Vysoká škola ekonomická v Praze
Fakulta financí a účetnictví
Katedra bankovnictví a pojišťovnictví
Studijní obor: Finance

Vyhodnocování pojistných událostí pomocí strojového učení

Autor bakalářské práce: Daniel Havíř
Vedoucí práce: prof. Ing. Eva Ducháčková, CSc.
Rok obhajoby: 2018

Čestné prohlášení

Prohlašuji, že bakalářskou práci na téma „Vyhodnocování pojistných událostí pomocí strojového učení“ jsem vypracoval samostatně a veškerou použitou literaturu a další prameny jsem řádně označil a uvedl v přiloženém seznamu.

V Praze dne 10.4.2018 Podpis

Poděkování

Rád bych poděkoval prof. Ing. Evě Ducháčkové, CSc. za ochotu a vstřícnou komunikaci při psaní této práce. Dále děkuji MMath Jakubovi Šiškovi za zasvěcení do problematiky vyhodnocování pojistných událostí v praxi a inspiraci pro výběr tématu.

Vyhodnocování pojistných událostí pomocí strojového učení

Abstrakt

Tato bakalářská práce se zabývá metodami strojového učení, zejména neuronovými sítěmi a jejich uplatněním pro vyhodnocování pojistných událostí. Je popsána jedna konkrétní skupina metod, která je následně aplikována na jednom konkrétním případu vyhodnocování severity škod u dopravních nehod. V úvodní části této práce je vysvětlena motivace pro modelování škod u pojistných událostí a jeho význam při tvorbě pojistných rezerv. Pro modelový příklad jsou použita data americké pojišťovny AllState, která je zpřístupnila v rámci online soutěže o nejlepší predikční model v roce 2016. V praktické části této práce jsou prezentovány 3 modely neuronových sítí, jejichž výsledek je zdokumentován a nejlepší z těchto modelů je použit pro výsledný ensemble modelů. Tento ensemble dosáhl o téměř 8% lepšího výsledku než výchozí model pojišťovny pro danou soutěž. Tato práce tak demonstruje potenciál metod strojového učení v pojistné praxi.

Klíčová slova: pojistná plnění, modelování severity škod, strojové učení, hlubokové učení, neuronová síť

Machine Learning Techniques for Insurance Claims Evaluation

Abstract

This bachelor thesis describes machine learning techiques, neural networks in particular and their application for insurance claims evaluation. One set of such techniques is then applied to a specific task of modeling the severity of insurance claims related to car accidents. The introductory part explains the motivation to model insurance claims and its importance in the creation of insurance reserves. The model example in this thesis uses data from the american insurance company AllState. This data was made available for an online predictive modelling competition held by the insurance company in 2016. In the applied section of this thesis, three different architectures of neural networks are presented and their performance is evaluated. The best neural network is then used for the final model ensemble. The ensemble achieves a result that is more accurate by 8% than the benchmark value submitted by the competition organizer. This thesis hence demonstrates the potential of machine learning applications in the insurance industry.

Key words: insurance claim, claim severity modeling, machine learning, deep learning, neural network

Notace

x	Skalár
$\mathbf{x}$	Vektor
$\mathbf{0}$	Nulový vektor
$\mathbf{X}$	Matice
$\|\mathbf{x}\|$	Eukleidovská norma vektoru $\mathbf{x}$
$\frac{dy}{dx}$	Derivace y vzhledem k x
$\frac{\partial y}{\partial x}$	Parciální derivace y vzhledem k x
$\nabla_{\mathbf{x}} y$	Gradient y vzhledem k $\mathbf{x}$
$a \sim P$	Náhodná proměnná a má rozdělení P
$a \sim U$	Náhodná proměnná a má rovnoměrné rozdělení
$log(x)$	Přirozený logaritmus x (logaritmus o základu Eulerova čísla)
$\sigma(x)$	Sigmoid funkce x
$\mathbb{H}(f)$	Obor hodnot funkce f

Seznam zkratek

RBNS	Reported But Not Settled
IBNR	Incurred But Not Reported
MSE	Mean Squared Error
MAE	Mean Absolute Error
SGD	Stochastic Gradient Descent
Adam	Adaptive Moment Estimation
ReLU	Rectified Linear Unit
PReLU	Parametric Rectified Linear Unit

Obsah

1 Úvod **1**

2 Pojistná část **3**

2.1 Neživotní pojištění . . . 3

2.2 Pojištění majetku . . . 5

2.2.1 Havarijní pojištění . . . 5

2.3 Technické rezervy . . . 6

2.4 Obecné poznámky k pojistné části . . . 7

3 Strojové učení **9**

3.0.1 Lineární regrese . . . 10

3.0.2 Generalizace . . . 11

3.1 Deep Learning . . . 12

3.2 Feed-Forward neuronová síť . . . 14

3.3 Loss funkce . . . 15

3.4 Optimalizace . . . 16

3.4.1 Gradient descent . . . 16

3.4.2 Stochastic gradient descent . . . 17

3.4.3 Backpropagation . . . 17

3.4.4 Adam . . . 18

3.5 Aktivační funkce . . . 19

3.6 Hyperparametry . . . 22

3.7 Inicializace parametrů . . . 23

3.8 Regularizace . . . 24

3.9 Shrnutí a obecné poznámky k části o strojovém učení . . . 25

4 Modelový příklad **26**

4.1 Dataset . . . 26

4.2 Vyhodnocovací metrika . . . 27

4.3 Modely . . . 27

4.3.1 Trénovací hyperparametry . . . 29

4.3.2 Adam vs SGD . . . 30

4.3.3 Výsledky jednotlivých modelů . . . 30

4.4 Ensemble a křížová validace . . . 31

4.5 Srovnání výsledků . . . 32

5 Závěr **33**

Zdroje **34**

Seznam obrázků **37**

Seznam tabulek **37**

1 Úvod

Tato práce se zabývá využitím metod strojového učení[1] pro vyhodnocovaní pojistných událostí. Nejprve se pokusí vysvětlit, kde jsou v rámci pojišťovnictví takové metody užitečné a proč je pro pojišťovnu důležité vyhodnocovat pojistné události co možná nejrychleji a nejpřesněji. Dále tato práce představí nové metody vyhodnocování a modelování pojistných událostí. Přestože jsou část teorie a některé principy představené v této práci uplatnitelné pro širší spektrum pojistných událostí, v jádru této práce je jeden konkrétní případ - modelování výše škod u dopravních nehod, které se vztahuje k havarijnímu pojištění či povinnému ručení z provozu motorových vozidel. Na tento konkrétní případ se většina teoretické části postupně zaměřuje a vytváří tak teoretický základ pro následnou praktickou část.

Cílem této práce je popsat a demonstrovat jednu vybranou skupinu metod strojového učení, jmenovitě neuronové sítě, na jednom konkrétním příkladu z pojišťovnictví. Přestože se jedná pouze o jednu skupinu metod, je tato skupina poněkud rozsáhlá a obsahuje několik částí. V praktické části pak tato práce demonstruje tyto metody na datech americké pojišťovny AllState. Tento příklad pak může sloužit jako ukázka potenciálu, který strojové učení skýtá pro pojišťovnictví. Tato práce se nesnaží popisovat veškeré možné případy využití strojového učení v odvětví pojišťovnictví, nesnaží se pokrýt celou škálu pojistných událostí a ani nepopisuje všechny metody strojového učení, které jsou v pojišťovnictví uplatnitelné.

Strojové učení a zejména neuronové sítě jsou v současnosti pravděpodobně nejvýznamnější částí oboru umělé inteligence, kterému se dostává nemalé pozornosti a stává se tak jedním z módních pojmů. Pojišťovnictví je ideálním odvětvím pro strojové učení, jelikož řeší celou řadou matematických problémů a pro jejich řešení má většinou dostatek dat. Z toho důvodu bylo motivací pro výběr tématu alespoň částečně osvětlit část metod a nastínit jejich uplatnění v pojistném odvětví.

Tématika neuronových sítí se těší velkému zájmu ze strany výzkumné komunity po celém světě, což v důsledku umožnilo výrazný posun celého oboru během posledních pár let. Jakkoliv se tato práce snaží pracovat převážně s českými pojmy všude, kde je to možné, v některých případech jsou anglické termíny více rozšířené a neexistuje pro ně zatím v českém jazyce ekvivalent.

Tato práce se skládá ze dvou teoretických bloků a jednoho praktického. První teoretická část popisuje základní pojmy a členění v pojišťovnictví a vysvětluje kontext modelového případu pro pojišťovnu. Druhá teoretická část popisuje konkrétní metody strojového učení, resp. neuronových sítí, které byly použity pro modelový příklad. Praktická část pak tyto

[1] Poněkud rozšířenější pak je anglický výraz machine learning

koncepty aplikuje pro modelování výše škod u dopravních nehod. Modely v praktické části byly napsány v programovacím jazyce Python [28] za pomoci programovací knihovny PyTorch [27]. Zdrojový kód s modely z praktické části je dostupný na adrese: https://github.com/danielhavir/claims-severity-model

2 Pojistná část

Komerční pojištění je finanční služba, kde se v případě vzniku pojistné události **pojistitel** zavazuje **pojistníkovi** vyplatit finanční krytí neboli **pojistné plnění** ve sjednaném rozsahu. Pojistník se na oplátku zavazuje za tuto ochranu platit určitý finanční obnos, tzv. **pojistné**. **Pojistnou událostí** se rozumí určitá nahodilá (náhodná) událost, kterou nelze předem předpovědět a jejíž důsledky negativně (většinou finančně) ovlivňují pojistníka. Tato událost je upravena v pojistné smlouvě a vymezuje krytá rizika [24].

Základní členění komerčního pojištění podle krytých rizik je [9]:

- **Životní pojištění** - kryje rizika smrti a dožití. Pojistná smlouva se většinou uzavírá na delší období a zahrnuje spořící složku.
- **Neživotní pojištění** - kryje širokou škálu rizik spojenou s majetkem, odpovědností aj. Pojistná smlouva se pak většinou uzavírá na kratší období, řádově nižší jednotky let.

Tato práce se dále věnuje výhradně tématice **neživotního pojištění**.

Obecně se dá pojištění členit následovně:

- Pojištění obnosová - pojistné plnění se rovná nebo odvíjí od předem sjednané částky
- Pojištění škodová - pojistné plnění se odvíjí od vzniklé škody

S ohledem na téma této práce je výše zmíněné členění důležité. U **obnosového pojištění** je pojistná částka předem sjednaná. Při vyhodnocování pojistného plnění se pak pouze rozhoduje, zda má být vyplacena či nikoliv. Jinými slovy, kontroluje se, zda skutečně nastala pojistná událost a byly splněny náležitosti pojistné smlouvy. Jedná se tedy o binární klasifikaci. V případě **pojištění škodového** částka předem sjednaná není, může však být sjednána její horní hranice. Výše pojistného se pak odvíjí od vzniklé škody. V momentě pojistného plnění se rozhoduje o výši pojistného plnění, neboli kolik se má pojištěnci vyplatit. V takovém případě se jedná o regresní problém.

2.1 Neživotní pojištění

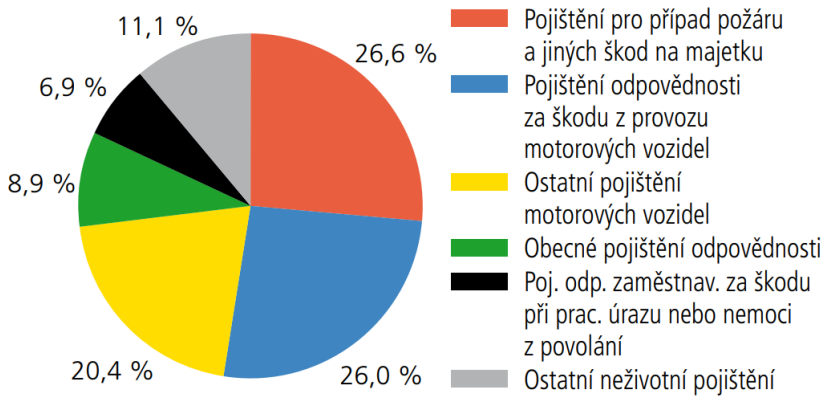
Neživotní pojištění kryje rizika neživotního charakteru, a to zejména rizika věcných škod, rizika újmy na zdraví nebo rizika finančních ztrát. Zákon č. 277/2009 Sb., o pojišťovnictví [37] dělí skupiny neživotních pojištění takto:

- Pojištění úrazu a nemoci
- Pojištění motorových vozidel
- Námořní a dopravní pojištění
- Letecké pojištění
- Pojištění proti požáru a jiným majetkovým škodám
- Pojištění odpovědnosti za újmu
- Pojištění úvěru a záruky
- Souhrnné neživotní pojištění

Pro zachování konzistentnosti s použitou literaturou se však tato práce dále řídí dělením z pohledu charakteristiky produktů podle [9], které dělí neživotní pojištění následovně:

- Neživotní pojištění osob - např. úrazová pojištění, komerční nemocenské a zdravotní pojištění, cestovní pojištění aj.
- Pojištění majetková - např. pojištění domácnosti, havarijní pojištění, pojištění budov, dopravní pojištění aj.
- Pojištění finančních ztrát - např. úvěrová pojištění, pojištění spojená s rizikem přerušení provozu aj.
- Pojištění odpovědnosti za škody - např. pojištění odpovědnosti za škodu způsobenou provozem vozidla

V České Republice vypadala struktura trhu neživotního pojištění v roce 2016 následovně:



Obrázek 1: Odvětví neživotního pojištění tuzemských pojišťoven dle výše hrubého předepsaného pojistného. Zdroj: [1]

Z výše zmíněného členění je pro tuto práci stěžejní a dále se zaměří zejména na **pojištění majetku**. Do jisté míry se pak dotkne i pojištění odpovědnosti za škody, konkrétně povinného ručení z provozu motorových vozidel, jelikož v případě dopravních nehod jsou tato pojištění podobná ve vyhodnocení (v obou případech jde o vyčíslení škod). Důvodem pro toto zaměření je modelový příklad praktické části, který se zabývá **vyhodnocováním rozsahu škod způsobených dopravní nehodou**. Obecně se však zdá, že metody popisované v této práci jsou v současnosti rozšířenější právě u pojištění majetku než např. u životního pojištění nebo pojištění osob.

