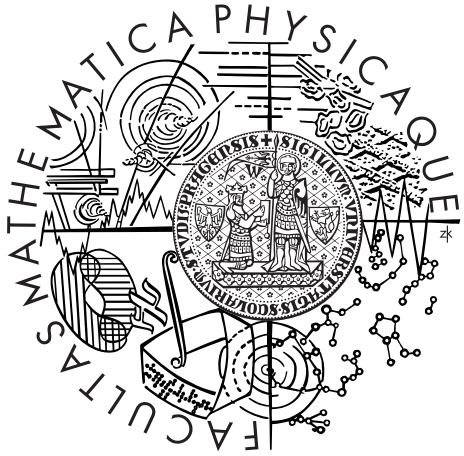
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Lubomír Šerý

Metody umělé inteligence a jejich využití při predikci

Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce:	Ing. Marek Omelka, Ph.D.
Studijní program:	Matematika
Studijní obor:	MFAPM

Praha 2012

Na tomto místě bych chtěl poděkovat Ing. Marku Omelkovi, Ph.D., za pomoc, cenné připomínky a trpělivost. Dále bych rád poděkoval své rodině za velkou podporu při psaní této práce.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 25.7.2012 Podpis autora

Název práce: Metody umělé inteligence a jejich využití při predikci

Autor: Lubomír Šerý

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce: Ing. Marek Omelka, Ph.D., Katedra pravděpodobnosti a matematické statistiky

Abstrakt: V předložené práci studujeme část oboru umělé inteligence - především umělé neuronové sítě. Nejprve je představen koncept umělé neuronové sítě a srovnání s jeho biologickou předlohou. Poté srovnáváme neuronovou síť s některými zobecněnými lineárními modely. Učení neuronové sítě představuje jeden ze stěžejních problému, a tak je mu věnována větší část této práce - především odhadům parametrů a specifickým výpočetním aspektům. V této části se pokoušíme vnést pohled na vnitřní strukturu neuronové sítě a navrhnout vylepšení učícího algoritmu. Existuje mnoho nadstaveb, kterými lze vylepšit nebo obohatit základní model neuronové sítě, některé tyto nadstavby včetně kombinace s genetickými algoritmy jsou představeny v závěru této práce. Tuto práci uzavírají simulační příklady, v kterých se snažíme prověřit některé představené teoretické předpoklady a závěry. Hlavním simulačním příkladem je aplikace konceptu neuronové sítě na úlohu predikce počtu gólů v hokejových zápasech.

Klíčová slova: Neuronová síť, Predikce, TwoStepNNL, Sport

Title: Methods of artificial intelligence and their use in prediction

Author: Lubomír Šerý

Department: Department of Probability and Mathematical Statistics

Supervisor: Ing. Marek Omelka, Ph.D., Department of Probability and Mathematical Statistics

Abstract: In the presented thesis we study field of artificial intelligence, in particular we study part dedicated to artificial neural networks. At the beginning, concept of artificial neural networks is introduced and compared to it's biological base. Afterwards, we also compare neural networks to some generalized linear models. One of the main problems of neural networks is their learning. Therefore biggest part of this work is dedicated to learning algorithms, especially to parameter estimation and specific computational aspects. In this part we attempt to bring in an overview of internal structure of neural network and to propose enhancement of learning algorithm. There are lots of techniques for enhancing and enriching basic model of neural networks. Some of these improvements are, together with genetic algorithms, introduced at the end of this work. At the very end of this work simulations are presented, where we attempt to verify some of the introduced theoretical assumptions and conclusions. Main simulation is an application of concept of neural networks on a problem of prediction of number of goals during an ice-hockey match.

Keywords: Neural network, Prediction, TwoStepNNL, Sport modeling

Obsah

Úvod

S rozvojem výpočetní techniky se stále častěji setkáváme s požadavky na statistickou analýzu dat. Dnes již máme k dispozici řadu nejrůznějších informací, díky kterým můžeme zkoumat různé závislosti a interakce, které dříve bez rozvoje počítačů nebylo možné precizně ověřovat. Díky internetu, informacím ve formě vhodné pro počítače a řadě metod a algoritmů dnes můžeme testovat nejrůznější hypotézy a dále zdokonalovat používané algoritmy z oblasti *dobývání znalostí*.
Tato práce představuje a popisuje jednu z oblíbených metod strojového učení - *umělé neuronové sítě*. Tato metoda v kombinaci s jinou metodou inspirovanou přírodními procesy - *genetické algoritmy* přestavuje jednu z mnoha takových používaných metod. V této práci nejprve v první kapitole představíme koncept umělých neuronových sítí. Ve druhé a třetí kapitole se pokusíme vnést pohled na některé výpočetní aspekty v rámci odhadů parametrů. V následující čtvrté kapitole se pokusíme najít případné vylepšení a prozkoumáme rozšiřující techniky nad rámec základního modelu. Za jedno z takových možných rozšíření je použití genetického algoritmu při stavbě neuronové sítě. V závěrečné páté kapitole se pokusíme některé teoreticky popsané aspekty ověřit pomocí simulačních příkladů. Hlavním simulačním příklad je srovnání popsaných a navržených metod v problematice modelování sportovních zápasů - konkrétněji počtu gólů v kanadsko-americké hokejové soutěži NHL a jejich srovnání proti kurzům bookmakerům sázkových kanceláří.
Tato práce v žádném případě nemá ambici popsat celou problematiku používání metody umělých neuronových sítí, představuje pouze poznatky, které autor posbíral během psaní této práce a během koníčku a samostudia ve snaze odhalit nejrůznější souvislosti a interakce v datech týkajících se sportovních utkání.

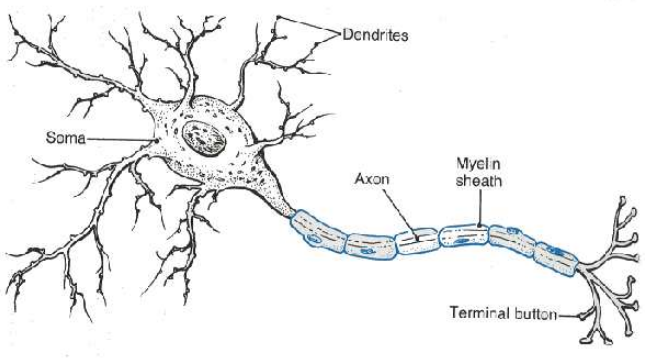
1. Umělá inteligence

Umělá inteligence představuje obor zabývající se tvorbou algoritmů a technik, které se snaží přiblížit projevům inteligentního chování. V poslední době řada autorů pod pojmem umělá inteligence míní především populární obor umělých neuronových sítí. Toto úzké zařazení opomíjí celou řadu přístupů z oblasti strojového učení, tvorby expertních systémů, algoritmů inspirovaných přírodními procesy - např. evoluční a genetické techniky nebo algoritmy prohledávání stavového prostoru. V této práci se budeme zabývat především oborem umělých neuronových sítí zaměřených na úlohu predikce. K hledání optimální topologie neuronové sítě budou použity genetické algoritmy.

