronů ve skryté vrstvě a jejich aktivační funkce. Provedl jsem proto několik měření uvedených v tabulce 1, kdy jsem při různých konfiguracích neuronové sítě měřil čas a chyby MSE (32) a sMAPE (34) při reprezentaci Lennard-Jonesova potenciálu daného rovnicí (1). Cílem bylo najít takovou konfiguraci, která by měla nejlepší poměr mezi výslednou chybou a časem trénování. Postupně jsem zkoušel všechny základní aktivační funkce



Obrázek 17: Diagram tříd knihovny org.neuroph.core (Zdroj: [30])

dostupné v knihovně Neuroph. Úspěch jsem zaznamenal s logistickou funkcí (7) a Gaussovou aktivační funkcí (13). Ostatní aktivační funkce nevedly ani po 60 minutách k úspěšnému naučení sítě. Je sice možné, že by se síť s některými z těchto aktivačních funkcí dokázala naučit řešení, ale vzhledem k existenci alternativ s mnohem větším potenciálem, jsem se jimi dále nezabýval.

Počet neuronů ve skryté vrstvě jsem začínal na 50 a postupně jsem jejich počet navyšoval až do 350. U aktivačních funkcí, které se nejevily perspektivní jsem měření omezil na 50 a 200 neuronů.

Jako nejlepší jsem vyhodnotil konfiguraci s 200 neurony ve skryté vrstvě a logistickou aktivační funkcí, která měla nejmenší chybu sMAPE a to 3.87. S touto konfigurací jsem pak pracoval během dalších výpočtů.

Tabulka 1: Porovnání různých typů použitých neuronových sítí

Typ Sítě	Přenosová funkce	Počet Neuronů	Čas trénování	MSE	sMAPE
Feed-Forward	Logistická	50	0:12	4.82E-04	9.6
Feed-Forward	Logistická	100	0:23	4.72E-04	10.7
Feed-Forward	Logistická	150	0:25	4.8E-04	4.49
Feed-Forward	Logistická	200	0:32	4.62E-04	3.87
Feed-Forward	Logistická	250	1:10	4.54E-04	15.07
Feed-Forward	Logistická	300	3:59	4.52E-04	14.79
Feed-Forward	Logistická	350	9:54	4.73E-04	15.69
Feed-Forward	Gaussian	50	0:54	3.4E-04	9.72
Feed-Forward	Gaussian	100	2:56	6.1E-04	18.37
Feed-Forward	Gaussian	150	5:03	6.33E-04	18:35
Feed-Forward	Log	50/200	60:00+	-	-
Feed-Forward	Ramp	50/200	60:00+	-	-
Feed-Forward	Sgn	50/200	60:00+	-	-
Feed-Forward	Sin	50/200	60:00+	-	-
Feed-Forward	Step	50/200	60:00+	-	-
Feed-Forward	Tanh	50/200	60:00+	-	-
Feed-Forward	Trapezoid	50/200	60:00+	-	-

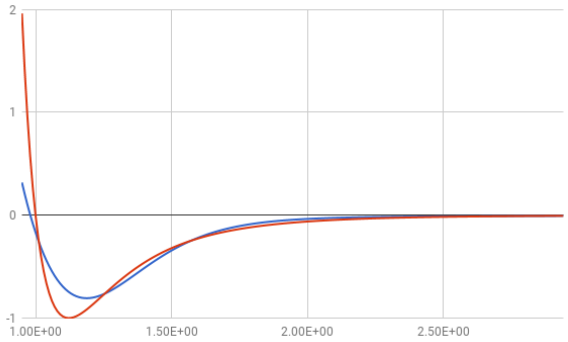
5 Výsledky a ověření modelů

5.1 Lennard-Jones potenciál

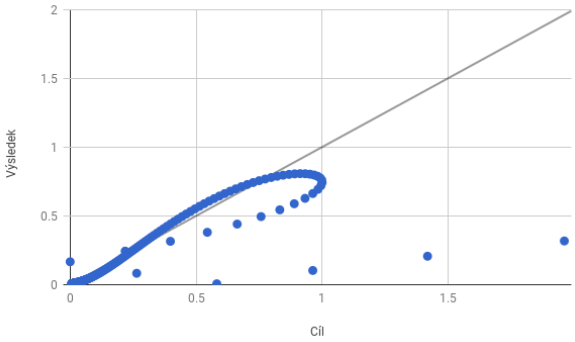
Po několika pokusech s různými druhy neuronových sítí jsem jako nejvhodnější variantu pro řešení zadaného problému zvolil feed-forward neuronovou síť s učením pomocí metody zpětného šíření, s jedním lineárním vstupním neuronem, jednou skrytou vrstvou obsahující 200 neuronů s logistickou aktivační funkcí (7) a jeden výstupní neuron s lineární aktivační funkcí (6).

Jako trénovací data posloužil vzorek 200 vstupů pro vzdálenosti molekul mezi 0,95 a 2,94Å. Výstup odpovídal Lennard-Jonesovy rovnici (1) pro $\varepsilon = 1$.

Už při jednom z prvních pokusů s trénováním sítě, kdy jako podmínka pro ukončení testování byla chyba MSE=0,01, byl výsledný graf vizuálně blízký s Lennard-Jonesovým grafem, jak je zobrazeno na grafu 18a. Odchylka však byla stále příliš velká. Chyby se vyskytovaly v oblasti potenciálové jámy, jak je zobrazeno na grafu 18b, kde je požadavek na přesnost největší.