Dalším důvodem, proč se tato práce zabývá právě vyhodnocováním rozsahu škod způsobených dopravní nehodou je i fakt, že pojištění odpovědnosti za škodu z provozu motorových vozidel a ostatní pojištění motorových vozidel společně tvoří téměř 50% objemu neživotního pojištění v ČR (viz obrázek (1)).

Mimo již zmíněné havarijní pojištění a povinné ručení se vyhodnocování pojistných událostí pomocí strojového učení, resp. umělé inteligence, rozšířilo u pojištění domácnosti. Za průkopníka na poli pojištění domácnosti na americkém trhu se dá považovat poměrně nová společnost Lemonade [22]. Ta byla schopna díky metodám strojového učení zautomatizovat proces vyhodnocování pojistné události a snížit tak dobu schvalování pojistných plnění na řády minut. Na svých stránkách pak uvádí, že nejkratší doba mezi nahlášením pojistné události a vyplacením pojistného plnění byla pouhé 3 vteřiny [23].

2.2 Pojištění majetku

Pojištění majetku kryje rizika možných škod na majetku, např. poškození, zničení, odcizení apod. U pojištění majetku se nejčastěji využívá škodové formy pojištění. [9]

V kontextu oceňování majetkových škod můžeme pojištění dělit na [9]:

- **Pojištění na časovou cenu** - výše pojistného plnění se odvíjí od hodnoty pojištěného poškozeného majetku těsně před pojistnou událostí (s ohledem na opotřebení)
- **Pojištění na novou cenu** - pojistné plnění je vyplaceno ve výši umožňující obnovu majetku nebo cenu opravy, pomocí které se uvádí majetek do původního stavu

2.2.1 Havarijní pojištění

Havarijní pojištění kryje rizika škody na motorových vozidlech v důsledku havárie, kolize nebo vandalství. Toto pojištění se často konstruuje jako pojištění na novou cenu. To znamená, že výše pojistného plnění se odvíjí od nákladů na opravu, které ovšem nesmí

překročit časovou cenu vozidla před havárií. [9] Ve většině případů se finální výše pojistného plnění určuje deterministicky, tj. odvíjí se od vyčíslení nákladů na opravu ze strany auto opravny či servisu.

Na rozdíl od např. pojištění domácnosti tak strojového učení v případě havarijního pojištění neurychlí proces vyhodnocování pro pojištěnce. Může však vylepšit modelování výše škod, které je důležité pro tvorbu adekvátních technických rezerv.

2.3 Technické rezervy

V případě nahodilé události se pojišťovna zavazuje vyplatit pojistnou částku. V mnoha případech však není možné vyplatit pojistnou částku okamžitě a to zejména ze 3 důvodů [36]:

1. Může docházet ke zpoždění v nahlášení pojistné události, tzn. existuje doba mezi nastáním pojistné události a nahlášením události pojišťovně. Tato doba může trvat i několik let např. při pojištění proti znečištění životního ovzduší.
2. Po nahlášení pojistné události může nastat zdržení ve vypořádání pojistné smlouvy. Ke zdržení dochází v důsledku vyjasňování všech okolností pojistné události a v některých případech se může protáhnout na dlouhou dobu (např. v důsledku soudního sporu). Toto zdržení může nastat i v případě, čeká-li se na konečné vyčíslení škod třetí stranou (auto opravnou, servisem apod.).
3. I v případě, že byla pojistná smlouva vypořádána, může být událost znovu otevřena, např. v případě výskytu nových informací.

V důsledku těchto zdržení musí pojišťovna vyrovnávat časový nesoulad vytvářením rezerv na finanční krytí. Současně musí tvořit rezervy na budoucí plnění, ke kterým se zavazuje v nově sjednaných pojistných smlouvách. Technické rezervy jsou tak klíčovou součástí hospodaření pojišťovny. [9]

Tento časový nesoulad má též nepříznivé účinky pro pojištěnce. Ti v případě nastání pojistné události často musí čekat na pojistná plnění několik dní až týdnů. S vývojem technologií si spotřebitelé finančních služeb zvykli na okamžité vyhotovení svých požadavků, např. bankovních převodů a finančních transakcí, čehož se jim ovšem v pojišťovnictví nedostává. Snaha urychlit vyhodnocování pojistných událostí a snížit administrativní zátěž je tak částečně motivována i cílem zlepšit celkovou reputaci odvětví v očích koncových spotřebitelů.

Technické rezervy se tvoří z přijatého pojistného a platí se z nich pojistná plnění [9]. Mezi tyto rezervy mimo jiné patří:

- Rezerva na pojistná plnění z událostí vzniklých, ale nenahlášených (IBNR, Incurred But Not Reported)
- Rezerva na pojistná plnění z událostí nahlášených, ale nezlikvidovaných (RBNS, Reported But Not Settled)
- Rezerva na nezasloužené pojistné

Přesná a včasná tvorba konkrétních technických rezerv ovlivňuje hospodaření a zisk pojišťovny. Z toho důvodu je důležité pro správné ekonomické fungování pojišťovny mít co nejpřesnější odhady počtu událostí (zejména pro IBNR rezervy) a výše škod (pro RBNS rezervy), které umožní pojišťovnám adekvátně reagovat při výskytu nahodilých událostí. Tyto odhady se dále využívají při výpočtu ceny pojistného pro pojistné smlouvy v budoucnosti.

V kontextu praktické části této práce je pro pojišťovnu modelování závažnosti škod důležité pro stanovení výše RBNS rezerv.

2.4 Obecné poznámky k pojistné části

Existuje několik způsobů, jak vyhodnocovat pojistné události a vytvářet modely pro odhadování výše pojistných škod. Různé metody tradiční pojistné matematiky a různě agregované data sety často vedou k velmi odlišným výsledkům. Až doposud výběr vhodné metody pro konkrétní data set vyžadoval dostatek zkušeností s aparátem pojistné matematiky. [36] S vývojem technologií a metod výpočetní techniky se však objevují nové metody, které často fungují lépe navzdory tomu, že nejsou specifické pro pojišťovnictví ani jiné konkrétní odvětví a nevyžadují zkušenosti z daného oboru. V porovnání s pojistnou matematikou se navíc vyvíjí podstatně kratší dobu.

Důkazem může být příklad americké pojišťovny AllState, která zveřejnila datový soubor s vlastnostmi automobilů a vypsala soutěž v predikci, u kterých z nich se bude v budoucnu řešit osobní odpovědnost za škody. Vítězné řešení pak bylo o více než 270 procent lepší než výchozí prognóza společnosti. [4] [16] Zajímavostí může být i fakt, že vítěz a většina účastníků této soutěže se nijak nespecializovali na odvětví pojišťovnictví ani pojistnou matematiku.

Strojové učení má potenciál nejenom vylepšit současné matematické modely v pojišťovnictví, ale také rozšířit možnosti vyhodnocování událostí pomocí nestrukturovaných dat, jako jsou například textové dokumenty, obraz nebo video, které doposud bylo velmi složité a omezené zpracovat. To v důsledku umožňuje vytvářet modely v případech, ve kterých to dosud nebylo možné (např. nastavit výši pojistného u pojištění

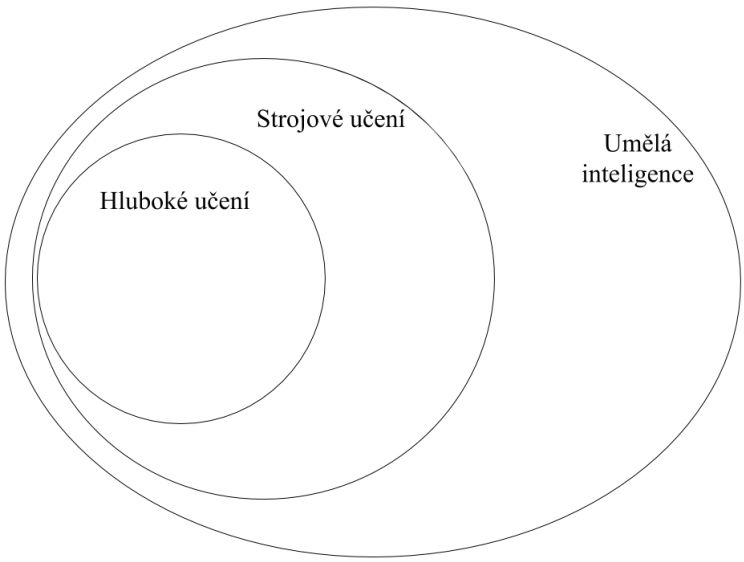
budov na základě satelitních snímků [5]). Lepší schopnost predikce by dále mohla do jisté míry umožnit zautomatizovat vyhodnocování některých pojistných událostí, které byly až doposud vyhodnocovány dedikovanými zaměstnanci pojišťovny, tzv. likvidátory. To může nejen snížit náklady pojišťoven, ale také zlepšit zkušenosti koncových zákazníků.

Předpovídat, zda se tato očekávání naplní je však nad rámec této práce. Zatímco strojové učení a metody umělé inteligence mají šanci zefektivnit provoz pojišťoven a snížit ceny pojištění, vyvstávají nové otázky a protiargumenty ohledně soukromí, spojené s neustálým sběrem dat.

3 Strojové učení

Strojové učení (z anglického výrazu Machine Learning) je v zásadě aplikovaná statistika s větším důrazem na využítí výpočetního výkonu pro statistické odhady složitých funkcí a menším důrazem na dokazování intervalů spolehlivosti těchto funkcí. [12] Algoritmus strojového učení je takový algoritmus, který je schopný rozpoznávat vzorce a učit se z dat.

Deep Learning (česky též hluboké učení), kterým se primárně zabývá tato práce je pododvětví strojového učení. Pro pochopení konceptů deep learningu je nicméně nezbytné znát alespoň základní principy strojového učení. V počítačových vědách se jak deep learning, tak strojové učení zařazuje pod obor umělé inteligence. Vztah hlubokého učení a strojového učení k umělé inteligenci je zjednodušeně vyjádřen Vennovým diagramem na obrázku:



Obrázek 2: Vennův diagram popisující vztah umělé inteligence, strojového učení a deep learningu. Zdroj: [12]

Většina metod strojového učení se dá rozdělit do 2 kategorií:

1. Supervised learning - zahrnuje metody, které se učí z anotovaných dat. Každé datum obsahuje sadu proměnných a příslušnou cílovou hodnotu, která může být spojitá (regresní úloha) či kategorická (klasifikační úloha). Jedna z takových metod se například může učit z historických dat o pojistných událostech a označení, zda bylo či nebylo přiznáno pojistné plnění.

2. Unsupervised learning - zahrnuje metody, které se snaží v neanotovaném datasetu hledat strukturu, případně aproximovat jeho pravděpodobnostní rozdělení. Velkou část těchto algoritmů tvoří shluková analýza.

Tato práce se dále zabývá výhradně **supervised learning metodami**.

Definice strojového učení jakožto matematické metody, která se učí z dat, se může zdát poněkud abstraktní. Konkrétním případem takové metody pak je například lineární regrese. V nejjednodušším pojetí je lineární regrese systém, který se ze vstupního vektoru $\mathbf{x} \in \mathbb{R}^n$ snaží predikovat výstupní skalár $y \in \mathbb{R}$. K příkladu lineární regrese se v zápětí dostaneme, jelikož na něm stojí složitější koncepty hlubokého učení.