1.1 Neuronové sítě

1.1.1 Biologická motivace

Neuronové sítě představují model inspirovaný biologickou strukturou lidského mozku. Základní stavební jednotkou je nervová buňka - *neuron*. Neurony jsou vzájemně propojené a na bázi elektrochemických procesů si navzájem předávají informace. Tvar biologického neuronu popisuje obr.1.1. Tělo neuronu se nazývá *soma*, do kterého přicházejí impulzy z jiných neuronů pomocí spojů - *dendritů*. Pokud jsou příchozí signály dostatečně velké, dojde k *excitaci* neuronu - předání informace. Signál je pomocí výstupní části - *axonu*, pomocí zakončení, tzv. *terminálů axonu*, předáván dalším neuronům v síti. Po průchodu signálu sítí neuronů se dostaví odpovídající reakce organismu. V lidském mozku jsou řádově miliardy (13 - 15 miliard uvádí [1, p.22]) takto zapojených neuronů, kde každý neuron může být propojen až s 5 000 dalšími neurony. Základní matematický model takové sítě pracuje s pouhými jednotkami neuronů. V kapitole 4.4 prozkoumáme konstrukci složitějších struktur využívající až stovky neuronů.

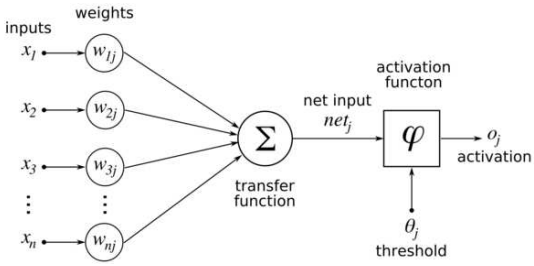


Obrázek 1.1: Biologický neuron, převzato [12].

1.1.2 Matematický model neuronové sítě

Při konstrukci umělých neuronových sítí se vychází ze sítí biologických. Základní jednotkou je opět neuron, který v matematickém smyslu kopíruje svoji biologickou

předlohu. Nejpoužívanější typ matematického neuronu zobrazuje obr. 1.2.



Obrázek 1.2: Matematický model neuronu, převzato [13].

Formálně můžeme zapsat jako

$$o = \varphi\Big(\sum_{i=1}^{n} w_i x_i + \theta\Big), \tag{1.1}$$

kde x_i jsou vstupní informace, w_i představují váhy - parametry neuronu, θ představuje tzv. *práh* neuronu - rovněž parametr, funkce $\varphi(x)$ je tzv. *aktivační funkce*. Nejpoužívanějšími typy aktivačních funkcí jsou

(a) Sigmoid

$$\varphi(x) = \frac{1}{1+e^{-x}}, \quad -\infty < x < \infty, \tag{1.2}$$

(b) Hyperbolický tangens

$$\varphi(x) = \tanh(x) = \frac{1-e^{-x}}{1+e^{-x}}, \quad -\infty < x < \infty, \tag{1.3}$$

(c) Arctan

$$\varphi(x) = \arctan(x), \quad -\infty < x < \infty. \tag{1.4}$$

Grafické znázornění aktivačních funkcí demonstruje obrázek 1.3.

1.1.3 Typy přístupů k neuronovým sítím

Základní dělení modelů neuronových sítí určuje typ řešené úlohy a způsoby zapojení jednotlivých neuronů. Základní typy úloh pro neuronové sítě můžeme rozdělit do tří oblastí

(1) Učení s učitelem

(2) Učení bez učitele

(3) Posilované učení

Učení s učitelem (Supervised learning) představuje typ úlohy, kde máme k dispozici dvojici matic $\mathbf{X}$, $\mathbf{Y}$ a hledáme funkci

$$f : \mathbf{X} \rightarrow \mathbf{Y}. \tag{1.5}$$

Matice $\mathbf{X}$ typu $n \times m$

$$\mathbf{X} = \{x_{i,j}\}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m, \tag{1.6}$$

je matice nezávislých proměnných (konstrukční matice nebo také matice vzorů), kde i-tý řádek $\mathbf{x_i}$ reprezentuje i-té pozorování a na j-tých pozicích jsou jednotlivé charakteristiky tohoto pozorování.
Matice $\mathbf{Y}$ typu $n \times r$

$$\mathbf{Y} = \{y_{i,k}\}, \quad i = 1, \ldots, n, \quad k = 1, \ldots, r. \tag{1.7}$$

představuje matici závisle proměnných (odezev nebo také targetů). Nejčastějším případem je úloha, kdy sloupcový rozměr matice $\mathbf{Y}$ je $r = 1$. Pravděpodobnostní rozdělení náhodné veličiny $y_{i,.}$ pak určuje povahu úlohy - ve smyslu dělení na klasifikaci (classification) a regresi (function approximation).

Nalezení funkce $f = f(\mathbf{x_i}, \mathbf{w}, \theta)$ a jejích optimálních hodnot parametrů $\{\mathbf{w}, \theta\}$ představuje tzv. *učící proces*. Obvykle použitou technikou při učení neuronových sítí je metoda nejmenších čtverců - učící proces pak můžeme popsat jako minimalizační úlohu (při $r = 1$)

$$\min_{\mathbf{w},\theta} \sum_{i=1}^{n} (f(\mathbf{x_i}, \mathbf{w}, \theta) - y_i)^2. \tag{1.8}$$

Řešením této úlohy se budeme zabývat podrobněji v kapitole 3.2. Pro tento typ úloh se rozlišuje učící proces neuronové sítě (řešení minimalizační úlohy 1.8) a aplikace již naučené neuronové sítě na nový vektor nezávislých proměnných (nové pozorování).