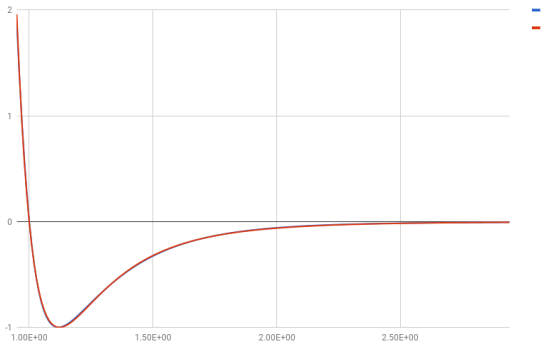
(a) Výstup neuronové sítě pro chybu 0,01



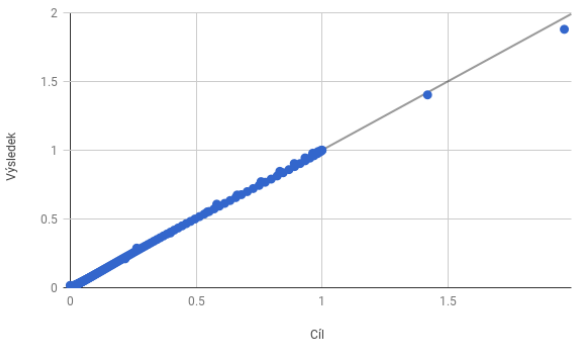
(b) Chyba neuronové sítě pro chybu 0,01

Obrázek 18: Neuronová síť s limitem učení MSE=0,01

Po několika dalších výpočtech se jako optimální testovací limit, s přihlédnutím na výpočetní náročnost a přesností výsledku, ukázala celková chyba MSE=0,00005. Při takové chybě oba grafy prakticky splynou, jak zobrazuje graf 19a. Odchylky jsou, byť velmi malé, rozloženy po celé délce dat. I ve vzdálenostech menších než 1 Å se vypočtené hodnoty prakticky neliší s Lennard-Jonesovým modelem, jak je vidět na obrázku 19b.



(a) Výstup neuronové sítě pro chybu 0,00005



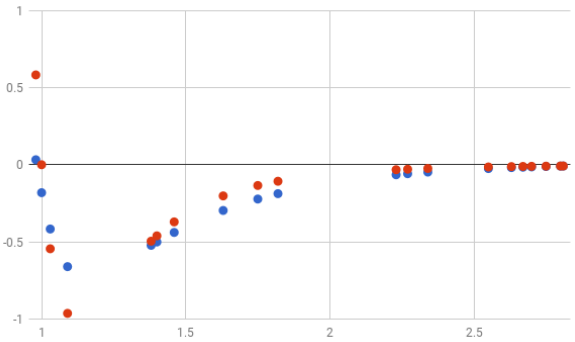
(b) Chyba neuronové sítě pro chybu 0,00005

Obrázek 19: Neuronová síť s limitem učení MSE=0,00005

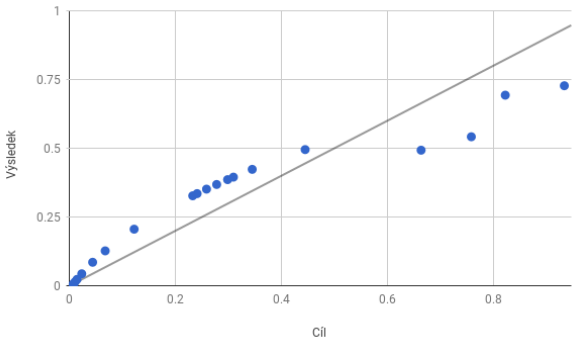
Pro další ověřováni vhodnosti mnou navržené sítě a zjištění, jak bude tato síť schopna predikovat data mimo trénovací vzorek dat, jsem použil metodu křížové validace popsané v sekci 3.7. Při použití této metody jsem rozdělil testovací data na dvě množiny podle principu náhodného vzorku dat. Rozdělení bylo náhodné a při každém výpočtu jiné. Poměr mezi trénovací množinou a testovací množinou začínal 90:10 ve prospěch trénovací množiny až k 10:90 ve prospěch testovací množiny. Pro každý poměr jsem provedl 5 výpočtů, pokaždé s jiným vzorkem dat a z výsledných hodnot jsem vypočítal aritmetický průměr, abych omezil možnost ovlivnění výsledných dat nevhodně náhodně vybranými daty, kde by například nebyla vybrána žádná data z některé důležité části grafu, jako je potenciálová jáma.

Pro vyhodnocení výpočtů posloužilo určení chyby sítě pomocí MSE (32) a sMAPE (34). Tyto způsoby výpočtu chyby jsem vybral s přihlédnutím na různé změny hodnot chyb podle druhu odchylky.

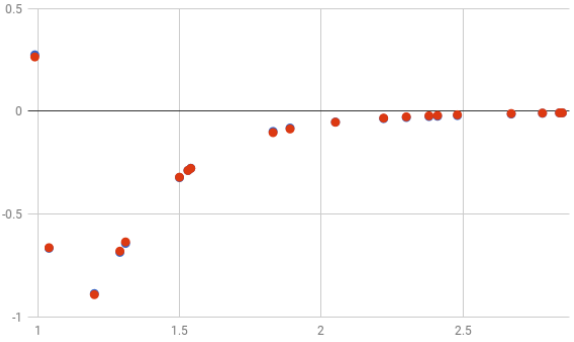
Výpočty jsem začal s poměrem 90:10 a následně jsem provedl 4 série výpočtů s postupně se snižujícím limitem chyby pro trénování od MSE=0,01 přes MSE=0,001 a MSE=0,0001 do MSE=0,00005. Určující parametry MSE a sMAPE se, postupně se zmenšujícím se limitem učení, snižovaly, jak lze vidět v tabulce 2. Při porovnání grafů v 20, jde vidět stejný trend jako bez použití křížové validace na grafech 18 a 19