3.0.1 Lineární regrese

Pravděpodobně nejjednodušší metodou strojového učení je lineární regrese:

$$\hat{y} = f(\mathbf{x}, \mathbf{w}, b) = \mathbf{w}^\top \mathbf{x} + b = \sum_i (w_i x_i) + b \tag{1}$$

Kde $\mathbf{x} \in \mathbb{R}^n$ je vektor vstupních dat, $\mathbf{w} \in \mathbb{R}^n$ je vektor parametrů a $b \in \mathbb{R}$ je konstanta. Parametrům se též někdy říká váhy (z anglického weights).

Malou kosmetickou úpravou pro zjednodušení zápisu je rozšířit vektor $\mathbf{w}$ o konstantu b a vektor dat $\mathbf{x}$ o 1. Konkrétně mějme $\mathbf{w} = (w_1, w_2, ..., w_n)^\top$ a $\mathbf{x} = (x_1, x_2, ..., x_n)^\top$, pak z rovnice (1) plyne $\hat{y} = w_1 x_1 + w_2 x_2 + ... + w_n x_n + b \times 1$. Rozšíříme-li $\mathbf{w}$ na $\mathbf{w}' = (w_1, w_2, ..., w_n, b)^\top$ a $\mathbf{x}$ na $\mathbf{x}' = (x_1, x_2, ..., x_n, 1)^\top$, můžeme rovnici (1) identicky zapsat jako $\hat{y} = \mathbf{w}'^\top \mathbf{x}'$.

Jeden ze způsobů jak vyhodnocovat výsledky je měřit průměrnou čtvercovou chybu odhadu (MSE z anglického mean squared error), která počítá chybu následovně:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \frac{1}{n} \|\hat{\mathbf{y}} - \mathbf{y}\|^2 \tag{2}$$

Kde y je skutečná hodnota, $\hat{y}$ je predikovaná hodnota a n je počet pozorování.

Intuitivně vidíme, že se tato chyba bude rovnat 0, pokud $\hat{\mathbf{y}} = \mathbf{y}$. Jinými slovy, chyba se snižuje pokud Eukleidovská vzdálenost mezi predikcemi a cílovými hodnotami klesá. [12]

Abychom tento model mohli nazvat metodou strojového učení, zbývá nám popsat způsob, jakým najít parametry $\mathbf{w}$ tak, abychom minimalizovali chybu MSE. Toho můžeme docílit tak, že gradient[2], tedy vektor parciálních derivací, MSE s ohledem na parametry $\mathbf{w}$ položíme roven $\mathbf{0}$:

[2] Používáme gradient, jelikož $\mathbf{w}$ je **vektor**

$$
\begin{aligned}
\nabla_{\mathbf{w}} MSE &= \mathbf{0} \\
\Rightarrow \nabla_{\mathbf{w}} \frac{1}{n} \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 &= \mathbf{0} \\
\Rightarrow \frac{1}{n} \nabla_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 &= \mathbf{0} \\
\Rightarrow \nabla_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) &= \mathbf{0} \\
\Rightarrow \nabla_{\mathbf{w}} (\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}) &= \mathbf{0} \\
\Rightarrow (2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y}) &= 0 \\
\Rightarrow \mathbf{w} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}
\end{aligned}
\tag{3}
$$

Lineární regrese je poněkud jednoduchý a omezený učící algoritmus, nicméně se jedná o vhodný příklad toho, jak může metoda strojového učení fungovat.

3.0.2 Generalizace

Hlavním cílem ve strojovém učení je najít takovou metodu, která bude přesná i nad rámec dat, která byla použita při jejím trénování, tj. při hledání parametrů. Tato schopnost se nazývá generalizace. [12]

Generalizace modelu se dá měřit zavedením generalizační chyby, neboli chyby testovacích dat. Abychom mohli nějakým způsobem změřit schopnost modelu generalizovat na nová data, rozděluje se většinou dataset na:

1. Trénovací set - data, pomocí kterých minimalizujeme trénovací chybu ve snaze najít optimální parametry score funkce.
2. Testovací set - data, pomocí kterých se snažíme odhadovat generalizační ztrátu, neboli testovací chybu. Jako testovací set se většinou bere 10% - 30% z celého datasetu.

Loss funkce pro trénovací a testovací chybu je stejná, liší se tak pouze podkladová data, na kterých se loss funkce počítá.

V kontextu odhadování generalizace a dělení dat na trénovací a testovací se pří vyhodnocování modelu zavádí 2 pojmy:

1. Underfitting - říkáme, že model „underfittuje”, pokud není schopný docílit dostatečně nízké trénovací chyby.
2. Overfitting - říkáme, že model „overfittuje”, pokud model dosahuje velmi nízké trénovací chyby, zatímco testovací chyba je podstatně vyšší. V takovém případě si model pamatuje konkrétní případy a ztrácí schopnost generalizovat.

Cílem při volbě metody a hledání parametrů je najít kompromis mezi underfittingem a overfittingem.

Na rozdíl od klasických optimizačních problémů, metody strojového učení neznají funkci, kterou se snaží optimalizovat, jelikož trénovací chyba pouze zastupuje generalizační chybu při trénování.

Fixní rozdělení na trénovací a testovací dataset však může omezovat schopnost modelu učit se, jelikož vědomě snižujeme množství dat pro trénování a to tak, že při hledání parametrů nebereme do úvahy testovací data. Současně pokud zvolíme malý testovací set, přesnost odhadu generalizační ztráty se snižuje. To může komplikovat volbu vhodného modelu, jelikož nejsme schopni říci, která metoda funguje nejlépe pro daný dataset.

Jeden z nejběžnějších způsobů jak využít všechna data pro trénování a současně mít dostatečně velký testovací set za cenu větší výpočetní náročnosti je **křížová validace**. Nejčastěji se pak využívá **k-fold křížová validace**, kdy se dataset rozdělí na k nezávislých podmnožin. Proces hledání parametrů a vyhodnocení generalizační ztráty se pak opakuje k-krát, kdy v každém případě volíme vždy jednu podmnožinu jako testovací set a zbytek podmnožin jako trénovací set. Při volbě nejvhodnější metody se výsledná přesnost modelu, resp. velikost generalizační ztráty, může například průměrovat.

3.1 Deep Learning

Deep Learning je podmnožina metod strojového učení zahrnující zejména **umělé neuronové sítě**, což je sada algoritmů volně inspirovaná lidským mozkem. Ke skutečným neuronovým sítím mají ty umělé poněkud daleko, některé podobnosti se však nalézt dají. Tato kapitola je úvod do problematiky neuronových sítí, které jsou využity v praktické části.

Každá deep learningová metoda má v zásadě 2 části [17]:

1. Score funkce - funkce, která vstupní hodnoty parametricky mapuje na výstupní hodnoty. Jinými slovy funkce, která generuje predikce.
2. Loss funkce (někdy též Cost funkce, česky překládáno jako chybová funkce) - funkce, která kvatifikuje míru shody mezi výstupními hodnotami score funkce (predikcemi) a skutečnými (cílovými) hodnotami.

V předchozím případě byla score funkcí lineární regrese (viz rovnice (1)) a loss funkcí průměrná čtvercová chyba odhadu (viz rovnice (2)). Lineární regrese jak název napovídá řeší regresní problém. V případě, že se ovšem snažíme řešit klasifikační problém (např.

u obnosového pojištění), by výstupní hodnota score funkce měla být kategorická, nikoliv spojitá. Jednou z možností je rozšířit rovnici (1) o sigmoid funkci, značenou σ:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

Rovnice (1) se tak rozšíří na:

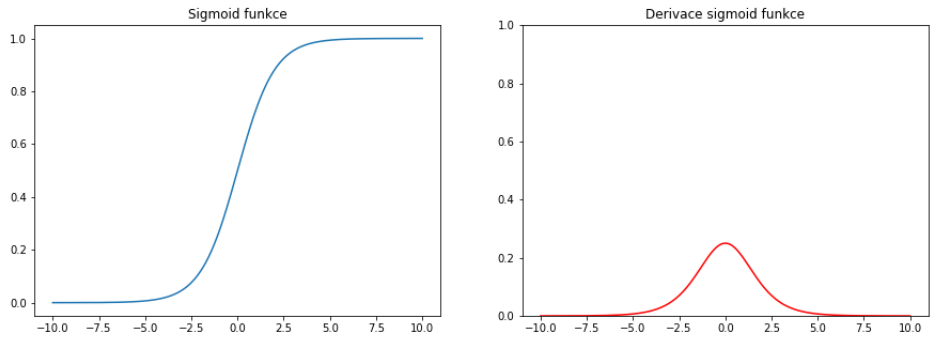
$$\hat{y} = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x})}} \tag{5}$$

Je možné si všimout, že rovnice (5) popisuje logistickou regresi. V odvětví deep learningu se ovšem častěji než logistická regrese používá pojem **sigmoid neuron**. Sigmoid funkce je vhodná pro binární klasifikace, jelikož má obor hodnot $\mathbb{H}(\sigma) = (0, 1)$. Výstupní hodnota se tak dá volně interpretovat jako binomické pravděpodobnostní rozdělení [12] neboli pravděpodobnost příslušnosti ke kategorii 1 [30]. V konkrétním případě např. obnosového pojištění, tj. binární klasifikace se 2 kategoriemi (pojistná událost nastala/nenastala) se dá výstupní hodnota vyložit jako pravděpodobnost s jakou pojistná událost nastala.

Sigmoid je diferencovatelná funkce s první derivací:

$$\begin{aligned} \frac{d}{dx}\sigma(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1+e^{-x}-1}{(1+e^{-x})}\frac{1}{(1+e^{-x})} = \\ = \left(1 - \frac{1}{(1+e^{-x})}\right)\frac{1}{(1+e^{-x})} = (1-\sigma(x))\sigma(x) \end{aligned} \tag{6}$$

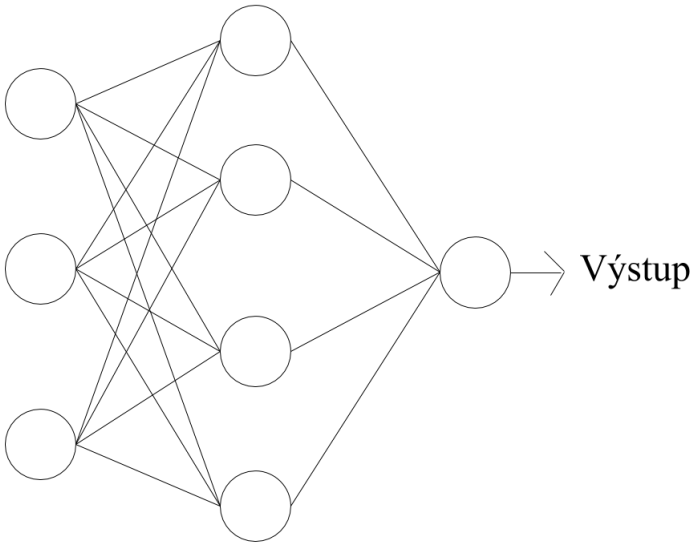
V upravené podobě má derivace sigmoid funkce $\sigma(x)$ tvar $(1 - \sigma(x))\sigma(x)$. V tomto tvaru je o něco intuitivnější, že blíží-li se hodnota $\sigma(x)$ ke krajní hodnotě 0 nebo 1, pak se hodnota derivace blíží 0 (viz obrázek (3)). V části o aktivačních funkcích bude blíže popsána praktická implikace této derivace.



Obrázek 3: Průběh sigmoid funkce v intervalu $< -10, 10 >$, též logistická křivka (vlevo). První derivace funkce sigmoid (vpravo). Zdroj: vlastní zpracování.

3.2 Feed-Forward neuronová síť

Následující sekce se pokusí vysvětlit architekturu umělé neuronové sítě. Následující obrázek představuje jednoduchou vizualizaci konceptu, který bude popsán:



Obrázek 4: Ukázka 2-vrstvé umělé neuronové sítě. Zdroj: vlastní zpracování.

Než bude koncept popsán obecně, mějme konkrétní příklad viz obrázek (4) s hodnotami $\mathbf{x} = (x_1, x_2, x_3)^\top$, které jsou vstupními hodnotami pro prostřední 4 neurony. Hodnota každého neuronu se spočítá nezávisle na sobě pomocí rovnice (5), tzn. $h_1 = \sigma(\mathbf{w}_{1h}\mathbf{x})$, $h_2 = \sigma(\mathbf{w}_{2h}\mathbf{x})$, apod. Výstupem pak je neuron, jehož vstupními hodnotami je vektor prostředních neuronů $\mathbf{h} = (h_1, h_2, h_3, h_4)$ a jeho hodnota se spočítá následovně $\hat{y} = \sigma(\mathbf{w}_{ho}\mathbf{h})$.