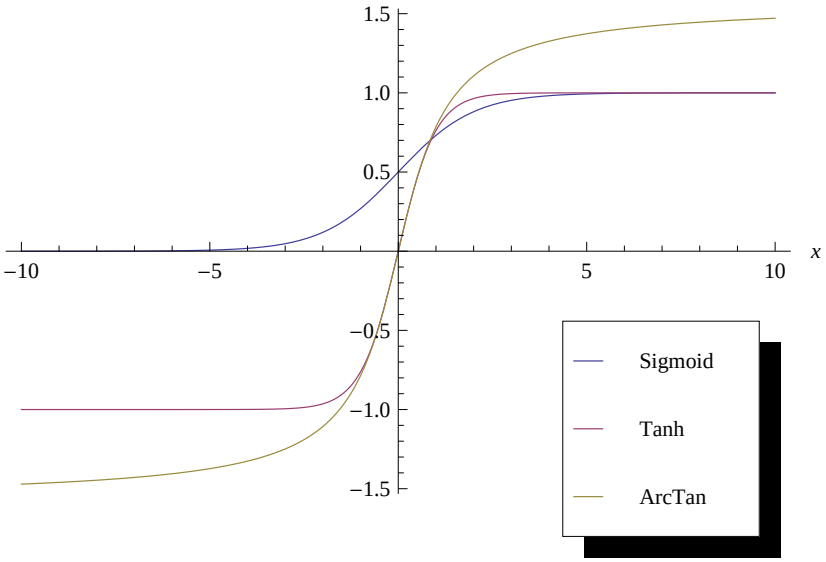
Příkladem může být úloha predikce neschopnosti klienta dostát svým závazkům v situaci splácení úvěru - například hypotéky. K dispozici bychom měli data o n klientech - matice $\mathbf{X}$, kde by i-tý řádek reprezentoval informace pro klienta i. Na pozicích j by pro daného klienta i byly nezávislé proměnné - informace typu {výše příjmu domácnosti klienta, počet dětí klienta, velikost půjčky, apod.}. Závisle proměnná y_i pro klienta i, pak vyjadřuje zda klient svůj úvěr splatil či nikoliv.

Učení bez učitele (Unsupervised learning) představuje například algoritmy shlukové analýzy, komprese dat nebo filtrování. Na rozdíl od učení s učitelem pracuje pouze s maticí $\mathbf{X}$. Volba funkce f je v tomto případě úzce svázána s typem řešené úlohy. V oblasti neuronových sítí je nejpoužívanějším modelem tzv. *Kohonenova samoorganizační mapa* často označovaná jako SOM (Self-Organizing Map) [1, p.103].

Posilované učení (Reinforcement learning) na rozdíl od předchozích dvou přístupů nerozlišuje mezi učící částí a aplikací neuronové sítě. Matice $\mathbf{X}$ často není

předem známá a data pro učení neuronové sítě jsou generovaná na základě interakce s vnějším prostředím. Příkladem může být získávání dat z různých senzorů robotického zařízení a následném vyhodnocování úspěšného či neúspěšného pohybu v cizím prostředí. Tento přístup se nejčastěji využívá v robotice.

V této práci se budeme dál zabývat prvním přístupem - *učení s učitelem* a především jeho použitím na úlohu predikce.



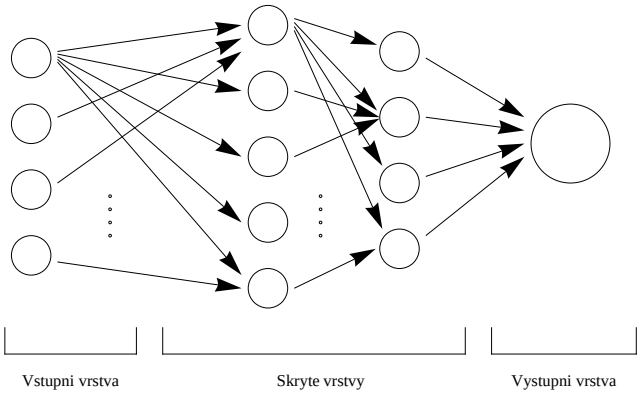
Obrázek 1.3: Typy aktivačních funkcí.

1.1.4 Architektura neuronové sítě

Jedním z důležitých kroků při konstrukci neuronové sítě je volba modelu f (dále uvažujeme pouze typ úloh tvaru $f : \mathbf{X} \to \mathbf{Y}$). Tvar modelu určují způsob zapojení a počty jednotlivých neuronů. Základním rozdělením architektur je dle výskytu cyklu při předávání informací. Pak můžeme rozlišit

- Neuronové sítě bez cyklů - tzv. dopředné, nejčastěji
  - Vícevrstvé neuronové sítě - MLP (Multi-layer perceptrons) - popis viz. dále
  - Radiální sítě - RBF (Radial basis function) - [1, p.118]
- Neuronové sítě s cyklem - tzv. rekurentní - nejčastěji Hopfieldova síť [1, p.196]

Vícevrstvé neuronové sítě jsou nejpoužívanějším typem architektury. Model vychází z předpokladu po vrstvách dopředného šíření signálu. První vrstva je vrstva vstupů, poslední je vrstva výstupu. Mezi nimi jsou tzv. *skryté vrstvy (hidden layers)*.



Obrázek 1.4: Síť typu MLP.

Volba počtu skrytých vrstev a počty neuronů v jednotlivých vrstvách jsou mimo parametrů $\{\mathbf{w}, \theta\}$ parametry modelu f. Pro optimalizaci architektury neuronové sítě lze například použít techniku genetických algoritmů, viz. následující kapitola.

1.2 Genetické algoritmy

Genetické algoritmy *(dále GA)* patří vedle neuronových sítí do další skupiny přístupů inspirovaných přírodními procesy. Představují techniku hledání řešení obecně pro jakýkoliv problém. Principem je generování velkého množství *populací* = množin možných řešení, které vhodnými evolučními operátory šlechtíme směrem k nalezení globálního optima. GA se často využívají jakožto optimalizační technika v případě, že účelová funkce (kriteriální funkce, fitness function) není v prostoru možných řešení spojitá nebo diferencovatelná a nelze použít žádnou ze známých gradientních metod. Další populární využití GA představuje koncept *genetického programování*, kdy se například GA snaží ze zadaných *elementárních kamenů* poskládat hledaný algoritmus, sadu pravidel nebo celý program. V této práci použijeme GA pro řešení optimalizační úlohy - viz. kapitola 3.3.4.

1.2.1 Optimalizace pomocí GA

Při řešení optimalizační úlohy hledáme v prostoru $\mathcal{A}$ argument minima(maxima) definované účelové funkce $g : \mathcal{A} \to \mathbf{R}$. Řešením takové úlohy je $x_0 \in \mathcal{A}$ takové, že $g(x_0) \leq g(x)$ pro $\forall x \in \mathcal{A}$. Pro řešení optimalizační úlohy pomocí GA je nutné pro $\forall x \in \mathcal{A}$ najít vhodnou reprezentaci a v ní kódování vhodné pro genetické operátory. Nejčastěji používaný je řetězec jedniček a nul. Pro složitější úlohy se volí složitější reprezentace například pomocí matic nebo stromů. Pro každý bod prostoru $\mathcal{A}$ existuje *překlad* do zvolené reprezentace $\phi : \mathcal{A} \to \mathbf{\Lambda}$ stejně jako inverzní zobrazení $\phi^{-1} : \mathbf{\Lambda} \to \mathcal{A}$.