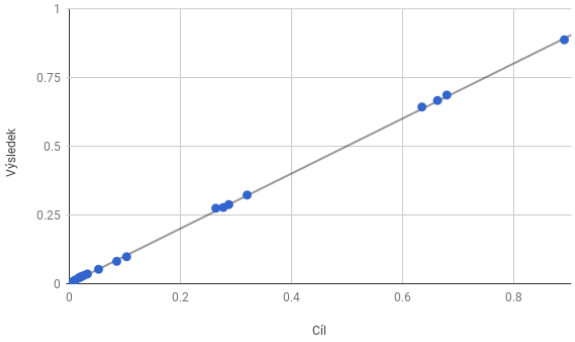
(a) Výstup neuronové sítě pro CV 90% a chybu 0,01



(b) Chyba neuronové sítě pro CV 90% a chybu 0,01



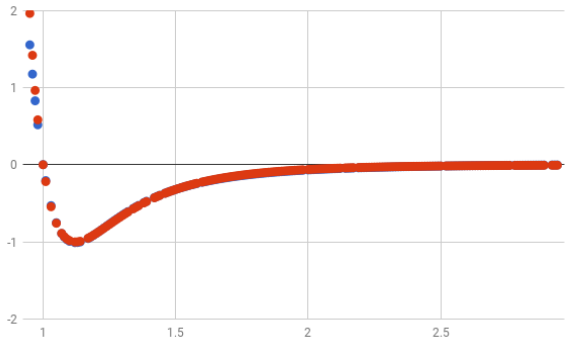
(c) Výstup neuronové sítě pro CV 90% a chybu 0,00005



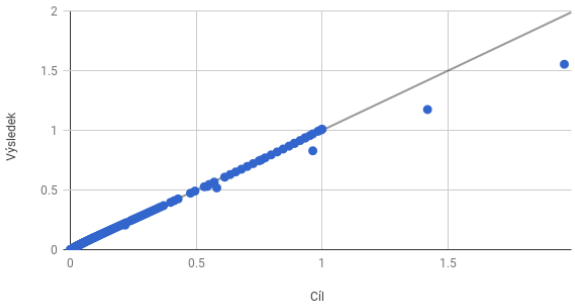
(d) Chyba neuronové sítě pro CV 90% a chybu 0,00005

Obrázek 20: Křížová validace 90%

Další výpočty byly v poměru CV 75:25, 50:50, 30:70, 20:80 a 10:90. Při těchto výpočtech se ukázala až překvapivě dobrá schopnost predikce této neuronové sítě. Rozdíl chyby MSE mezi trénovací a testovací množinou u poměru 50:50 je jen 0,0001 a při poměru 10:90 pořád výborných 0,00995. Chyba sMAPE byla dokonce u poměru 50:50 lepší než u poměru 90:10. To bylo pravděpodobně způsobeno šťastnějším výběrem náhodných dat. Ale i u poměru 10:90 byla chyba sMAPE stále nízkých 16,8. Při bližším prozkoumání grafů 21 lze vidět že oblast potenciálové jámy se predikuje stále bez větších odchylek a rozdíly se projevují především u rozestupů atomů menších než 1 Å.



(a) Výstup neuronové sítě pro CV 10% a chybu 0,00005



(b) Chyba neuronové sítě pro CV 10% a chybu 0,00005

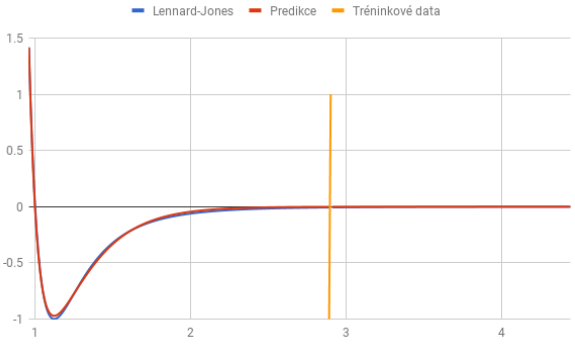
Obrázek 21: Křížová validace 10%

Všechny výsledky jsou pak uvedeny v tabulce 2.

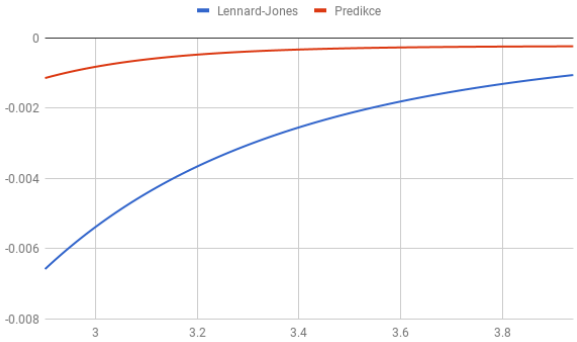
Tabulka 2: LJ testování neuronových sítí

Testovací data	Limit tréninku	Směrodatná odchylka MSE/sMAPE	MSE	sMAPE
90 %	0,01	0,006/33,1	0,01	62,5
90 %	0,001	0,001/10,5	2,6E-03	26,6
90 %	0,0001	0,001/6,86	1,05E-3	12,6
90 %	0,00005	0,000006/4,15	2,1E-5	9,6
75 %	0,001	0,001/10,9	1,6E-3	26,2
75 %	0,0001	0,0004/2,67	5,2E-4	12,3
75 %	0,00005	0,0001/3,14	7,4E-5	5,1
50 %	0,0001	0,0005/7,93	5,9E-4	12,3
50 %	0,00005	0,0001/3,73	1,5E-4	6,5
30 %	0,00005	0,012/3,6	7,6E-3	9,1
20 %	0,00005	0,002/3,49	1,2E-3	10
10 %	0,00005	0,011/6,64	1E-2	16,8