Obecně pak můžeme můžeme takovou síť zapsat následovně:

$$\hat{y} = \sigma(\mathbf{w}_{ho}(\sigma(\mathbf{W}_{ih}\mathbf{x}))) \tag{7}$$

Kde $\mathbf{x} \in \mathbb{R}^{n\times 1}$ je vektor vstupních dat, $\mathbf{W}_{ih} \in \mathbb{R}^{m\times n}$ je matice parametrů mezi vstupními hodnotami a prostřední vrstvou (nazývaná „skrytá vrstva")[3] a $\mathbf{w}_{ho} \in \mathbb{R}^{1\times m}$ je vektor parametrů mezi skrytou vrstvou a výstupní hodnotou[4].

Role sigmoid funkce σ v rovnici (7) je důležitá, jelikož zajišťuje nelinearitu funkce. Bez ní by se rovnice (7) dala zapsat následovně[5]: $\hat{y} = \mathbf{w}_{ho}\mathbf{W}_{ih}\mathbf{x}$. Pakliže v takovém případě položíme $\mathbf{w} = \mathbf{w}_{ho}\mathbf{W}_{ih}$, dostaneme[6] $\hat{y} = \mathbf{w}_{ho}\mathbf{W}_{ih}\mathbf{x} = \mathbf{w}\mathbf{x}$, čož je identický zápis lineární

[3] ih z anglického input - hidden
[4] ho z anglického hidden - output
[5] K této rovnici se dostaneme, odstraníme-li z rovnice (7) funkci σ
[6] Plyne z asociativity maticového násobení.

regrese (viz rovnice (1)).

Díky nelineární funkci σ pak můžeme formovat vícevrstvou neuronovou síť tak, že koncept rozšíříme o více skrytých vrstev, čímž se celá síť stane hlubší. Proto se celému konceptu říká hluboké učení, resp. deep learning, kterým se převážně zabývá tato práce. Cybenko v roce 1989 dokázal, že neuronová síť s alespoň jednou skrytou vrstvou a konečným počtem neuronů zvládne aproximovat libovolnou spojitou funkci [8]. V části o aktivačních funkcích budou popsány další nelineární funkce.

3.3 Loss funkce

Druhým klíčovým komponentem neuronových sítí je loss funkce (též chybová funkce), která kvantifikuje míru shody mezi predikcemi a skutečnými cílovými hodnotami.

Jedna z nejběžnějších loss funkcí již byla představena, konkrétně průměrná čtvercová chyba odhadu MSE (viz rovnice (2)), která je běžně používaná při modelování spojitých hodnot, tj. u regresních problémů.

V případě modelování kategorických hodnot je nejběžnější loss funkcí cross-entropy[7]:

$$H(p,q) = -\sum_{x} p(x) log(q(x)) \tag{8}$$

Kde p je skutečná distribuce a q je predikovaná distribuce proměnné x. Skutečnou distribuci $p(x)$ je nutno interpretovat jako distribuci, kde celá pravděpodobností míra je na pozici skutečné kategorie. Jinými slovy $p(x)$ představuje vektor obsahující samé 0 s vyjímkou 1 na pozici i, pakliže se jedná o i-tou kategorii ($p(x) = (0, 0, ..., 1, ..., 0)$)[17].

Pakliže rozepíšeme rovnici (8) pro případ binární klasifikaci, dostaneme:

$$H(y,\hat{y}) = -(y * log(\hat{y}) + (1-y) * log(1-\hat{y})) \tag{9}$$

Kde $y \in \{0,1\}$ je binární cílová hodnota a $\hat{y} \in (0,1)$ je predikovaná hodnota (např. výstupní hodnota posledního sigmoid neuronu).

Pro pochopení rovnice (9), mějme konkrétní příklad, kdy skutečná hodnota $y = 1$ a predikovaná hodnota $\hat{y} = 0.01$. V takovém případě se predikce neshoduje se skutečnou hodnotou. Dosadíme-li do rovnice (9), dostaneme: $H(y,\hat{y}) = -(1 * log(0.01) + (1-1) * log(1-0.01)) = -log(0.01) \doteq 4.65$. S rostoucí mírou shody predikce a skutečné hodnoty bude tato chyba klesat.

Předchozí dvě části popsaly, jakým způsobem parametricky mapovat vstupní hodnoty na výstupní predikce (score funkce) a jakým způsobem posuzovat shodu (resp. chybovost) této

[7] *log* symbolizuje přirozený logaritmus

predikce (loss funkce). Následující část se zaměří na třetí a poslední část - optimalizaci, která popisuje, jakým způsobem najít optimální parametry score funkce vzhledem k loss funkci.

3.4 Optimalizace

Cílem optimalizace je najít takovou sadu parametrů $\mathbf{W}$, která bude minimalizovat loss funkci. V případě neuronových sítí však nelze optimální parametry najít tak snadno jako v případě lineární regrese (viz rovnice (3)).

Téměr všechny metody deep learningu pro optimalizaci využívají algoritmus stochastic gradient descent, který je nadstavbou metody gradient descent (česky též gradientní sestup).

3.4.1 Gradient descent

Předpokládejme, že máme funkci $y = f(x)$, kde x a y jsou reálná čísla a cílem je minimalizovat funkci f. Označme $f'(x)$ derivaci funkce f. Derivace je vhodná, jelikož napovídá jak změnit hodnotu x tak, abychom snížili hodnotu y. Toho můžeme docílit posunováním x v malých krocích v opačném směru derivace, neboli pro dostatečně malé $\gamma > 0$:

$$x' = x - \gamma f'(x) \Rightarrow f(x') \leq f(x) \tag{10}$$

Pro funkce s více parametry (označme $\mathbf{x}$) můžeme gradient descent zapsat následovně:

$$\mathbf{x}' = \mathbf{x} - \gamma \nabla_{\mathbf{x}} f \Rightarrow f(\mathbf{x}') \leq f(\mathbf{x}) \tag{11}$$

Kde $\gamma > 0$ a $\nabla_{\mathbf{x}} f$ symbolizuje gradient parametrů $\mathbf{x}$ vzhledem k funkci f. Gradient je vektor parciálních derivací, tj. i-tý prvek gradientu odpovídá $\frac{\partial}{\partial x_i} f(\mathbf{x})$. Rovnice (11) platí za předpokladu, že funkce f je spojitá a diferencovatelná v okolí bodu x. Pakliže budeme krok z rovnice (11) opakovat, bude funkce f konvergovat ke svému minimu.

Parametr γ se v kontextu strojového učení často nazývá **learning rate**.

Současně je nutné zmínit, že gradient descent je iterativní algoritmus a na začátku učení jsou parametry neuronové sítě náhodně inicializované. V pozdější části této práce jsou strategie počáteční inicializace blíže rozebrány.

3.4.2 Stochastic gradient descent

Stochastic gradient descent (SGD) je stochastický odhad gradientního sestupu. Jelikož strojové učení (a zejména hluboké učení) vyžaduje velké množství dat, aby bylo schopné generalizovat, bylo by neefektivní z pohledu výpočetního výkonu upravovat parametry napříč celým data setem. S rostoucím množstvím dat by navíc každý jednotlivý krok úpravy parametrů trval déle.

SGD odhaduje gradient pomocí vzorku ze všech dat. Formálně, v každém kroku vybíráme podmnožinu dat z rovnoměrně rozděleného trénovacího setu a opakujeme gradientní sestup viz rovnice (11). Počet prvků této podmnožiny pak stanovíme fixní a nemění se ani s rostoucím množstvím trénovacích dat. [12] Např. můžeme brát podmnožiny o velikosti[8] 32, na této podmnožině spočítáme chybu (např. pomocí rovnice (2) nebo (8)), provedeme úpravu parametrů (rovnice (11)) a postupně opakujeme pro všechny podmnožiny.

Mezi hlavní výhody SGD patří: [21]

1. SGD je většinou daleko rychlejší než gradient descent
2. SGD často najde lepší řešení
3. SGD umožňuje sledovat změny (např. mění-li se v čase distribuce vstupních dat)

Optimizační proces nezaručuje nalezení globálního minima, ani lokálního minima, nicméně často najde dostatečně nízkou hodnotu v krátkém čase. [12]

3.4.3 Backpropagation

V případě SGD u neuronových sítí vyvstává otázka jak spočítat gradient pro jednotlivé parametry. Mějme funkci $f(\mathbf{W}, \mathbf{x})$, kde $\mathbf{x}$ je vektor vstupních dat, $\mathbf{W}$ jsou parametry funkce a potřebujeme spočítat gradient funkce f pro $\mathbf{W}$, neboli $\nabla_{\mathbf{W}} f$. V případě neuronových sítí funkce f odpovídá loss funkci, $\mathbf{x}$ trénovacím datům a $\mathbf{W}$ parametrům neuronové sítě. Trénovací data jsou ovšem daná a gradient se tak počítá pouze pro parametry neuronové sítě. [17]

Feed-forward neuronové sítě ze vstupního vektoru $\mathbf{x}$ přes jednotlivé neurony skrytých vrstev spočítají výstupní hodnoty $\hat{\mathbf{y}}$. Při trénování se pak spočítá chyba $L(\hat{\mathbf{y}}, \mathbf{y})$. Backpropagation (backward propagation of errors) umožňuje odzadu spočítat gradienty vzhledem k loss funkci L pomocí řetězového pravidla derivace.

[8] Často se jako velikost podmnožiny volí mocnina 2.

Řetězové pravidlo derivace slouží pro výpočet derivace složené funkce. V nejjednodušším případě mějme $x \in \mathbb{R}$ a funkce $f : \mathbb{R} \to \mathbb{R}$ a $g : \mathbb{R} \to \mathbb{R}$. Je-li $y = f(x)$ a $z = g(y) = g(f(x))$, pak dle řetězového pravidla:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{12}$$

Tento koncept se pak přirozeně z jednorozměrné podoby dá rozvést. Mějme $\mathbf{x} \in \mathbb{R}^n$, funkce $f : \mathbb{R}^n \to \mathbb{R}^m$ a $g : \mathbb{R}^m \to \mathbb{R}$. Je-li $\mathbf{y} = f(\mathbf{x})$ a $z = g(\mathbf{y}) = g(f(\mathbf{x}))$, pak:

$$\begin{aligned} \frac{\partial z}{\partial x_i} &= \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i} \\ \Rightarrow \nabla_{\mathbf{x}} z &= (\frac{\partial \mathbf{y}}{\partial \mathbf{x}})^\top \nabla_{\mathbf{y}} z, \end{aligned} \tag{13}$$

kde $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ je Jacobiho matice tvaru $n \times m$ funkce g [12].

Jakmile pomocí backpropagation spočítáme gradienty jednotlivých parametrů, můžeme upravit parametry pomoci SGD (tj. provedeme krok viz rovnice (11) na podmnožině z trénovacích dat). Pakliže zvolíme learning rate γ dostatečně malý, bude chyba postupně klesat až funkce f zkonverguje k nějakému minimu.

3.4.4 Adam

Existuje několik rozšíření pro SGD jako například AdaGrad [10] nebo RMSProp [33]. V praktické části této práce však je použita rozšířená verze SGD nazvaná zkráceně Adam (z anglického adaptive moment estimation) [19].

Adam rozšiřuje SGD o exponenciální pohyblivé průměry gradientu m_t a čtvercového gradientu v_t. Tyto pohyblivé průměry slouží jako odhady prvního momentu (průměru) a druhého momentu (rozptylu) gradientu. Na začátku jsou tyto momenty inicializovány pomocí vektoru 0. Označme $g_t := \nabla_{\mathbf{x}_{t-1}} f$ jako gradient funkce f v t-tém kroku s parametry $\mathbf{x}_{t-1}$ v $(t-1)$-ním kroku (tj. před úpravou parametrů), pak rovnice Adam vypadá následovně: [19]

$$
\begin{aligned}
\mathbf{m_t} &= \beta_1 \mathbf{m_{t-1}} + (1-\beta_1)\mathbf{g_t} \\
\mathbf{v_t} &= \beta_1 \mathbf{v_{t-1}} + (1-\beta_1)\mathbf{g_t}^2 \\
\widehat{\mathbf{m_t}} &= \frac{\mathbf{m_t}}{1-\beta_1{}^t} \\
\widehat{\mathbf{v_t}} &= \frac{\mathbf{v_t}}{1-\beta_2{}^t} \\
\mathbf{x}_t &= \mathbf{x}_{t-1} - \gamma \frac{\widehat{\mathbf{m_t}}}{\widehat{\mathbf{v_t}} + \epsilon},
\end{aligned}
\tag{14}
$$

kde γ je learning rate, obdobně jako v rovnici (11). β_1, β_2 jsou parametry exponenciální míry poklesu u odhadu 1. a 2. momentu, většinou se volí $\beta_1 = 0.9$ a $\beta_2 = 0.999$. Parametr ϵ se přidává z praktických důvodů, aby se předešlo dělení 0 a volí se velmi malé kladné číslo, např. 10^{-8}.

Motivací k zavedení momentů při optimalizaci je snaha adaptivně snižovat výraz upravující parametry[9], jakmile se funkce přibližuje k optimu a snižují se tak gradienty. Pakliže by tento výraz nebyl dostatečně malý blízko minima, hodnota funkce by se pohybovala kolem minima a přestala by se k minimu přibližovat.