Samotný algoritmus

GA je iterativní algoritmus. V první iteraci je vygenerována (většinou náhodně) prvotní populace vhodných řešení $P_0 \subset \Lambda$. Každého jedince v populaci $\lambda_{i,k} \in P_i$ lze ohodnotit hodnotou účelové funkce $g(\phi^{-1}(\lambda_{i,k}))$. Každá následující populace P_i je zkonstruována z předchozí P_{i-1} pomocí genetických operátorů. Mezi základní genetické operátory patří:

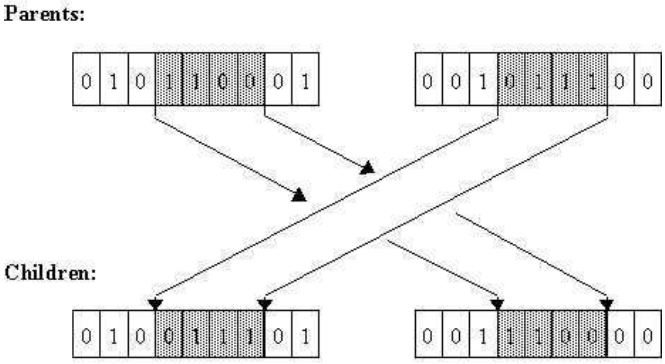
- Mutace - Náhodná změna binární hodnoty na náhodné pozici v řetězci. Míra pravděpodobnosti mutací a intenzita (počet změněných pozic) výrazně ovlivňuje schopnost a rychlost konvergence procesu optimalizace. Mutační operátor zvyšuje odolnost algoritmu vyhnout se lokálním extrémům.

- Křížení - Při křížení dojde k výběru dvou jedinců v populaci - *rodičů* a vytvoření jejich *potomka* kombinací binárních informací od obou rodičů. Nejčastěji potomek vzniká nahrazením části řetězce jednoho z rodičů částí druhého rodiče. Podstatnou částí je samotný výběr rodičů. Pro křížení se používají s vyšší pravděpodobností ti jedinci, kteří mají lepší ohodnocení účelovou funkcí - čímž se snažíme vytvořit populaci s lepšími vlastnostmi.

- Elitářství - V případě, kdy chceme zaručit, že každá nová generace je stejná nebo lepší (ve smyslu ohodnocení nejsilnějšího jedince populace) použijeme tzv. *elitářství*. To spočívá v zachování nejsilnějšího jedince (nebo více jedinců) do každé další generace. Tento operátor pro některé úlohy zlepšuje

rychlost konvergence, pro některé úlohy může způsobit zaseknutí v lokálním extrému.

Velikost populace v každém kroku a počet iterací závisí většinou na výpočetních možnostech. GA je obecně pomalejší než gradientní metody. Doporučují se řádově tisíce jedinců v populaci. Ohodnocení jedinců je paralelizovatelná úloha, a tak se nabízí řešení pomocí grafických čipů - například pomocí CUDA technologie. Iterační cyklus zastavíme ve chvíli, kdy nejlepší jedinec v populaci splňuje požadavky na hodnotu účelové funkce.



Obrázek 1.5: Křížení v populaci, převzato z [14].

2. Neuronové sítě jako regresní a prediktivní metoda

V této kapitole nejprve představíme nejčastěji používané regresní lineární modely, poté přejdeme k úloze regrese s využitím neuronové sítě.

2.1 Lineární modely

Častým přístupem zpracování dat jsou lineární modely. Především díky snadné a přímočaré interpretaci modelu, interpretaci vysvětlení nalezených souvislostí a rychlými odhadovacími algoritmy, představují lineární modely nejvyhledávanější přístupy pro zpracování dat. Základním konceptem pro hledání souvislostí v datech je model **lineární regrese**. Dle modelované závisle proměnné se mimo základní lineární regresi například používá logistická regrese nebo poissonovská regrese (modely z rodiny Zobecněných lineárních modelů [24]). Obě metody použijeme pro srovnání s navrhnutým modelem používající neuronové sítě, a proto krátce uvedeme základy těchto modelů - stejně jako v [2].

2.1.1 Logistická regrese

V případě, že modelovaná proměnná Y nabývá pouze dvou hodnot (označme si je 0 a 1), mluvíme o tzv. binární proměnné. Nejčastějším přístupem pro modelování binární proměnné je pravděpodobnostní interpretace, kdy odhad proměnné Y_i vyjadřuje pravděpodobnost $P(Y_i = 1)$, a tedy Y_i představuje náhodnou veličinu. Stejně jako v úloze (1.5) předpokládáme znalost konstrukční matice $\mathbf{X}$

$$\mathbf{X} = \{x_{i,j}\}, \quad i = 1, \dots n, \quad j = 1, \dots m, \tag{2.1}$$

kde řádkový vektor $\mathbf{x_i}$

$$\mathbf{x_i} = \{x_{i,1}, \dots, x_{i,m}\}, \qquad j = 1, \dots m, \tag{2.2}$$

odpovídá jednomu pozorování. Dále předpokládáme znalost vektoru odpovídajících závislých proměnných $\mathbf{y}$

$$\mathbf{y} = \{y_i\}^T, \quad i = 1, \dots, n. \tag{2.3}$$

Obvyklý formální zápis odpovídající pravděpodobnostnímu modelu je

$$P(Y_i = 1|\mathbf{x_i}, \boldsymbol{\beta}) = 1 - F(-\mathbf{x_i}\boldsymbol{\beta}), \quad i = 1, \dots, n, \tag{2.4}$$

kde $F(\cdot)$ je vhodná pravděpodobnostní distribuční funkce a $\boldsymbol{\beta}$ je sloupcový vektor hledaných parametrů modelu. Výrazem $P(Y_i = 1|\mathbf{x_i}, \boldsymbol{\beta})$ rozumíme podmíněnou pravděpodobnost jevu, že náhodná veličina Y_i nabude hodnoty 1 za podmínky nastání $\mathbf{x_i}$ a použití parametrů $\boldsymbol{\beta}$. Jeden z oblíbených modelů je model *logit*. Tento model má tvar

$$P(Y_i = 1|\mathbf{x_i}, \boldsymbol{\beta}) = 1 - F(-\mathbf{x_i}\boldsymbol{\beta}) = 1 - \frac{e^{-\mathbf{x_i}\boldsymbol{\beta}}}{1 + e^{-\mathbf{x_i}\boldsymbol{\beta}}} = \frac{e^{\mathbf{x_i}\boldsymbol{\beta}}}{1 + e^{\mathbf{x_i}\boldsymbol{\beta}}}. \tag{2.5}$$

Odhad parametrů modelu se nejčastěji provádí metodou maximální věrohodnosti. Logaritmická věrohodnostní rovnice má tvar

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \cdot \ln F(\mathbf{x_i}\boldsymbol{\beta}) + \sum_{i=1}^{n} (1 - y_i) \cdot \ln(1 - F(\mathbf{x_i}\boldsymbol{\beta})). \tag{2.6}$$