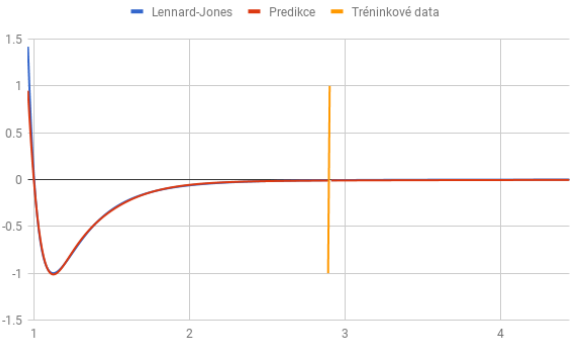
Pro zhodnocení prediktivních schopností neuronové sítě je také důležité, jak síť dokáže predikovat data mimo rozsah tréninkových dat. Toto jsem vyzkoušel pro data od vzdálenosti atomů 2.95 Å dále. Tato vzdálenost, i když je mimo náš hlavní zájem, je důležitá pro zjištění, zda si neuronová síť udržuje určitý trend, kdy by se hodnoty měly asymptoticky přibližovat k 0. To zvolená neuronová síť potvrdila, malé odchylky zobrazené na grafech 22b a 22d nemusí být na závadu jelikož při těchto velkých vzdálenost Lennard-Jonesův model ztrácí přesnost oproti skutečných hodnotám. A tak mohou být vypočtené hodnoty blíže k realitě než původní model.



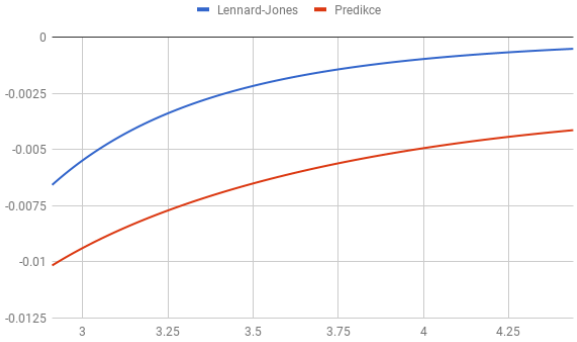
(a) Výstup neuronové sítě pro chybu 0,00005 a predikce dat



(b) Predikce dat z 22a



(c) Výstup neuronové sítě pro CV 90% a chybu 0,00005 a predikce dat



(d) Predikce dat z 22c

Obrázek 22: Predikce dat

5.2 Morseho potenciál

Pro výpočet Morseho potenciálu jsem zvolil obdobnou neuronovou feed-forward síť jako pro výpočet Lennard-Jonesova potenciálu s jedním vstupním lineárním neuronem, jednou skrytou vrstvou s 250 neuronu se stejnou logistickou aktivační funkcí (7) s jedním výstupním neuronem s lineární aktivační funkcí (6). Zvýšení počtu neuronů ve skryté vrstvě jsem provedl kvůli zhoršené prediktivní schopnosti oproti předchozím výpočtům.

Jako tréninková data posloužil vzorek 200 vstupů v rozmezí mezi 0.88-2.87 Å. Výstup tréninkových dat byl počítán podle Morseho potenciálové rovnice (2) posunutý o -1 pro lepší vizualizaci. Toto posunutí nemá žádné dopady, jelikož potenciální energie je relativní. Zvolené konstanty jsou $D_e = 1, a = 10, r_e = 1$.

Stejně jak při výpočtech Lennard-Jonesova potenciálu jsem pro ověření prediktivní schopnosti mé sítě použil metodu křížové validace (viz sekce 3.7). Pro předchozích zkušenostech jsem určil, že všechny výpočty budou mít limit učení 0,00005, jelikož tato hodnota se ukázala při předchozích měření jako nejpřesnější. Provedl jsem 9 sérií výpočtů, kdy jsem postupně ubíral velikost poměru trénovacích množiny ku testovací množiny z 90% až na 10% a tím se nepřímou

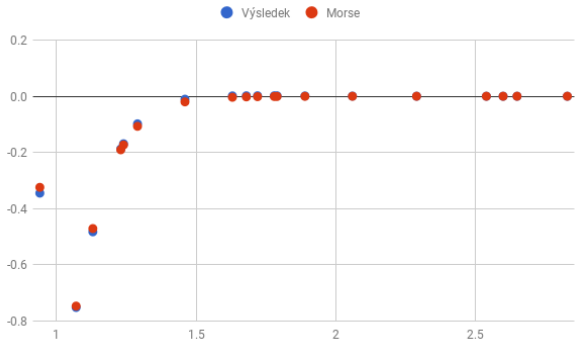
úměrou zvětšoval rozsah kontrolních dat. Jako určující hodnoty „správnosti“ neuronové sítě opět posloužily chyby MSE (32) a sMAPE (34). U každé série jsem znovu provedl 5 výpočtů a následně jsem pracoval s jejich aritmetickými průměry. Všechny výpočty probíhaly na superpočítači Salomon popsaném v sekci 4.3. Vypočtené hodnoty lze vidět v tabulce 3.