Přestože v mnoha případech Adam dospěje k lepší sadě parametrů s nižší hodnotou loss funkce a současně rychleji během trénování, nedávný výzkum diskutuje, že SGD dosahuje lepších generalizačních výsledků u reálných dat [18]. Jednoznačně určit, který z těchto algoritmů je lepší tak nelze.

3.5 Aktivační funkce

V části o feed-forward neuronové síti byl představen koncept skryté vrstvy společně se sigmoid funkcí (viz rovnice (4)), která mimo jiné zajišťovala nelinearitu celé funkce. Tato část popisuje **aktivační funkce**, mezi něž patří již zmíněná funkce sigmoid. Každá aktivační funkce (anglicky též non-linearity) provádí fixní matematickou operaci na jednom čísle, v případě vektoru pak na každém prvku zvlášť s cílem zajisti nelinearitu funkce (neuronové sítě) [17].

Jak bylo nastíněno v předchozí části, **sigmoid** funkce má obor hodnot $\mathbb{H}(\sigma) = (0, 1)$, jinými slovy „stlačí" reálné číslo mezi 0 a 1. Historicky byla sigmoid funkce oblíbená, jelikož se dá interpretovat jako míra vzruchu u neuronu, od žádného vzruchu (0) až po vzruch s maximální intenzitou (1). V současnosti se od sigmoid funkce u skrytých vrstev ustupuje a to zejména z následujícího důvodu: [17]

Sigmoid funkce stlačuje gradienty a zpomaluje učení. Nežádoucí vlastností sig-

[9] V rovnici (14) je tímto výrazem $\frac{\widehat{\mathbf{m_t}}}{\widehat{\mathbf{v_t}}+\epsilon}$

moid funkce je, že hodnota derivace je téměř nulová u hodnot blízkých krajním hodnotám 0 nebo 1 (viz obrázek (3)) a současně nejvyšší hodnota derivace je pouze 0,25. Připomeňme, že během backpropagation algoritmu se uplatňuje řetězové pravidlo, které násobí gradient sigmoid funkce s gradientem jejího výstupu (viz rovnice (13)). Jsou-li však hodnoty gradientu velmi nízké, bude pak v důsledku násobení celý gradient blízký nulovému vektoru, bude potlačen signál a výrazně se zpomaluje úprava parametrů (viz rovnice (11)). I v nejlepším případě se pak při backpropagation gradient zmenší alespoň o faktor 4. V konečném důsledku pak trvá déle než celá funkce zkonverguje ke svému minimu. Tento efekt se někdy nazývá vytrácející se gradient (z anglického vanishing gradient) a jedná se o palčivý problém zejména u více-vrstvých neuronových sítí.

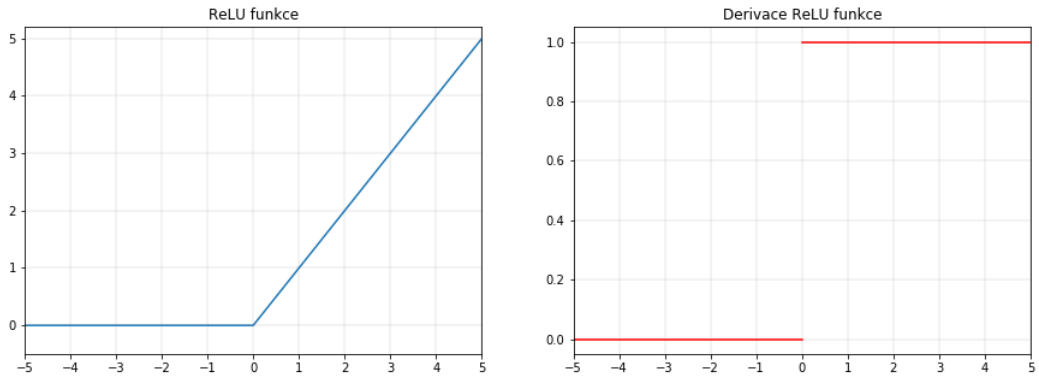
Z tohoto důvodu se sigmoid funkce téměř nepoužívá jako aktivační funkce u skrytých (prostředních) vrstev neuronové sítě. I nadále je však hojně využívaná jako aktivační funkce výstupní (poslední) vrstvy v případě modelování proměnné y s Bernoulliho distribucí, tedy predikce $P(y = 1|\mathbf{x})$ [12], například při binární klasifikaci.

V současnosti nejrozšířenější aktivační funkcí je **Rectified Linear Unit**, neboli **ReLU**, která je definována následovně:

$$ReLU(x) = max(0, x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{15}$$

Funkce ReLU je diferencovatelná na intervalu $(-\infty, 0) \cup (0, \infty)$ s první derivací:

$$\frac{d}{dx} ReLU(x) = \mathbb{1}(x > 0) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases} \tag{16}$$



Obrázek 5: Průběh ReLU funkce v intervalu $< -5, 5 >$ (vlevo). První derivace ReLU funkce (vpravo). Zdroj: vlastní zpracování.

Na rozdíl od sigmoid funkce, ReLU výrazně zrychluje konvergenci optimalizačního algoritmu (např. SGD), jelikož nestlačuje gradienty během backpropagation. [20] Další

výhodou ReLU funkce je, že neobsahuje výpočetně náročné matematické operace (exponenciálu apod.). Pouze porovnává číslo s prahovou hodnotou 0. [17]

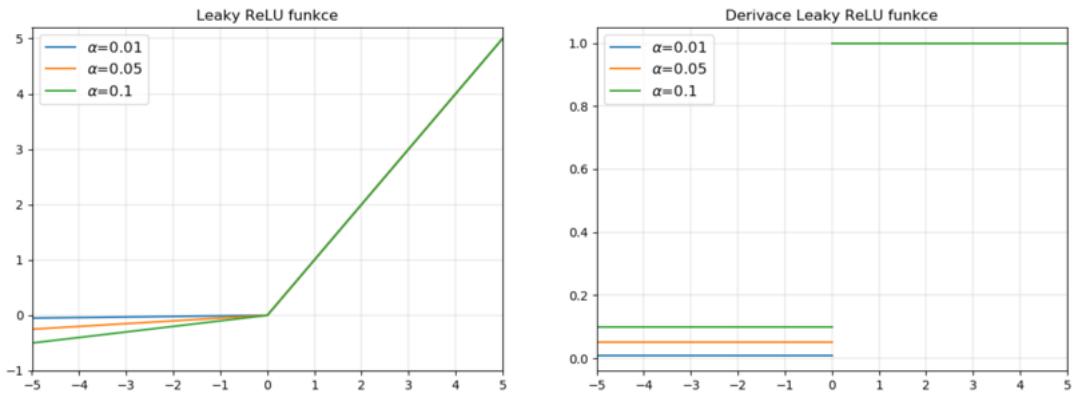
Nevýhodou ReLU je, že v některých případech může v důsledku velkého gradientu dojít k zásadní úpravě parametrů, která v důsledku potlačí většinu hodnot pod prahovou hodnotu 0 a neuron se tak nikdy neaktivuje. Hodnota takového neuronu už bude stále nulová a takový neuron tzv. „umře". To bývá problém zejména v případě, kdy parametr learning rate (tj. γ v rovnici (11)) je moc velký. Při správné volbě hyperparametrů (viz. následující sekce) se však tomuto efektu dá předejít.

Leaky ReLU je jeden ze způsobů, jak se dá zabránit efektu „umírajících"ReLU neuronů, stlačujících hodnotu na 0. Místo 0, má Leaky ReLU malý záporný sklon (např. 0.01), je-li $x < 0$ [17]. Matematicky lze zapsat následovně:

$$LeakyReLU(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x \geq 0)(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases} \tag{17}$$

V rovnici (17) je α malá konstanta. Obdobně jako ReLU je Leaky ReLU diferencovatelná na intervalu $(-\infty, 0) \cup (0, \infty)$ s první derivací:

$$\frac{d}{dx} LeakyReLU(x) = \begin{cases} 1 & x > 0 \\ \alpha & x < 0 \end{cases} \tag{18}$$



Obrázek 6: Průběh Leaky ReLU funkce v intervalu $< -5, 5 >$ pro $\alpha \in \{0.01, 0.05, 0.1\}$ (vlevo). První derivace Leaky ReLU funkce (vpravo). Zdroj: vlastní zpracování.

Alternativní verzí Leaky ReLU je PReLU (Parametric ReLU) [13], kde parametr α je součástí parametrů upravovaných během učení pomocí optimalizačního algoritmu a backpropagation. Matematický zápis je identický jako u Leaky ReLU, s tím rozdílem, že α není konstanta, ale parametr.

V některých případech Leaky ReLU a PReLU fungují lépe než ReLU, avšak výsledky nejsou konzistentní a často se v rozchází [17]. ReLU tak je nadále pravděpodobně nejběžnější

aktivační funkcí navzdory zdánlivým výhodám, které nabízí Leaky ReLU a PReLU. V praktické části této práce se však parametrická verze ReLU funkce osvědčila a proto je v této práci zmíněna.

Existuje několik dalších běžně používaných aktivačních funkcí, jako jsou například hyperbolický tangens (tanh) nebo ELU [7], kterými se však tato práce nezabývá. Současně jsou aktivační funkce i nadále předmětem výzkumu [29].

3.6 Hyperparametry

Parametry neuronové sítě nazýváme ty hodnoty neuronové sítě, které se upravují pomocí optimalizačního algoritmu a backpropagation během učení. Na rozdíl od toho, hyperparametry nazýváme takové parametry funkce, které se během učení nemění a většinou jsou fixní pro daný model. Hyperparametry jsou například:

- Počet vrstev neuronové sítě
- Počet jednotek (neuronů) v konkrétní vrstvě neuronové sítě
- Learning rate γ
- Velikost podmnožiny (vzorku dat) pro dílčí krok SGD - běžně o tomto parametru mluvíme jako o **batch size**
- Volba aktivační funkce
- Volba optimalizačního algoritmu - např. SGD, Adam apod.
- Dílčí parametry konkrétního optimalizačního algoritmu - např. β_1 a β_2 u Adam

Tyto hyperparametry definují architekturu konkrétní neuronové sítě a jejího učení. Důležitým hyperparametrem, který dosud nebyl popsán je počet tzv. **epoch**. SGD upravuje parametry po jednotlivých podmnožinách (anglicky též batch). Dílčí kroky můžeme nazývat iterace. Jednou epochou pak je kompletní průběh iterací přes všechna data. Pro příklad mějme dataset se 100 záznamy a zvolme batch size (velikost podmnožiny) o velikosti 20. Pak optimalizační algoritmus (např. SGD) provede 5 iterací po 20 záznamech, na jejichž konci proběhne úprava parametrů. Jednou epochou nazýváme v tomto případě všech 5 iterací, jelikož v tu chvíli model vyhodnotí každý záznam z datasetu alespoň jednou.

Volba batch size, počtu epoch a learning rate je při učení stěžejní, jelikož se jimi snažíme balancovat již zmíňovaný kompromis mezi underfittingem a overfittingem. Zvolíme-li například nízký learning rate a malý počet epoch, bude se model učit pomalu a pravděpodobně v

takovém případě bude „underfitovat". Na druhou stranu, zvolíme-li velký počet epoch, bude ke konci neuronová síť modelovat trénovací data tak přesně až ztratí schopnost generalizovat na jiná data.