2.1.2 Poissonovská regrese

Předpokládáme, že modelovaná závisle proměnná má čítací charakter [2, p.185] s Poissonovým rozdělením. Potom podmíněná střední hodnota $m(\boldsymbol{x_i}, \boldsymbol{\beta})$ čítací vysvětlované proměnné Y_i má tvar

$$m(\mathbf{x_i}, \boldsymbol{\beta}) = \mathbf{E}(Y_i|\mathbf{x_i}, \boldsymbol{\beta}) = e^{\mathbf{x_i}\boldsymbol{\beta}}, \tag{2.7}$$

kde $\mathbf{x_i}$ je řádkový vektor regresorů a $\boldsymbol{\beta}$ je sloupcový vektor odhadovaných parametrů. Čítací model pro Poissonovo rozdělení pak můžeme psát

$$P(Y_i = j|\mathbf{x_i}, \boldsymbol{\beta}) = e^{-m(\mathbf{x_i}, \boldsymbol{\beta})} \cdot \frac{m(\mathbf{x_i}, \boldsymbol{\beta})^j}{j!}. \tag{2.8}$$

Důležitý předpoklad modelu, který můžeme na datech ověřovat je rovnost střední hodnoty a rozptylu

$$v(\mathbf{x_i}, \boldsymbol{\beta}) = \mathrm{var}(Y_i|\mathbf{x_i}, \boldsymbol{\beta}) = m(\mathbf{x_i}, \boldsymbol{\beta}). \tag{2.9}$$

Pro odhad parametrů se použije metoda maximální věrohodnosti. Logaritmická věrohodnostní rovnice má tvar

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \{y_i \cdot \ln m(\mathbf{x_i}, \boldsymbol{\beta}) - m(\mathbf{x_i}, \boldsymbol{\beta}) - \ln y_i!\}. \tag{2.10}$$

Predikce vysvětlované proměnné se pak získají jako

$$\widehat{y_i} = m(\mathbf{x_i}, \widehat{\boldsymbol{\beta}}) = e^{\mathbf{x_i}\widehat{\boldsymbol{\beta}}}. \tag{2.11}$$

2.1.3 Stepwise regrese

Jedna z obtížných úloh při stavbě regresních modelů je výběr nezávislých proměnných vstupujících do modelu. Zejména v případě, kdy počet možných nezávislých proměnných je relativně vysoký, nebo nevíme, zda všechny proměnné jsou relevantní nebo v případě jejich vzájemné závislosti. Výběru proměnných (features selection) se věnuje několik možných přístupů, často používané jsou přístupy založené na testech významnosti přidávání či odebíraní jednotlivých nezávisle proměnných. Jedním z takových algoritmů je *Stepwise regrese*. Předpokládejme model logistické regrese - kapitola 2.1.1. Dále předpokládejme, že první sloupec konstrukční matice je sloupec samých 1. Popíšeme variantu postupně se rozšiřující regrese - stejně jako v [9] můžeme postup rozepsat do tří kroků:

Krok 1

Označme L_M hodnoty logaritmu věrohodnostní funkce (například pro model logistické regrese - (2.6)) za předpokladu, že v modelu jsou regresory obsažené v množině M. Předpokládejme, že v modelu je zahrnut regresor s příslušným parametrem β_1 a výběr dalšího případného kandidáta založíme na významnosti statistiky

$$G_j^{(0)} = 2(L_{\{1,j\}} - L_{\{1\}}). \tag{2.12}$$

Statistika $G_j^{(0)}$ má za platnosti hypotézy nulovosti β_j asymptoticky rozdělení χ^2 o jednom stupni volnosti. Označme $p_j^{(0)}$ pravděpodobnost

$$p_j^{(0)} = P[\chi^2(1) > G_j^{(0)}]. \tag{2.13}$$

Čím menší je tato pravděpodobnost, tím spíše je model s příslušným regresorem j vhodnější. Označme

$$e_1 = \underset{j}{\operatorname{argmin}}\ p_j^{(0)}, j = 2, \ldots, n, \tag{2.14}$$

index, který přísluší nejmenší hodnotě $p_j^{(0)}$. Stanovme předem hodnotu p_E, kterou budeme používat jako kritérium pro zařazení do modelu. V praxi se často nastavuje na hodnotu $p_E = 0{,}15$ ([9, p.10]). Algoritmus skončí v případě, když $p_{e_1} \geq p_E$. Jinak položme $s = 0$ (index iterace) a pokračujme krokem 2.

Krok 2

Označme E_s množinu indexů v modelu po iteraci s. Položme $s = s + 1$. V dalším kroku vybereme stejným způsobem další proměnnou, kterou přidáme do modelu. Tedy spočítáme hladinu významnosti pro statistiku

$$G_j^{(s)} = 2(L_{E_{s-1} \cup \{j\}} - L_{E_{s-1}}), j \notin E_s. \tag{2.15}$$

Stanovíme hodnotu $e_s = \underset{j \notin E_s}{\operatorname{argmin}}\ p_j^{(s)}$, kde $p_j^{(s)} = P[\chi^2(1) > G_j^{(s)}]$. Pokud $p_{e_s} > p_E$ algoritmus končí, jinak se položí $E_s = E_s \cup \{e_s\}$ a přejde se na krok 3.

Krok 3

V tomto kroku se prověří, zda proměnná přidávaná v předchozím kroku do modelu neznamená, zda by se některá z již dříve zařazených proměnných neměla z modelu vyloučit v závislosti na nově zařazené proměnné. Toto se provádí opět prostřednictvím statistik

$$G_j^{(s_B)} = 2(L_{E_s} - L_{E_s \setminus j}), j \notin \{1, e_s\}. \tag{2.16}$$

Horní index s_B znamená, že jde o výpočet statistik ve zpětné fázi. Vypočteme hodnoty $p_j^{s_B} = P[\chi^2(1) > G_j^{(s_B)}]$. Ve zpětném kroku je potřeba určit, která proměnná má největší šanci být vyloučena, tedy proti kroku 2 nás zde zajímají vysoké hodnoty $p_j^{s_B}$, které svědčí o pravdivosti hypotézy, že β_j je nulová. Označme proto

$$r_s = \underset{j \in E_s \setminus \{1, e_s\}}{\operatorname{argmax}} p_j^{s_B}. \tag{2.17}$$

Pokud hodnota p_{r_s} poklesne pod předem danou hladinu p_R, proměnnou r_s vyloučíme z modelu, tj. položíme $E_s = E_s \setminus \{r_s\}$. Potom přejdeme na krok 2. Hodnota p_R musí být větší než p_E, čímž je zaručeno, že se algoritmus nezacyklí.

Popsaný algoritmus je zřejmě konečný vzhledem ke konečnému počtu sloupců konstrukční matice a nemožnosti se zacyklit volbou $p_R > p_E$. Doporučené hodnoty pro volbu p_E jsou z intervalu $[0{,}15; 0{,}2]$. Hodnotu p_R pak volíme o 0,05 až 0,1 větší [9, p.10].