Tabulka 3: Tabulka výpočtů křížové validace Morseho potenciálu

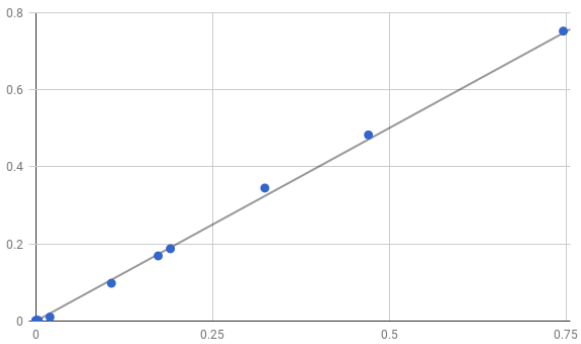
Testovací data	Limit tréninku	Směrodatná odchylka MSE/sMAPE	MSE	sMAPE
90%	0,00005	0,00004/16,3	9,41E-5	124
80%	0,00005	0,0008/12,2	5,86E-4	112
70%	0,00005	0,002/31,3	1,67E-3	119
60%	0,00005	0,012/19,1	7,77E-3	116
50%	0,00005	0,004/13,2	3,15E-3	132
40%	0,00005	0,047/25,0	3,3E-4	123
30%	0,00005	0,003/12,5	3,3E-3	125
20%	0,00005	0,014/6,35	1,09E-2	118
10%	0,00005	0,023/65,6	0,0273	131

Velikost chyb je o něco větší než v případě výpočtů Lennard-Jonesova potenciálu 2 avšak stále jsou v očekávaném rozsahu. Větší průměrná chyba sMAPE je nejspíše způsobená větší průměrnou odchylkou jednotlivých bodů než většími odchýleními do extrémů jak je možné pozorovat například na obrázku 23b a také rozdílným rozsahem dat, když v porovnání s výpočty Lennard-Jonesova potenciálu začíná vzorek už na 0,88Å oproti 0,95 Å. V těchto malých hodnotách je odchylka zpravidla větší, jelikož hodnoty začínají velmi rychle stoupat, což v kombinaci s malým počtem dat, z tohoto rozsahu, neumožňuje neuronové sítí dostatečně dobře určit tyto hodnoty. Změnou rozsahu dat, jsem chtěl lépe zjistit jak se bude neuronová síť chovat při takto dramatické změně hodnot.

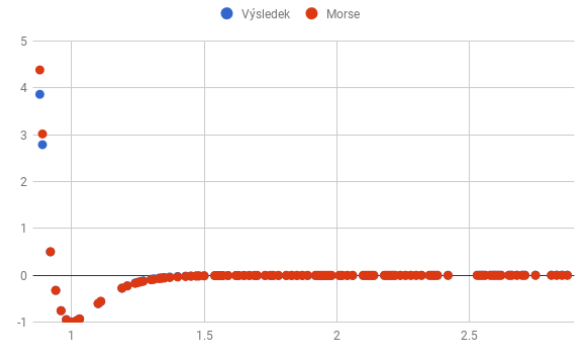
Grafy vypočtených hodnot jsou vizuálně blízké pro všechny počítané CV varianty. Větší odchylky se zpravidla objevují u velmi malých hodnot rozestupů atomů nebo u velkých rozestupů, ty však nepřestavují hlavní zkoumaný úsek a nejsou překážkou pro většinu případných reálných aplikací modelu.



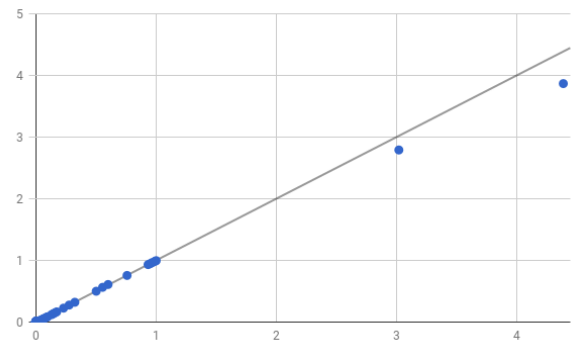
(a) Výstup neuronové sítě pro CV 90% a chybu 0,00005



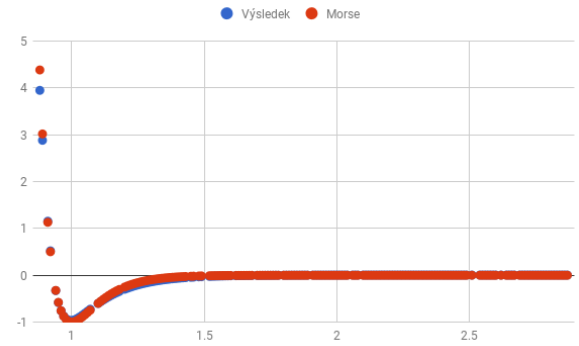
(b) Chyba neuronové sítě pro CV 90% a chybu 0,00005



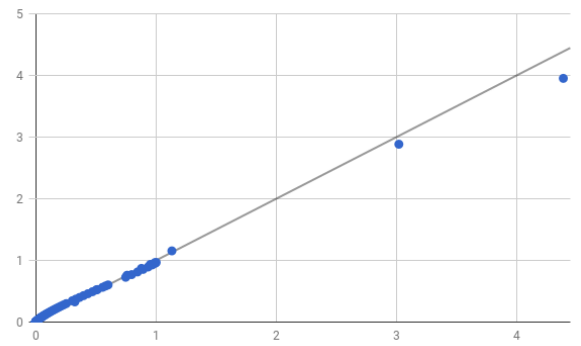
(c) Výstup neuronové sítě pro CV 50% a chybu 0,00005