3.7 Inicializace parametrů

Optimalizační algoritmy postavené na gradientním sestupu jsou iterativní a potřebují výchozí bod, ze kterého můžou postupně upravovat parametry. Tento výchozí bod však může ovlivnit, zda algoritmus zkonverguje k nějaké minimální hodnotě a pokud ano, jak rychle učením zkonverguje. [12]

Intuitivně se nabízí možnost inicializovat všechny parametry pomocí 0. To ovšem zdaleka není rozumná strategie, jelikož v takovém případě všechny neurony spočítají stejnou výstupní hodnotu, která povede ke stejnému gradientu a všechny parametry tak budou upravovány identicky. Jsou-li tedy všechny parametry inicializovány stejnou hodnotou, pak mezi nimi nenastává žádná asymetrie. [17]

Cílem počáteční inicializace tedy je narušit asymetrii mezi jednotlivými neurony. Současně chceme, aby hodnoty parametrů byly blízké 0. [12] Jednou z možností je inicializovat parametry náhodně s normálním rozdělením s průměrem 0 a nějakou malou směrodatnou odchylkou (např. 0.01). Místo normálního rozdělení je možné zvolit i rovnoměrné rozdělení. Empiricky má volba rozdělení v důsledku jen malý vliv na finální výsledek. [17]

Existuje několik heuristik jak zvolit rozsah počátečních parametrů s m vstupní dimenzí a n výstupní dimenzí. U feed-forward neuronových sítí je asi nejběžnější tzv. **normalizovaná inicializace** (z anglického normalized initialization) podle Glorota a Bengia [11], která inicializuje počáteční parametry z rovnoměrného rozdělení následovně:

$$W_{i,j} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right) \tag{19}$$

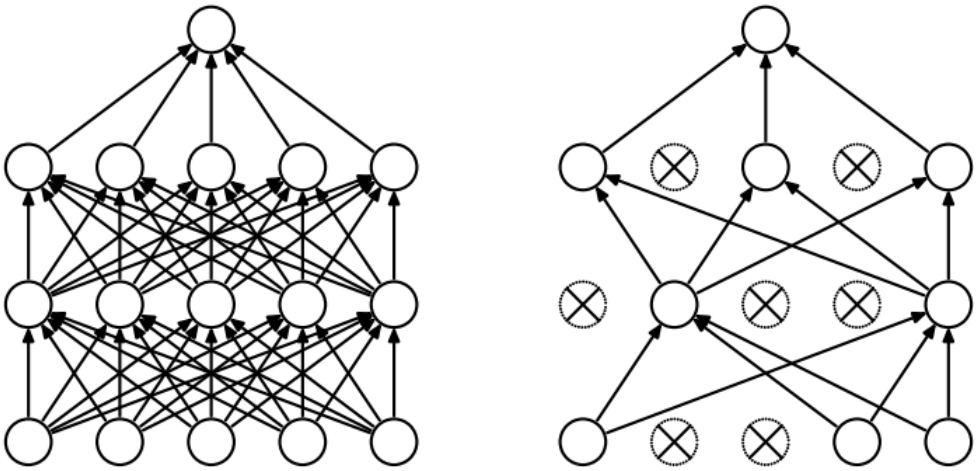
Cílem této heuristiky je inicializovat váhy tak, aby všechny vrstvy měly stejný rozptyl hodnot i gradientů [11]. Rovnice (19) byla odvozena s předpokladem, že se neuronová síť skládá výhradně z maticového násobení (tzn. neobsahuje žádné aktivační funkce). Přestože je tento předpoklad u neuronových sítí porušen, se tento způsob inicializace vah osvědčil. [12]

3.8 Regularizace

Hlavním cílem ve strojovém učení je docílit generalizace, tj. přesné fungování modelu i u dat, která nebyla využita při trénování. Za regularizaci se dá považovat jakákoliv modifikace učícího algoritmu, jejíž cílem je snížit generalizační ztrátu avšak ne trénovací ztrátu [12]. Existuje několik regularizačních metod, tato práce však krátce popisuje pouze 2 způsoby - **Dropout** a **Ensembling**.

Dropout

Dropout je regularizační technika, která umožňuje trénovat podsítě dané neuronové sítě tak, že během trénování náhodně ignoruje některé neurony a jejich spoje [32]. Jinými slovy, během trénování neuron zůstane aktivní (je ponechána jeho hodnota) jen s určitou pravděpodobností p, v opačném případě je jeho hodnota položena rovno 0. [17] Tato metoda se provádí pouze během trénování, během predikce pak zůstávají všechny neurony aktivní (pravděpodobnost $p = 1$).



Obrázek 7: Standardní neuronová síť se 2 skrytými vrstvami (vlevo). Užší neuronová síť po aplikaci dropout metody, neurony s křížkem jsou ignorovány (vpravo). Zdroj: [32]

Ensembling

Ensembling je metoda, která trénuje několik různých modelů zvlášť a následně kombinuje predikce jednotlivých modelů do jedné ve snaze docílit přesnější predikce. [12] Existuje více způsobů, jak několik modelů kombinovat. Pravděpodobně nejrozšířenější je tzv. **Bagging** metoda (z anglického bootstrap aggregation) [3]. U bagging metody jsou predikce jednotlivých modelů jednoduše průměrovány do výsledné predikce.

Důvod, proč průměrování více modelů funguje je ten, že jednotlivé modely většinou nedělají stejné chyby u testovacího data setu [12].

3.9 Shrnutí a obecné poznámky k části o strojovém učení

Neuronová síť se skládá ze 2 částí - score funkce a loss funkce. U score funkce předem stanovujeme počet vrstev sítě, počet neuronů v rámci každé vrsty, volíme aktivační funkci a další hyperparametry. Parametry sítě inicializuje náhodně z rovnoměrného nebo normálního rozdělení s malým rozsahem. Při volbě loss funkce je důležitý typ úlohy, kterou řešíme. V případě regresní úlohy se většinou volí průměrná čtvercová chyba odhadu, u klasifikačních úloh se pak ve většině případů používá cross entropy. Pro optimalizaci existuje několik podob SGD, v současnosti jednou z nejběžnějších je Adam. Aby se při učení předešlo overfittingu, může neuronová síť obsahovat Dropout. Pro finální predikci se dále může využít vícero modelů v rámci ensemble metody, případně i ve spojení s K Fold křížovou validací. Pro libovolnou modelovací úlohu jsou metody deep learningu vhodné, jelikož jakkákoliv neuronová síť s alespoň jednou skrytou vrstvou a konečným počtem neuronů dokáže aproximovat libovolnou spojitou funkci [8].

Deep learning se osvědčil i v dalších případech, například konvoluční neuronové sítě pro zpracování obrazu [13] [20] nebo rekurentní neuronové sítě pro zpracování přirozeného jazyka [25] [34] a modelování časových řad [31].

4 Modelový příklad

Praktická část této práce se zabývá modelováním severity škod u dopravních nehod motorových vozidel pomocí neuronových sítí. Modelový příklad této práce byl vypracován v programovacím jazyce Python [28], zejména pomocí programovací knihovny PyTorch [27] [26]. Zdrojový kód k této práci je veřejně dostupný na adrese https://github.com/danielhavir/claims-severity-model.

4.1 Dataset

Pro modelový příklad v této práci byla použita data od americké pojišťovací společnosti AllState. Data byla zveřejněna v rámci soutěže *AllState Claims Severity* hostované na platformě Kaggle [14].

Online projekt Kaggle se zaměřuje na úkoly náročné na práci s daty. Cílem je dospět k prognóze, která je lepší než základní výchozí prognóza společnosti. Nejednalo se o první soutěž vypsanou pojišťovací společností AllState. V jedné z předchozích soutěží, kde cílem byla predikce pojištění odpovědnosti za škodu na základě charakteristik pojištěného vozidla, pak vítězný model byl o 270% lepší než výchozí model společnosti [4] [16]. Účastníci soutěží dle předem stanovené metriky na datech zveřejněných organizacemi [35]. Odměny jsou většinou finančního rázu, od nižších jednotek tisíc dolarů až po jednotky milionů dolarů. V případě *AllState Claims Severity* se však jednalo o soutěž náborovou.

Ve snaze vyvinout lepší vyhodnocování predikce škod a jejich rozsahu organizovala americká pojišťovna zmiňovanou soutěž. Predikce se týkala škod způsobených dopravními nehodami. V jádru soutěže byla snaha zodpovědět otázku "Jak závážný je rozsah škod pojistné události?". Jak bylo popsané v pojistné části této práce, tato otázka je důležitá pro stanovení výše RBNS rezerv.

Dataset obsahuje 131 proměnných a byl předem rozdělen na trénovací a testovací set. Trénovací set obsahuje 188 318 záznamů včetně cílové hodnoty. Testovací set obsahuje 125 546 záznamů a cílové hodnoty k němu nejsou známy. Ze 131 proměnných je 116 kategorických a 14 spojitých. Data jsou částečně anonymizovaná. Jednotlivé proměnné jsou označeny $cat1, cat2$ až $cat116$ u kategorických proměnných a $cont1, cont2$, až $cont14$ u proměnných spojitých. Můžeme předpokládat, že všech 131 proměnných je relevantních pro daný problém, tj. kvalifikovanou predikci rozsahu škod.

Soutěž probíhala v 2.polovině roku 2016. V čase psaní této práce je soutěž již vyhodnocena. Vítězné řešení bylo popsáno v diskuzi, avšak nebylo publikováno. [15]

4.2 Vyhodnocovací metrika

Jako metrika pro vyhodnocení nejúspěšnějšího modelu byla pojišťovnou zvolena střední absolutní chyba odhadu (Mean Absolute Error, MAE):

$$MAE = \frac{\sum_i^n |y_i - x_i|}{n} = \frac{\sum_i^n |e_i|}{n} \tag{20}$$

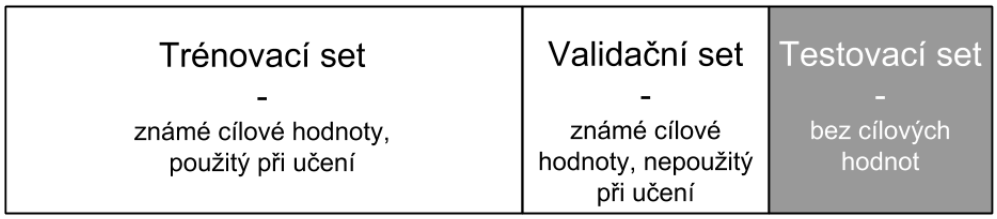
Kde y_i je skutečná výše škod, x_i je predikovaná výše škod a n je počet predikovaných událostí.

Vyhodnocování testovacího setu zaštiťovala společnost Kaggle a probíhalo na jejich infrastruktuře na základě odeslaného souboru s identifikačním číslem testovacího záznamu a predikovanou hodnotou. Skutečné cílové hodnoty pro testovací set nejsou známy.

Výchozí hodnota MAE metriky zadaná pořadatelem soutěže pro daný testovací set byla **1227.74973**.

4.3 Modely

Jelikož data byla již předem rozdělena na trénovací a testovací set, bylo třeba za účelem odhadu generalizační (testovací) chyby vytvořit podmnožinu dat z trénovacího souboru, tzv. validační set. Máme tak 3 podmnožiny dat viz obrázek:



Obrázek 8: Zjednodušená vizualizace rozdělení na trénovací, validační a testovací set. Zdroj: vlastní zpracování.

Při volbě nejlepšího modelu byl nejprve fixně oddělen validační set o velikosti 10% z trénovací setu anotovaných dat[10].

Kategorické proměnné byly převedeny do tzv. „dummy"proměnných, tzn. i-tá kategorie proměnné s k kategoriemi byla reprezentována vektorem nul s vyjímkou 1 na i-té pozici.

[10] 10% ze 188 318 záznamů je 18 832 záznamů. Zmenšený trénovací set tak obsahoval 169 486 záznamů.

Pro demonstraci konceptů z části o strojovém učení jsou dále popsány 3 modely a poté jejich výsledek na daném datasetu.

Model 1: 2-vrstvá feed-forward neuronová síť

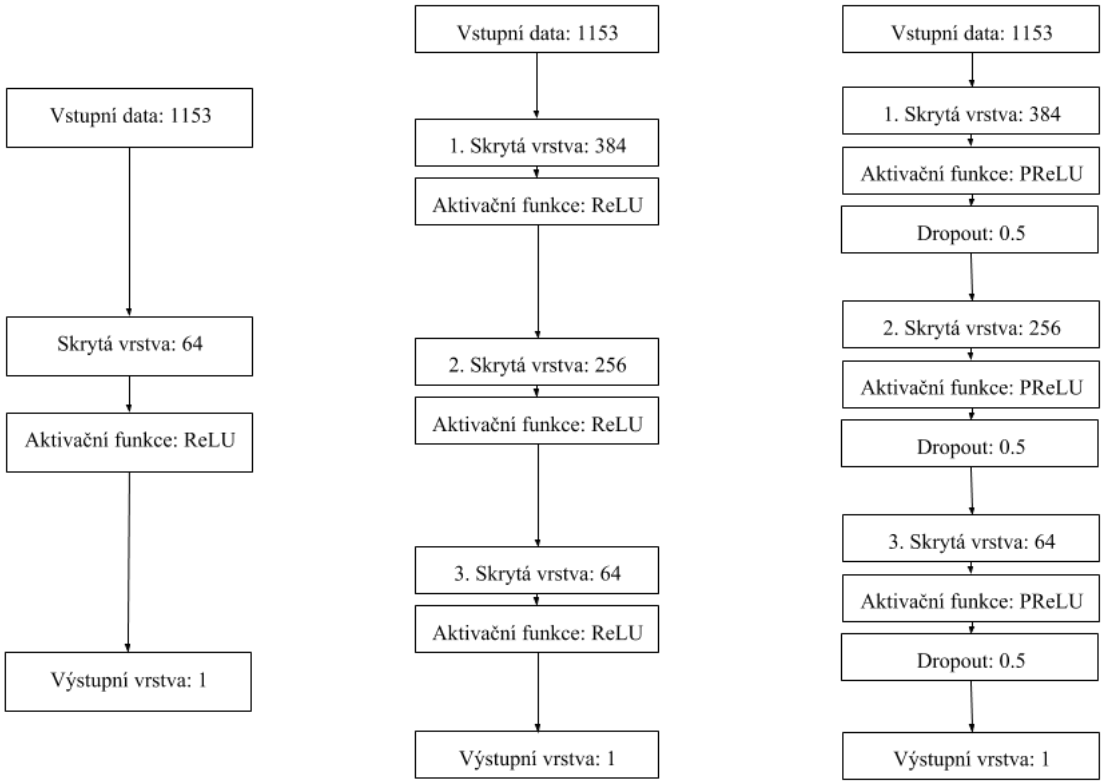
Při experimentech byl výchozí model 2-vrstvá neuronová síť s 1 skrytou vrstvou, kterou následovala ReLU funkce a výstupní vrstva. Jelikož se jedná o regresní úlohu, po výstupní vrstvě již nebyla žádná aktivační funkce. Skrytá vrstva obsahovala 64 jednotek. Žádná regularizace nebyla aplikována.