2.2 Neuronové sítě a regrese

Koncept dopředné neuronové sítě představuje nelineární regresní model, který obecně zapisujeme jako

$$y = f(\mathbf{x}, \boldsymbol{\beta}) + \varepsilon. \tag{2.18}$$

Nelinearita modelu je způsobena především aktivační funkcí a architekturou sítě. Analytický tvar dopředné neuronové sítě s jednou skrytou vrstvou obsahující Q neuronů, jeden výstupní neuron a zpracovávající P vstupů je

$$y_i = \xi_v \left[\theta_v + \sum_{k=1}^{Q} \beta_k \cdot \varphi_k(\theta_k + \sum_{j=1}^{P} \alpha_{j,k} \cdot x_{i,j}) \right] + \varepsilon_i, \tag{2.19}$$

kde $\xi_v(x)$ je aktivační funkce na výstupním neuronu, $\varphi_k(x)$ jsou aktivační funkce neuronů ve skryté vrstvě, θ_v je práh výstupního neuronu, θ_k jsou prahy neuronů ve skryté vrstvě, β_k a $\alpha_{j,k}$ jsou váhy v síti a ε_i reziduální složka. Proměnné $\{\alpha_{j,k}, \beta_k, \theta_k, \theta_v\}$ představují parametry modelu. V případě použití všech regresorů konstrukční matice $\mathbf{X}$, platí shoda indexů $n = P$ (n z (1.6), P z (2.19)).

Volba $\xi_v(x)$ závisí na tvaru modelované závisle proměnné. Nejčastěji se používá sigmoid a vhodná transformace závisle proměnné na interval $[0, 1]$.

Po zvolení architektury sítě - počtu vstupů, počtu skrytých vrstev, počtu neuronů ve skrytých vrstvách, aktivačních funkcí ve všech vrstvách - řešíme odhad parametrů modelu (2.19) (= učení neuronové sítě). Odhadu parametrů se podrobněji věnujeme v kapitole 3.2.

2.2.1 Neuronové sítě a predikce dynamických řad

Častým případem v praxi je modelovaní závisle proměnné, která má povahu autoregresní posloupnosti a zároveň závisí na dalších faktorech - například finanční časové řady. Předpokládáme časové uspořádání $\{y_t\}_{t=1}^{n}$. Formálně pak můžeme zapsat

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-d}, x_{t,1}, \ldots, x_{t,m}, \boldsymbol{\beta}) + \varepsilon_t, \tag{2.20}$$

kde f je zvolený model - například (2.19), y_{t-i}, kde $i = 1, \ldots, d$ jsou autoregresní členy, $x_{t,j}$, kde $j = 1, \ldots, m$ jsou ostatní regresory, $\boldsymbol{\beta}$ parametry modelu a ε_t je reziduální složka.

2.2.2 Volba modelu f

Při použití lineárních modelů (v případě logistické a poissonovské regrese lineárních v parametrech) se omezujeme na pouze aditivní zpracování vstupních informací. Tento přístup může vyžadovat linearizaci regresorů - například modelovaná závisle proměnná závisí na jednom z regresorů kvadraticky, a proto je vhodné jej nahradit jeho vhodnou transformací. Takové předzpracování dat nemusí být vždy zřejmé, případně vhodné. Například kódování pomocí dummy proměnných je obecně ztrátová komprese informace - například nahrazení kvantitativní

proměnné dummy proměnnou [2, strana 73]. Alternativní přístup ke zpracování nelineárních regresorů nabízí koncept neuronových sítí. Problematikou jejich aproximativních vlastností se zabývalo mnoho autorů. Nejdůležitějším jsou zřejmě důsledky *Kolmogorovovy věty* o reprezentaci spojitých funkcí více proměnných pomocí spojitých funkcí jedné proměnné, především

Lemma 1 *Je-li aktivační funkce spojitá, omezená a nekonstantní na kompaktní množině $X \subset R^n$, pak může dopředná síť s jednou skrytou vrstvou ψ aproximovat každou funkci $g \in C(X)$ libovolně přesně (ve smyslu $|g - \psi| < \varepsilon$ pro libovolně malé ε), je-li k dispozici dostatečně mnoho neuronů ve skryté vrstvě.*

Důkaz a podrobnosti například ve [1, strana 328].

Předchozí lemma nám proto zaručuje dostatečně silný model pro libovolné úlohy typu (1.5) - na rozdíl od lineárních modelů. Počtu neuronů ve skryté vrstvě se věnuje kapitola 3.1.1.

2.2.3 Interpretace modelu

Jednou z nejvíce kritizovaných vlastností je interpretace výsledného modelu. Často je neuronová síť označována za *blackbox*, a to z důvodu složité interpretace nalezených parametrů. Vzhledem k silně nelineární povaze modelu, je interpretace jednotlivých odhadnutých parametrů netriviální, u komplexnějších struktur značně obtížná. Uvažujme neuronovou síť s jednou skrytou vrstvou, obsahující Q neuronů s aktivačními funkcemi typu sigmoid a jedním výstupním neuronem opět s aktivační funkcí sigmoid. Výstup k-tého neuronu ve skryté vrstvě pak odpovídá modelu logistické regrese

$$s_{i,k} = \varphi_k\left(\sum_{j=1}^{P} \alpha_{j,k} x_{i,j} + \theta_k\right) = \frac{1}{1 + e^{\sum_{j=1}^{P} \alpha_{j,k} x_{i,j} + \theta_k}}, \tag{2.21}$$

kde θ_k představuje práh neuronu. V přímém srovnání tak odpovídá parametru β_0 (parametr u konstantního členu) modelu logistické regrese, jehož interpretace (při splnění předpokladu na ne-multikolinearitu uvnitř konstrukční matice a její vycentrování) je modelování průměrné úrovňového hodnoty targetu y. V modelu dopředné neuronové sítě lze interpretovat parametr θ_k podobně. V terminologii neuronových sítí představuje úrovňovou hodnotu signálů vycházející z k-tého neuronu skryté vrstvy.

Podobně můžeme interpretovat parametr prahu θ_v výstupního neuronu. Protože výstupní neuron opět odpovídá modelu logistické regrese

$$y_i = \xi_v\left(\sum_{k=1}^{Q} \beta_k s_{i,k} + \theta_v\right) = \frac{1}{1 + e^{\sum_{k=1}^{Q} \beta_k s_{i,k} + \theta_v}}, \tag{2.22}$$

odpovídá θ_v opět úrovňovému parametru β_0 logistické regrese. Praktické výpočty ovšem ukazují, že na výstupním neuronu je často významná multikolinearita signálů přicházejících ze skryté vrstvy. Je nutné proto s interpretací θ_v jakožto úrovňové hodnoty nakládat opatrně a nejprve studovat strukturu přicházejících

signálů. Tomuto případu se věnujeme v simulačním příkladu 5.3.