(d) Chyba neuronové sítě pro CV 50% a chybu 0,00005



(e) Výstup neuronové sítě pro CV 10% a chybu 0,00005



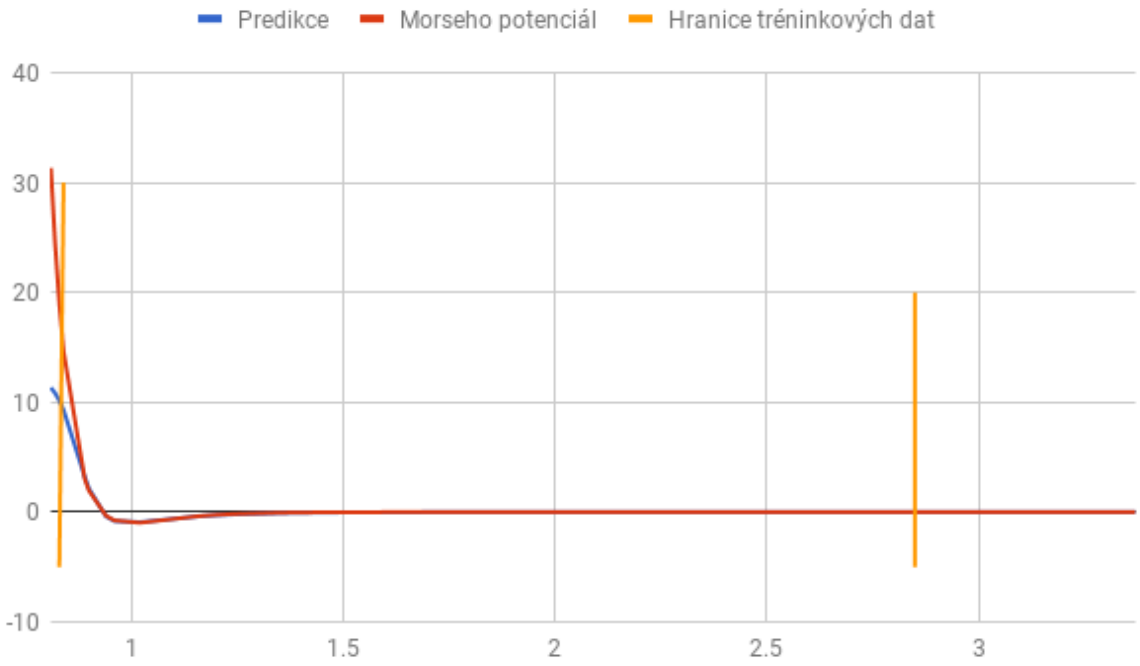
(f) Chyba neuronové sítě pro CV 10% a chybu 0,00005

Obrázek 23: Křížová validace Morseho potenciálu

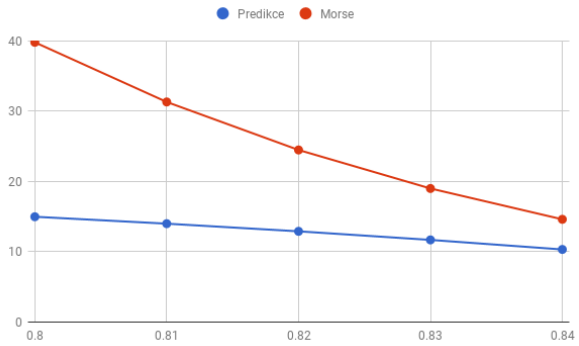
Pro zhodnocení schopností neuronové sítě predikovat data i mimo rozsah trénovacích dat bylo v této práci stěžejní udržení trendu a ověření, že nedojde k náhlému vychýlení dat mimo očekávání. To jsem testoval jak pro vzdálenosti menší něž 0,88 Å, tak pro větší vzdálenosti až k 3,35 Å.

U menších vzdáleností, v oblasti tzv. potenciálové stěny, se ukázalo, že síť má problém

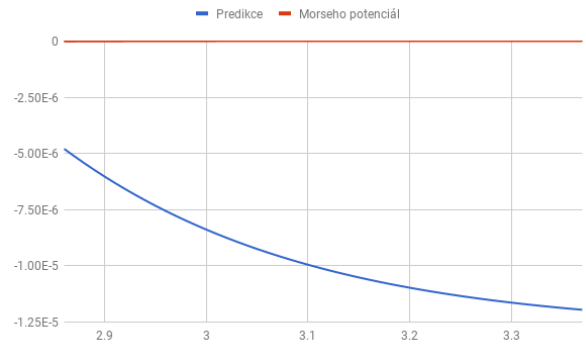
přizpůsobit se tak razantní změně trendu. Vypočtené hodnoty však správně pokračovaly v růstu, jak je vidět na grafu 24b, i když ne tak rychlém. Naštěstí se neprojevil nechtěný jev kdy by se funkční hodnoty „vychýlily“ jiným směrem než chceme.



(a) Predikce dat Morseho potenciálu



(b) Detail predikce dat pro vzdálenosti 0,8-0,84 Å



(c) Detail predikce dat pro vzdálenosti 2,85-3,35 Å

Obrázek 24: Predikce Morseho potenciálu

U větších vzdáleností se tento nechtěný jev objevil, jak je patrné na grafu 24c, i když jen v malé míře. Vypočtené hodnoty začaly opět klesat a vzdalovat se tak od 0. Tyto změny však byly „hodnotově“ velmi malé a rychlost poklesu byla poměrně malá.

6 Závěr

V této práci jsem implementoval řadu neuronových sítí z nichž jsem vybral dvě, které dosahovaly nejlepších prediktivních schopností. První reprezentuje Lennard-Jonesův potenciál, druhá pak Morseho potenciál.

Obě neuronové sítě prokázaly dobré schopnosti predikce. Vyhodnocení jsem prováděl pomocí dvou metod výpočtu chyb. Konkrétně se jednalo o MSE a sMAPE. Pro testování prediktivní schopnosti posloužila metoda křížové validace s náhodným výběrem vzorků. Zkoumaným hlediskem byla i nutná velikost „limitu učení“, resp. konvergenčního kritéria. Pomocí srovnání různých hodnot jsem pak určil, která je již dostatečná, aby síť byla schopna spolehlivé predikce a zároveň se neučila zbytečně dlouho.