Model 2: 4-vrstvá feed-forward neuronová síť

Přirozenou nádstavbou výchozího modelu je přidat více skrytých vrstev a „prohloubit"ho tak. Model 2 tak obsahoval 3 skryté vrstvy, každá následovaná ReLU aktivační funkcí. Skryté vrstvy obsahovaly 384, 256 a 64 jednotek.

Model 3: 4-vrstvá feed-forward neuronová síť, Dropout, PReLU

Model 2 se dá dále rozšířit o Dropout. Počet vrstev i jednotek zůstal stejný, místo ReLU jakožto aktivační funkce však byla použita PReLU (Parametric ReLU) aktivační funkce.



Obrázek 9: Model 1 (vpravo), Model 2 (uprostřed), Model 3 (vlevo). (Číslo popisuje dimenzionalitu) Zdroj: vlastní zpracování.

4.3.1 Trénovací hyperparametry

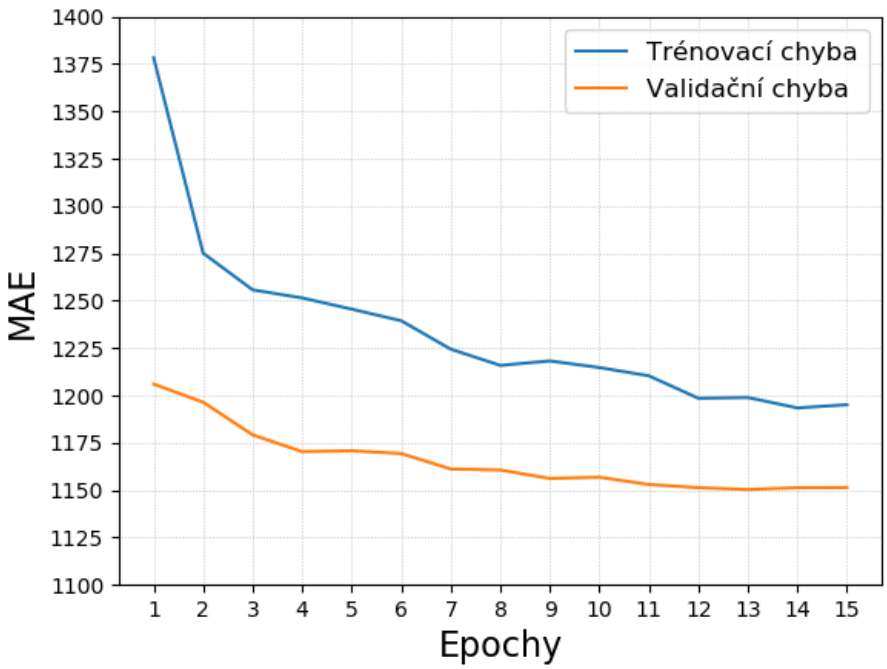
U všech modelů byly pro trénování stanoveny následující hyperparametry:

- **Batch size** - 1024
- **Počáteční learning rate** γ - 0.003
- **Počet epoch** - 15

Současně hodnota learning rate klesla o polovinu každých 5 epoch. Pakliže byl jako optimalizační algoritmus použit Adam, hyperparametry β_1 a β_2 byly ponechány na výchozí hodnotě, tj. $\beta_1 = 0.9$ a $\beta_2 = 0.999$.

Různé hodnoty parametrů byly testovány během experimentů. Po 15. epoše trénovací a validační chyba klesala jen zanedbatelně. Vyšší learning rate pak urychlil trénování v počátku, avšak výsledná hodnota byla horší, než při hodnotě 0.003.

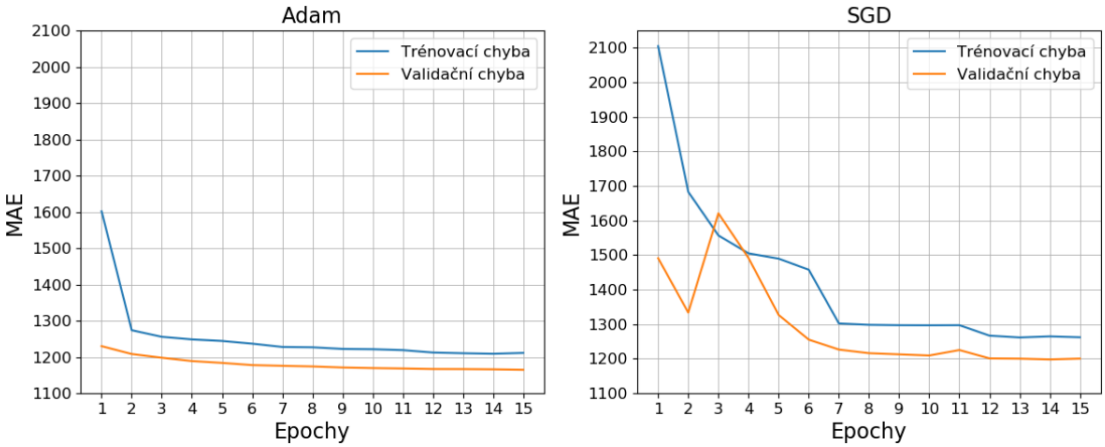
Na obrázku níže je vidět průběh učení 4-vrstvé sítě s PReLU aktivační funkcí a Dropout regularizací. Trénovací chyba je po celou dobu učení vyšší než validační chyba, současně však konstantně klesá. Model neprojevuje známky overfitting-u. Učení bylo stabilní a při opakovaném běhu výsledná chyba dospěla k podobné hodnotě.



Obrázek 10: Vývoj trénovací a validační chyby v průběhu učení modelu 3. Zdroj: vlastní zpracování.

4.3.2 Adam vs SGD

Z obrázku (11) je vidět, že průběh učení s optimalizací Adam je stabilnější. Od první epochy je trénovací i validační chyba nižší ve srovnání s SGD. Výsledná validační chyba je taktéž nižší v případě metody Adam.



Obrázek 11: Srovnání průběhu učení za použití optimalizačního algoritmu Adam (vlevo) a SGD (vpravo). Zdroj: vlastní zpracování.

4.3.3 Výsledky jednotlivých modelů

Každý model byl vytrénován 10x a nejlepší dosažené výsledky jsou zobrazeny v tabulce níže. Metrika MAE byla vyhodnocena stránkou Kaggle na testovacích datech. Zvolený optimalizační algoritmus při učení byl Adam.

Model	MAE	Počet parametrů	Doba 1 epochy
Výchozí hodnota zadaná pořadatelem	1227.74973	-	-
2-vrstvá síť	1177.53523	73 921	$\sim 28.01s$
4-vrstvá síť	1144.02269	558 209	$\sim 28.13s$
4-vrstvá síť, Dropout, PReLU	1136.32526	558 212	$\sim 29.14s$

Tabulka 1: Výsledky učení jednotlivých modelů se stejnými trénovacími parametry. Zdroj: vlastní zpracování.

Je vidět, že i nejjednodušší model 2-vrstvé neuronové sítě dokázal daná data modelovat s lepším výsledkem než byla výchozí hodnota. Výsledek se dále zlepšuje s více skrytými vrstvami a zavedením Dropout regularizační techniky.

4.4 Ensemble a křížová validace

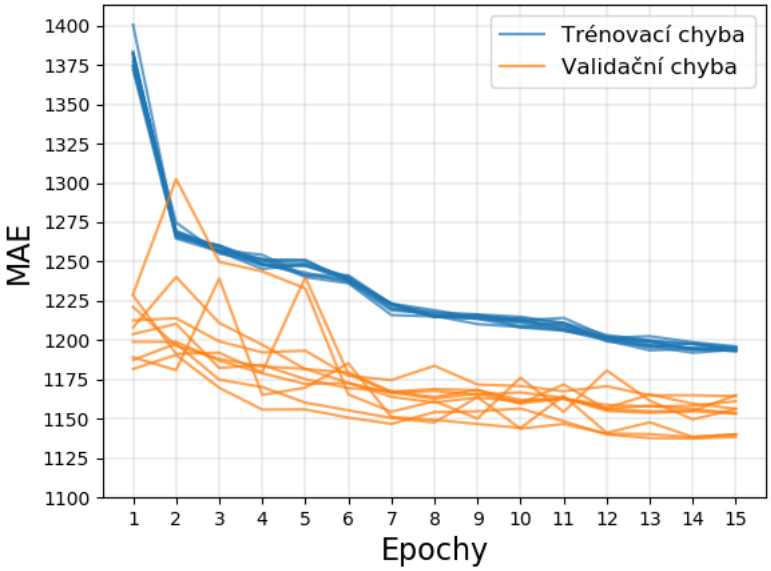
Do tohoto bodu byl validační set pro odhad generalizační chyby fixně stanoven na 10% z anotovaných dat. Prakticky jsme se tak omezili na 90% dostupných dat. V části o strojovém učení byl popsán koncept **křížové validace** a **ensemble** metoda. V této části ukážeme aplikaci K-Fold křížové validace, která rozdělí všechna anotovaná data na k podmnožin a následně vytrénuje k modelů, kde vždy jedna z podmnožin slouží jako validační set a zbylých $(k-1)$ podmnožin jako trénovací set.

V modelovém příkladu této práce bylo stanoveno $k = 10$. V takovém případě se každý z 10 modelů učí na 90% ze všech dat a zbylých 10% slouží k validaci. Těchto 10% se však liší pro každý z modelů. Současně se také aplikuje Bagging metoda popsaná v části o regularizaci, jelikož se učí 10 různých modelů a finální predikce je průměrována přes všechny modely. Jednotlivé modely převzali architekturu modelu 3 popsaného v předchozí části. Přestože jsou hyperparametry a celková architektura neuronové sítě identické, jsou modely navzájem odlišné, jelikož každý model se učí z různých dat[11] a současně každý model má různé počáteční parametry[12] a tedy jiný výchozí bod, ze kterého začíná optimalizace. V tabulce (1) vidíme počty parametrů pro jednotlivé modely, 4-vrstvá neuronová síť s Dropout regularizací a PReLU aktivační funkcí obsahuje 558 112 parametrů. Z důvodu vysoké dimenzionality daného prostoru, ve kterém optimalizační algoritmus hledá ideální parametry, je vysoce nepravděpodobné, že by se výsledné parametry jednotlivých modelů shodovaly.

Na testovacích datech dosáhl ensemble modelů hodnoty **1131.23558** a překonal tak výsledky jednotlivých modelů. Potvrdil tak hypotézu, že průměrování více modelů vytváří přesnější predikci.

Na obrázku níže je vidět, že vývoj trénovací chyby jednotlivých ensemble modelů byl velice podobný a výsledná hodnota dospěla při učení k téměř identické hodnotě mezi 1190 až 1200. Oproti tomu průběh validační chyby jednotlivých modelů byl poněkud odlišný a také výsledná hodnota validační chyby na konci učení se lišila.

[11] K-Fold křížová validace generuje různé validační sety a ve výsledku jsou tak i trénovací sety odlišné.
[12] Všechny ensemble modely byly inicializovány podle heuristiky Glorota a Bengia.



Obrázek 12: Průběh učení ensemble modelů. Zdroj: vlastní zpracování.

Koncept více modelů a křížové validace jde rozšířit i dále, například pro každý krok křížové validace vytrénovat m modelů a celkem tak vytrénovat $k \times m$ modelů. Další rozšíření je však nad rámec této práce a jeho skutečný přínos je diskutabilní.

4.5 Srovnání výsledků

Praktická část této práce popisovala modely a jejich výsledky na datech o dopravních nehodách pro havarijní pojištění. Základní 2-vrstvá neuronová síť dokázala dosáhnout lepšího výsledku než byla výchozí hodnota zadaná pořadatelem soutěže. Tento výsledek se dále zlepšil s více skrytými vrstvami, parametrickou ReLU aktivační funkcí a zavedením regularizačních metod, nejprve technikou Dropout a dále Bagging metodou, která vytrénovala 10 modelů na různých podmnožinách dat pomocí K-Fold křížové validace.

Výsledný ensemble modelů vytváří o 7.8% přesnější predikce než výchozí model pořadatele soutěže *AllState Claims Severity Challenge* [16]. Ve srovnání s ostatními účastníky dané soutěže by se pak výsledný model umístil na 1271.místě z 3056 účastníků[13].