Nyní můžeme přistoupit k interpretaci parametrů β_i neuronů ve výstupní vrstvě a $\alpha_{j,k}$ ve skryté vrstvě. V modelu logistické regrese (2.5) můžeme označit

$$\pi(x) = P(Y_i = 1|\mathbf{x_i}, \boldsymbol{\beta}) = 1 - F(-\mathbf{x_i}\boldsymbol{\beta}) = 1 - \frac{e^{-\mathbf{x_i}\boldsymbol{\beta}}}{1 + e^{-\mathbf{x_i}\boldsymbol{\beta}}} = \frac{e^{\mathbf{x_i}\boldsymbol{\beta}}}{1 + e^{\mathbf{x_i}\boldsymbol{\beta}}} \tag{2.23}$$

a zavést

$$g(x) = \ln \frac{\pi(x)}{1 - \pi(x)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i x_i + \cdots \tag{2.24}$$

Výraz

$$\frac{\pi(x)}{1 - \pi(x)} \tag{2.25}$$

se často označuje jako *šance*. Funkce $g(x)$ pak představuje jeho logaritmickou transformaci, často označovanou jako *logit*. Pro interpretaci parametru β_i použijeme souvislost právě s funkcí logit. Pokud se změní hodnota regresoru x_i při neměnnosti ostatních proměnných o hodnotu 1, pak se změní hodnota funkce logit o hodnotu parametru β_i. Stejně (až na již zmíněnou multikolinearitu) můžeme interpretovat parametr β_i modelu výstupního neuronu (2.22). *Vstupní* proměnné v tomto případě představují signály přicházející z předchozí skryté vrstvy. Analogické proměnné $\alpha_{j,k}$ v neuronech ve skrytých vrstvách můžeme interpretovat stejně. Vstupní proměnné do těchto neuronů představují skutečné nezávislé proměnné a výstupní hodnota (= ekvivalent závisle proměnné modelu logistické regrese) představuje signál přenášený do další vrstvy.

3. Výpočetní aspekty neuronových sítí

3.1 Volba architektury sítě

V případu dopředné sítě je nutné řešit několik otázek, které charakterizují architekturu sítě. Především

- počet skrytých vrstev a počty neuronů v jednotlivých vrstvách
- počet a volba vstupů do první vrstvy
- tvar aktivačních funkcí

3.1.1 Počet skrytých vrstev a jejich neuronů

Volba počtu skrytých vrstev se většinou redukuje na výběr z jedné nebo dvou vrstev. Uvedené lemma 2.2.2 zaručuje, že neuronová s jednou skrytou vrstvou je dostatečně silný aproximátor. Vzhledem k riziku přeučení viz. kapitola 3.2.2 se proto dál omezíme pouze na jednu skrytou vrstvu.

Volba počtu neuronů ve skryté vrstvě spolu s volbou počtu vstupů do sítě určují celkový počet parametrů, a tedy *volnost* (aproximativní schopnost) sítě. Počet neuronů ve skryté vrstvě je proto často volen s ohledem na velikost učícího vzorku a počtu vstupů. V praxi se obvykle volí počty heuristicky - počet neuronů je o něco větší než počet vstupů.

3.1.2 Počet a volba vstupů do první vrstvy

Jedna z klíčových otázek je počet a volba vstupních regresorů. Vzhledem k složitější časové náročnosti učícího procesu odpadá možnost vyzkoušet všechny kombinace. Jedna z variant pro situaci s velkým počtem regresorů je použití *Stepwise regrese* a výsledné regresory použít i pro neuronovou síť. I když tato možnost zaručuje *kvalitu* zvolených regresorů, snižuje možnost využít nelineární závislosti a souvislosti, které Stepwise algoritmus implicitně nehledá. Lineární model - například logit, je totiž lineární v parametrech. I přes tuto nevýhodu je výběr proměnných pomocí Stepwise algoritmu častá varianta. Jinou metodu, která používá všechny proměnné, představíme v kapitole 4.4.

3.2 Učící proces

Odhadu parametrů modelu se v terminologii neuronových sítí říká učící proces. Existuje řada algoritmů pro dopředný typ sítě. Pro numerické příklady byl použit software *Mathematica* a knihovna pro neuronové sítě *Neural Package*. Níže uvedené algoritmy jsou převzaty z její dokumentace - [3]. Společným prvkem všech uvedených algoritmů je metoda nejmenších čtverců, a tedy minimalizace chybové funkce

$$E_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{\beta}, \mathbf{x_i}))^2, \tag{3.1}$$

kde $\boldsymbol{\beta}$ jsou veškeré parametry modelu a $\mathbf{x_i}$ je vektor vstupů. Vzhledem k takto definované účelové funkci pak odhad parametrů modelu je

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ E_n(\boldsymbol{\beta}). \tag{3.2}$$

Všechny algoritmy v knihovně *Neural Package* jsou iterační. Počátečním bodem je náhodně zvolený $\boldsymbol{\beta}^0$. Trénovací algoritmus iteračně snižuje $E_n(\boldsymbol{\beta})$ inkrementálními změnami hodnot $\boldsymbol{\beta}^j$ vždy proti směru gradientu následovně

$$\boldsymbol{\beta}^{j+1} = \boldsymbol{\beta}^j + \mu\mathbf{R}\nabla_{\boldsymbol{\beta}}E_n, \tag{3.3}$$

kde μ je parametr ovlivňující velikost změny $\boldsymbol{\beta}$. Matice $\mathbf{R}$ ovlivňuje směr pohybu a společně s gradientem $\nabla_{\boldsymbol{\beta}}E_n$ (vektoru parciálních derivací funkce E_n dle jednotlivých složek parametru $\boldsymbol{\beta}$) určuje $\boldsymbol{\beta}$ v dalším kroku. Jednotlivé algoritmy se pak od sebe liší ve volbě matice $\mathbf{R}$ a parametru μ.

Levenberg-Marquardt

Je výchozím algoritmem v knihovně *Neural Package*. Algoritmus volí

$$\mathbf{R} = (H + e^{\lambda}I)^{-1} \text{ a } \mu = 1, \tag{3.4}$$

kde H je matice druhých derivací - tzv. *Hessova matice*

$$\frac{d^2E_n(\boldsymbol{\beta})}{d\boldsymbol{\beta}^2} = \frac{2}{n}\sum_{i=1}^{n}\nabla_{\boldsymbol{\beta}}f(\boldsymbol{\beta}, \mathbf{x_i})\nabla_{\boldsymbol{\beta}}f(\boldsymbol{\beta}, \mathbf{x_i})^T - \frac{2}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{\beta}, \mathbf{x_i}))\nabla^2_{\boldsymbol{\beta}}f(\boldsymbol{\beta}, \mathbf{x_i}), \tag{3.5}$$

kde $\nabla^2_{\boldsymbol{\beta}}f(\boldsymbol{\beta}, \mathbf{x_i})$ je matice druhých parciálních derivací modelu f dle jednotlivých složek vektoru parametru $\boldsymbol{\beta}$. Spočtená Hessova matice však obecně nemusí být ve všech bodech $\boldsymbol{\beta_j}$ pozitivně definitní [3], a tak změna $\boldsymbol{\beta}$ může jít ve směru gradientu, čímž by se zvýšil minimalizovaný součet čtverců. Proto se používá alternativa ve formě pouze první části této matice

$$H = \frac{2}{n}\sum_{i=1}^{n}\nabla_{\boldsymbol{\beta}}f(\boldsymbol{\beta}, \mathbf{x_i})\nabla_{\boldsymbol{\beta}}f(\boldsymbol{\beta}, \mathbf{x_i})^T. \tag{3.6}$$

Parametr λ v (3.4) je volen automaticky. V každém kroku jsou zkoušeny takové hodnoty λ, aby došlo ke snížení hodnoty minimalizované účelové funkce. Tento algoritmus je často preferován díky schopnosti vypořádat se *špatně podmíněnou* ([27]) úlohou minimalizace na rozdíl například od dalšího algoritmu.