Oblast potenciálové jámy vykazovala nejmenší odchylky. Pro větší vzdálenosti mezi atomy je pak extrapolace poměrně přesná, ale vyskytly se zde i malé odchylky, kdy predikce přestaly konvergovat k nule, stále jí však byly velmi blízko. Největší odchylky byly zaznamenány pro malé vzdálenosti mezi atomy, v oblasti tzv. potenciální stěny, kde se velmi rychle zvětšují odpudivé síly a tím se zvyšuje i hodnota potenciálové energie. V této oblasti měly neuronové sítě potíže přizpůsobit se tak velké změně trendu. Nedošlo však k žádnému výskytu jevu, kdy by na tuto změnu neuronová síť nezareagovala vůbec, nebo dokonce opačně, než se očekávalo.

Při dalším výzkumu bude vhodné zaměřit se na výpočty nad reálnými daty. Zvláště na reprezentaci interakcí nízkoteplotního plazmatu[4] se vzduchem, např. komplexu $[N_2/He]^+$, a to nejen v základních, ale i v excitovaných stavech. Dále bude potřeba pracovat s více než dvoučásticovými systémy, což zapříčiní zvýšení výpočetní náročnosti. Bude tak nutné prozkoumat možnosti masivní paralelizace, kdy by výsledný software měl být schopen nejen distribuce výpočtu mezi uzly, ale využívat k paralelizaci i vlákna a efektivní vektorizaci.

Při současné implementaci posloužil jako deskriptor, tj. vstupní hodnota, vzdálenost mezi atomy. Toto řešení umožnilo vyhnout se problémům přítomným u některých souřadnicových systémů. Například v kartézském systému souřadnic by otočení nebo posunutí částic způsobilo změny výstupů neuronové sítě díky odlišným vstupům, zatímco reálně zůstává vzdálenost mezi částicemi stejná a potenciálová energie se tak nemění. Vzdálenost mezi částicemi však není obecně nejlepším řešením. V případě velkých systémů by při nutnosti mít vzdálenost každé částice s každou, výrazně narostl počet vstupů do neuronové sítě. Dále tak bude nutné zabývat se i výzkumem translačně a rotačně invariantních souřadnicových systémů jako vstupů a navázat tak na již probíhající výzkum[8].

Výsledky této práce poslouží k teoretickému návrhu a pozdější implementaci masivně paralelní knihovny pro tvorbu neuronových sítí, která umožní efektivní reprezentaci i v případě výrazně rozsáhlých systémů.

A Zdrojové kódy

```
public static void CrossValidationRandom(DataSet dataset, double percentage){
       int row;
       DataSet newdataset=dataset;

      if(percentage<=0.0 || percentage >=100.0) {
          System.err.println("Parameter percentage out of range");
       }

       Double rows = dataset.size()*(percentage/100.0);
       DataSet newData = new DataSet(1, 1);
       for(int i=0;i<rows;i++){
          Random random = new Random();
          row= random.nextInt(newdataset.size());
          newData.add(newdataset.get(row));
           newdataset.removeRowAt(row);
       }
   }
```

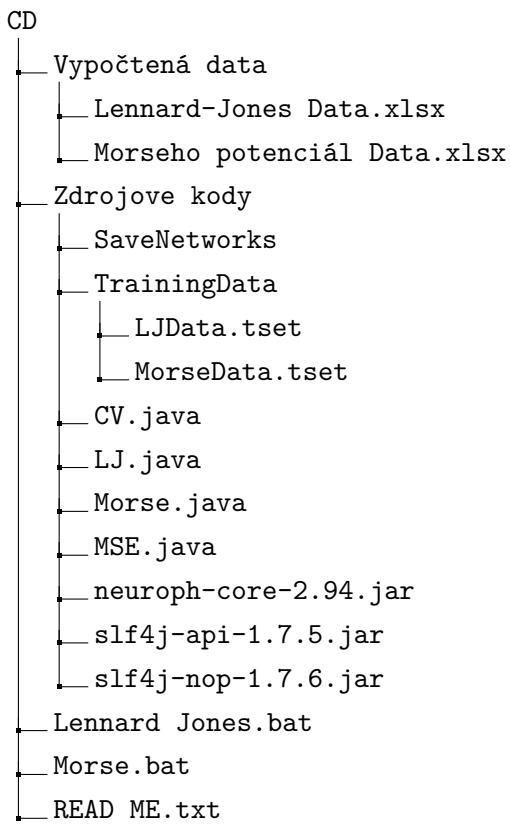
Výpis 1: Křížová validace

```
   public static void create_network(){
       List<Integer> neuronsInLayers = new ArrayList();
       neuronsInLayers.add(1);
       neuronsInLayers.add(200);
       neuronsInLayers.add(1);

       NeuralNetwork myNetwork = new MultiLayerPerceptron(neuronsInLayers, TransferFunctionType.
           SIGMOID);
       Neuron neuron = (Neuron)myNetwork.getOutputNeurons().get(0);
       neuron.setTransferFunction(new Linear());
```

Výpis 2: Vytvoření sítě pro výpočet Lennard_jonesova potenciálu

B Obsah CD


CD
Vypočtená data
Lennard-Jones Data.xlsx
Morseho potenciál Data.xlsx
Zdrojove kody
SaveNetworks
TrainingData
LJData.tset
MorseData.tset
CV.java
LJ.java
Morse.java
MSE.java
neuroph-core-2.94.jar
slf4j-api-1.7.5.jar
slf4j-nop-1.7.6.jar
Lennard Jones.bat
Morse.bat
READ ME.txt

Odkazy

[1] Attila Szabo a Neil S Ostlund. *Modern quantum chemistry: introduction to advanced electronic structure theory*. Courier Corporation, 2012.