Zásadní nevýhodou prezentované metody pak je její interpretovatelnost. Počet parametrů jednotlivých modelů (viz tabulka (1)) prakticky znemožňuje pochopit jak model k výsledné predikci dospěl. V některých případech pak možnost interpretovat model a jeho predikční schopnost může mít větší prioritu než jeho samotná přesnost. Je-li však cílem modelovat data s co největší přesností, je neuronová síť vhodným nástrojem.

[13]Toto srovnání je čistě ilustrativní. V čase psaní této práce je již soutěž vyhodnocena a uzavřena.

5 Závěr

Cílem této práce bylo představit vybranou skupinu metod strojového učení a demonstrovat jejich aplikaci v pojišťovnictví na příkladu modelování severity škod u dopravních nehod.

V úvodní části byly popsány základní pojmy z pojišťovnictví, bylo představeno členění pojištění a popsány technické rezervy s cílem naznačit, kde mohou být dané metody v rámci pojišťoven užitečné. Jednou z hlavních motivací pro vyhodnocování výše škod u dopravních nehod je stanovení výše technických rezerv, zejména pak rezerv na pojistná plnění z událostí nahlášených, ale nezlikviodaných (tzv. RBNS rezervy).

Část o strojovém učení nejprve uvedla tento obor a představila základní koncept lineární regrese. Na tento koncept poté bylo navázáno v části o hlubokém učení, jemuž se tato práce primárně věnovala, a které zahrnuje zejména neuronové sítě. V navazujících sekcích pak byly popsány jednotlivé části neuronových sítí - feed-forward neuronová síť generující predikce, chybová funkce kvantifikující míru shody mezi predikcemi a skutečnými hodnotami a optimalizační metody uplatněné při hledání ideálních parametrů. Dále byly představeny nelineární aktivační funkce, způsob inicializace parametrů a dvě základní regularizační metody.

V praktické části se tato teorie spojila a představené metody byly aplikovány na modelovém příkladu. Pro tento příklad byla použita data od americké pojišťovací společnosti AllState, která je sdílela v rámci pořádané soutěže *Claims Severity Challenge*, hostované na online soutěžní platformě Kaggle. Na tomto příkladu se všem prezentovaným modelům podařilo překonat výchozí hodnotu klíčové metriky zadanou pořadatelem. Touto metrikou byla střední absolutní chyba odhadu. Výsledný ensemble deseti 4-vrstvých neuronových sítí s PReLU aktivační funkcí a Dropout regularizací vytrénovaný pomocí K-Fold křížové validace pak dosáhl o téměř 8% přesnější predikce než výchozí model pořadatele dané soutěže. Tato práce tak ukázala, že metody strojového učení mohou být vhodným nástrojem pro vyhodnocování pojistných událostí a mohou rozšířit současné metody používané v pojistné matematice. S největší pravděpodobností už se alespoň částečně v pojistné praxi používají.

Teorie popsaná v této práci může mít značné uplatnění v rámci pojistné praxe nad rámec demonstrovaného příkladu. Strojové učení má potenciál posunout odvětví pojišťovnictví a přinést tak kýženou inovaci pro poněkud konzervativní odvětví.

Zdroje

[1] ČNB. *Zpráva o výkonu dohledu nad finančním trhem 2016*. [online] [cit. 14.2.2018] Dostupné z: https://www.cnb.cz/cs/dohled_financni_trh/souhrnne_informace_fin_trhy/zpravy_o_vykonu_dohledu/index.html. 2018, s. 109.

[2] AHMADI, Babak. *Artificial Intelligence in the Insurance Sector*. [online] [cit. 20.3.2018] Dostupné z: http://insurers.ai/documents/AI_in_Insurance.pdf.

[3] BREIMAN, Leo. „Bagging predictors". In: *Machine Learning* 24.2 (1996). Dostupné z: https://www.stat.berkeley.edu/~breiman/bagging.pdf, s. 123–140. ISSN: 0885-6125. DOI: 10.1007/BF00058655.

[4] BRYNJOLFSSON, Erik a MCAFEE, Ander. *Druhý věk strojů: práce, pokrok a prosperita v éře špičkových technologií*. Jan Melvil Publishing, 2015, s. 87–88.

[5] *Cape Analytics, INC.* [online] [cit. 11.3.2018] Dostupné z: https://capeanalytics.com/. 2018.

[6] CIPRA, Tomáš. *Pojistná matematika: teorie a praxe*. Ekopress, 2006.

[7] CLEVERT, Djork-Arné, UNTERTHINER, Thomas a HOCHREITER, Sepp. „Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *ICLR 2016* (2016). Dostupné z: http://arxiv.org/abs/1511.07289. ISSN: 09226389. DOI: 10.3233/978-1-61499-672-9-1760. eprint: 1511.07289.

[8] CYBENKO, George. „Approximation by Superpositions of a Sigmoidal Function". In: *Control Signal Systems* 2 (1989). Dostupné z: https://web.eecs.umich.edu/~cscott/smlrg/approx_by_superposition.pdf, s. 303–314.

[9] DUCHÁČKOVÁ, Eva. *Pojištění a pojišťovnictví*. Ekopress, 2015. ISBN: 978-80-87865-25-5.

[10] DUCHI, John, HAZAN, Elad a SINGER, Yoram. „Adaptive Subgradient Methods for Online Learning and Stochastic Optimization *". In: *Journal of Machine Learning Research* 12 (2011). Dostupné z: http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf, s. 2121–2159.

[11] GLOROT, Xavier a BENGIO, Yoshua. „Understanding the difficulty of training deep feedforward neural networks". In: *PMLR* 9 (2010). Dostupné z: http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf, s. 249–256. ISSN: 15324435. DOI: 10.1.1.207.2059. eprint: arXiv:1011.1669v3.

[12] GOODFELLOW, Ian, BENGIO, Yoshua a COURVILLE, Aaron. *Deep Learning*. MIT Press, 2017. ISBN: 9780262035613.

[13] HE, Kaiming et al. „Delving deep into rectifiers: Surpassing human-level performance on imagenet classification“. In: *ICCV 2015* 2015 Inter (2015). Dostupné z: http://arxiv.org/abs/1502.01852, s. 1026–1034. ISSN: 15505499. DOI: 10.1109/ICCV.2015.123. eprint: 1502.01852.

[14] Kaggle. *Allstate Claims Severity.* [online] [cit. 31.1.2018]. Dostupné z: https://www.kaggle.com/c/allstate-claims-severity. 2017.

[15] Kaggle. *AllState Claims Severity competition discussion. 1st place solution.* [online] [cit. 31.1.2018] Dostupné z: https://www.kaggle.com/c/allstate-claims-severity/discussion/26416. 2018.

[16] Kaggle. *Claim Prediction Challenge.* [online] [cit. 31.1.2018] Dostupné z: https://www.kaggle.com/c/ClaimPredictionChallenge.

[17] KARPATHY, Andrej a JOHNSON, Justin. *Stanford University CS231n: Convolutional Neural Networks for Visual Recognition.* [online] [cit. 22.2.2018]. Dostupné z: https://cs231n.github.io/. 2018.

[18] KESKAR, Nitish Shirish a SOCHER, Richard. „Improving Generalization Performance by Switching from Adam to SGD“. In: *CoRR* (2017). Dostupné z: http://arxiv.org/abs/1712.07628. eprint: 1712.07628.

[19] KINGMA, Diederik a BA, Jimmy. „Adam: A Method for Stochastic Optimization“. In: *ICLR 2015* (2014). Dostupné z: http://arxiv.org/abs/1412.6980. eprint: 1412.6980.

[20] KRIZHEVSKY, Alex, SUTSKEVER, Ilya a HINTON, Geoffrey. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems 25 (NIPS2012)* (2012). Dostupné z: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, s. 1–9. ISSN: 10495258. DOI: 10.1109/5.726791. arXiv: 1102.0183.

[21] LECUN, Yann et al. „Efficient backprop“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2012). ISSN: 03029743. DOI: 10.1007/978-3-642-35289-8-3. eprint: 9809069v1.

[22] *Lemonade Insurance Agency, LLC.* [online] [cit. 11.3.2018] Dostupné z: https://www.lemonade.com/. 2018.

[23] *Lemonade Sets New World Record. Lemonade Insurance Agency, LLC.* [online] [cit. 11.3.2018] Dostupné z: https://www.lemonade.com/blog/lemonade-sets-new-world-record/. 2017.

[24] MANDL, Petr a MAZUROVÁ, Lucie. *Matematické základy neživotního pojištění.* Matfyzpress, 1999. ISBN: 80-85863-42-1.

[25] MIKOLOV, Tomáš et al. „Distributed Representations of Words and Phrases and their Compositionality“. In: *Advances in Neural Information Processing Systems* (2013). Dostupné z: https://arxiv.org/pdf/1310.4546.pdf. arXiv: arXiv:1310.4546v1.

[26] PASZKE, Adam et al. „Automatic differentiation in PyTorch“. In: Dostupné z: https://openreview.net/forum?id=BJJsrmfCZ. 2017.

[27] PASZKE, Adam et al. *PyTorch*. Dostupné z: http://pytorch.org/. 2018.

[28] Python Software Foundation. *Python Language Reference, version 3.6*. Dostupné z: https://www.python.org/. 2018.

[29] RAMACHANDRAN, Prajit, ZOPH, Barret a LE, Quoc Viet. *Searching for Activation Functions*. Dostupné z: http://arxiv.org/abs/1710.05941. 2017. eprint: 1710.05941.

[30] RUSSELL, Stuart a NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. 3. vyd. Pearson, 2009. ISBN: 9781292153964.

[31] SIAMI NAMIN, Sima a SIAMI NAMIN, Akbar. *Forecasting Economics and Financial Time Series: ARIMA vs. LSTM*. Dostupné z: http://arxiv.org/abs/1803.06386. 2018. arXiv: 1803.06386.

[32] SRIVASTAVA, Nitish et al. „Dropout: A Simple Way to Prevent Neural Networks from Overfitting“. In: *Journal of Machine Learning Research* 15 (2014). Dostupné z: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf, s. 1929–1958.

[33] TIELEMAN, Tijmen a HINTON, Geoffrey. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012.

[34] WANG, Wenhui et al. „R-NET: Machine Reading Comprehension with Self-matching Networks“. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Dostupné z: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf. 2017, s. 1–11.

[35] Wikipedia. *Kaggle. Wikipedia, the free encyclopedia*. [online] [cit. 31.1.2018]. Dostupné z: https://en.wikipedia.org/wiki/Kaggle. 2018.

[36] WÜTHRICH, Mario a MERZ, Michael. *Stochastic Claims Reserving Methods in Insurance*. John Wiley & Sons, Ltd, 2008. ISBN: 978-0-470-72346-3.

[37] Zákon č. 277/2009 Sb., o pojišťovnictví. *Sbírka zákonů*. 22. 7. 2009.

Seznam obrázků

1 Odvětví neživotního pojištění tuzemských pojišťoven dle výše hrubého předepsaného pojistného. Zdroj: [1] 4

2 Vennův diagram popisující vztah umělé inteligence, strojového učení a deep learningu. Zdroj: [12] 9

3 Průběh sigmoid funkce v intervalu $< -10, 10 >$, též logistická křivka (vlevo). První derivace funkce sigmoid (vpravo). Zdroj: vlastní zpracování. 13

4 Ukázka 2-vrstvé umělé neuronové sítě. Zdroj: vlastní zpracování. 14

5 Průběh ReLU funkce v intervalu $< -5, 5 >$ (vlevo). První derivace ReLU funkce (vpravo). Zdroj: vlastní zpracování. 20

6 Průběh Leaky ReLU funkce v intervalu $< -5, 5 >$ pro $\alpha \in \{0.01, 0.05, 0.1\}$ (vlevo). První derivace Leaky ReLU funkce (vpravo). Zdroj: vlastní zpracování. 21

7 Standardní neuronová síť se 2 skrytými vrstvami (vlevo). Užší neuronová síť po aplikaci dropout metody, neurony s křížkem jsou ignorovány (vpravo). Zdroj: [32] 24

8 Zjednodušená vizualizace rozdělení na trénovací, validační a testovací set. Zdroj: vlastní zpracování. 27

9 Model 1 (vpravo), Model 2 (uprostřed), Model 3 (vlevo). (Číslo popisuje dimenzionalitu) Zdroj: vlastní zpracování. 28

10 Vývoj trénovací a validační chyby v průběhu učení modelu 3. Zdroj: vlastní zpracování. 29

11 Srovnání průběhu učení za použití optimalizačního algoritmu Adam (vlevo) a SGD (vpravo). Zdroj: vlastní zpracování. 30

12 Průběh učení ensemble modelů. Zdroj: vlastní zpracování. 32

Seznam tabulek

1 Výsledky učení jednotlivých modelů se stejnými trénovacími parametry. Zdroj: vlastní zpracování. 30