[2] Lubomír Skála. *Kvantová teorie molekul*. Karolinum, 1994.

[3] Jon R Maple, Uri Dinur a Arnold T Hagler. "Derivation of force fields for molecular mechanics and dynamics from ab initio energy surfaces". In: *Proceedings of the National Academy of Sciences* 85.15 (1988), s. 5350–5354.

[4] Mounir Laroussi. *Plasma medicine: applications of low-temperature gas plasmas in medicine and biology*. Cambridge University Press, 2012.

[5] Mohammed Yousfi et al. "Low-temperature plasmas at atmospheric pressure: toward new pharmaceutical treatments in medicine". In: *Fundamental & clinical pharmacology* 28.2 (2014), s. 123–135.

[6] Adam Coates et al. "Deep learning with COTS HPC systems". In: *International Conference on Machine Learning*. 2013, s. 1337–1345.

[7] Trishul M Chilimbi et al. "Project Adam: Building an Efficient and Scalable Deep Learning Training System." In: *OSDI*. Sv. 14. 2014, s. 571–582.

[8] Jorg Behler. "Atom-centered symmetry functions for constructing high-dimensional neural network potentials". In: *The Journal of chemical physics* 134.7 (2011), s. 074106.

[9] Dr. Ian Hunt. *Bonding in H2*. University of Calgary. URL: `http://www.chem.ucalgary.ca/courses/351/Carey5th/Ch02/ch2-2-1.html` (cit. 24. 04. 2018).

[10] Chang Lyoul Kong. "Combining rules for intermolecular potential parameters. II. Rules for the Lennard-Jones (12–6) potential and the Morse potential". In: *The Journal of chemical physics* 59.5 (1973), s. 2464–2467.

[11] *Morse-potential.png*. wikipedia.com. URL: `https://en.wikipedia.org/wiki/Morse_potential#/media/File:Morse-potential.png` (cit. 24. 04. 2018).

[12] Robert J Le Roy et al. "Accurate analytic potentials for Li 2 (X Σ 1 g+) and Li 2 (A Σ 1 u+) from 2 to 90 Å, and the radiative lifetime of Li (2 p)". In: *The Journal of chemical physics* 131.20 (2009), s. 204309.

[13] Daniel Svozil, Vladimir Kvasnicka a Jiri Pospichal. "Introduction to multi-layer feed-forward neural networks". In: *Chemometrics and intelligent laboratory systems* 39.1 (1997), s. 43–62.

[14] Ivo Vondrák. *Neuronové sítě*. Vysoká škola báňská-Technická univerzita Ostrava, 2009.

[15] Kun Jeremy. "Neural Networks and the Backpropagation Algorithm". In: *Math Programming Posted on December* 9 (2012).

[16] Jonathan Richard Shewchuk. “An Introduction to the Conjugate Gradient Method Without the Agonizing Pain”. In: (1994).

[17] Iveta Mrázová. “Neuronové sítě. Prezentace k přednášce k předmětu Neuronové sítě (NAIL002)”. In: 2014, s. 22–29. URL: http://ksvi.mff.cuni.cz/~mraz/nn/Neuronove_Site_Prednaska_AM.pdf.

[18] *Learning and Adaptation*. tutorialspoint.com. URL: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_learning_adaptation.htm (cit. 24. 04. 2018).

[19] Michael Nielsen. “How the backpropagation algorithm works”. In: *Neural networks and deep learning. Determination Press* (2015).

[20] Hafidz Zulkifli. *Understanding Learning Rates and How It Improves Performance in Deep Learning*. towardsdatascience.com. URL: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 (cit. 24. 04. 2018).

[21] S Volokitin. “Parallel implementation of a neural network learning algorithm”. In: *International Journal of Computer Applications* 85.3 (2014).

[22] Statsoft CR. “Úvod do neuronových sítí”. In: 2013, s. 4–5. URL: http://www.statsoft.cz/file1/PDF/newsletter/2013_02_05_StatSoft_Neuronove_site_linky.pdf.

[23] *Mean Squared Error: Definition and Example*. statisticshowto.com. URL: http://www.statisticshowto.com/mean-squared-error/#MSE (cit. 24. 04. 2018).

[24] Chris Tofallis. “A better measure of relative prediction accuracy for model selection and model estimation”. In: *Journal of the Operational Research Society* 66.8 (2015), s. 1352–1362.

[25] *K-fold vs. Monte Carlo cross-validation*. stackexchange.com. URL: https://stats.stackexchange.com/questions/51416/k-fold-vs-monte-carlo-cross-validation/60967 (cit. 24. 04. 2018).

[26] *Směrodatná odchylka*. matematika.cz. URL: https://matematika.cz/smerodatna-odchylka (cit. 24. 04. 2018).

[27] *tensoflow*. URL: https://www.tensorflow.org/ (cit. 24. 04. 2018).

[28] *Caffe*. URL: http://caffe.berkeleyvision.org/ (cit. 24. 04. 2018).

[29] *Torch*. URL: http://torch.ch/ (cit. 24. 04. 2018).

[30] *Neuroph*. Neuroph. URL: http://neuroph.sourceforge.net/ (cit. 24. 04. 2018).

[31] TOP500.org. *TOP500 List*. TOP500.org. 2017. URL: https://www.top500.org/list/2017/11/?page=1.

[32] *IT4I Salomon Cluster*. Národní superpočítačové centrum. URL: https://docs.it4i.cz/salomon/introduction/ (cit. 24. 04. 2018).