Gauss Newton

Tento algoritmus volí $R = H^{-1}$ a parametr $\mu = 1$. GN je rychlý a spolehlivý algoritmus pro řadu optimalizačních úloh. Pro neuronové sítě je z důvodu špatné podmíněnosti úlohy preferován předchozí algoritmus.

Steepest descent

Tento algoritmus volí $R = I$, což určuje směr iterace pouze proti směru gradientu. Parametr μ je v každém kroku zdvojnásoben. Ve srovnání s výše uvedenými algoritmy je rychlejší, ale mnohem méně efektivní viz. například [15] nebo [16].

Genetické algoritmy

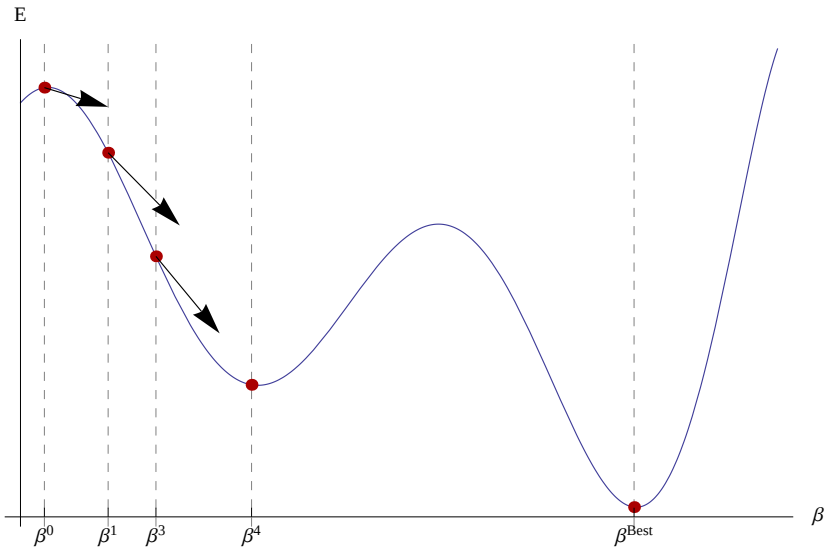
Jiným možným přístupem jsou genetické algoritmy. Jejich výhodou při učení neuronové sítě je absence požadavku na spojitost modelu neuronové sítě (spojitost aktivačních funkcí uvnitř neuronů). Za aktivační funkci může být například volena funkce Sign(x)

$$\text{Sign(x)} = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0, \end{cases} \tag{3.7}$$

která nejvíce odpovídá biologické předloze. Zásadní nevýhodou je ale rychlost. Ve srovnání s výše uvedenými metodami je z principu generování a vyhodnocování velkého počtu jedinců v populaci pomalejší.

3.2.1 Stabilita a lokální řešení

Vzhledem ke složitosti matematického modelu neuronové sítě je učící proces netriviální optimalizační problém. Iterační algoritmy často naráží na lokální minima, ve kterých učící proces ukončí. Takovou situaci například zachycuje obrázek 3.1.



Obrázek 3.1: Lokální minimum v jednorozměrném případu.

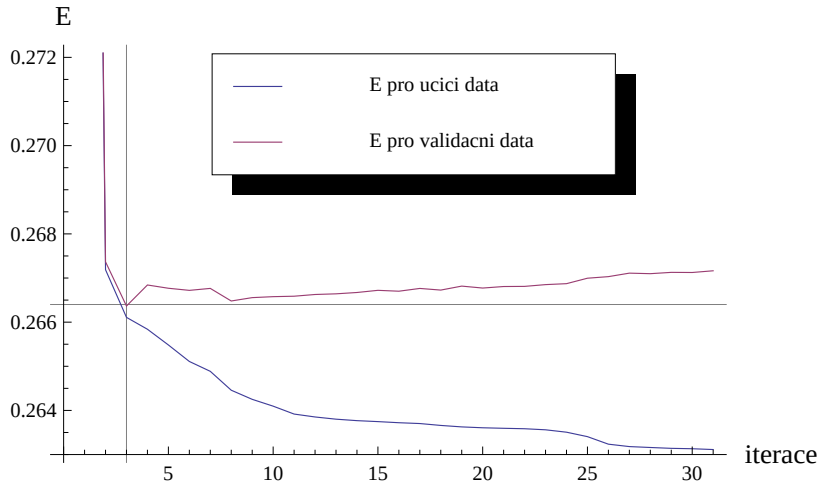
Důsledek takového chování je *nestabilita* neuronových sítí. Především fakt, že dva běhy učícího procesu na stejných datech (ale z různých startovacích bodů) skončí ve dvou různých lokálních řešení, je často kritizovaná vlastnost. Proto je například nutné pouštět učící proces opakovaně z různých počátečních bodů, což vzhledem k výpočetní, a tím i časové, náročnosti komplikuje možnost provést řadu pokusů jako například u *Stepwise regrese*.

3.2.2 Přeučení

Jedním z nejsložitějších problému při učení neuronové sítě je riziko přeučení (overfitting). Přeučení představuje situaci, kdy učící proces přestává extrapolovat vztahy a souvislosti obsažené v datech a namísto toho začíná data interpolovat - učit se je zpaměti. Složitější struktura neuronových sítí, na rozdíl například od lineárních modelů, obsahuje vyšší počet (třeba i řádově) odhadovaných parametrů, díky čemuž je tomuto problému více náchylná. Tradiční dva postupy pro vypořádání se s přeučením jsou *rozdělení vzorku* a *křížová validace*. Velmi účinný algoritmus proti přeučení s využitím ensemblovacích technik prezentoval J. Howard [17] - budeme dále popisovat v kapitole 4.4.

Rozdělení vzorku

Tento přístup spočívá v rozdělení datového vzorku na *učící část*, *validační část* a *testovací část*. Zvolený algoritmus řeší minimalizační úlohu vzhledem k učícímu vzorku - výpočet E_L. Zároveň je počítána střední čtvercová chyba i pro validační vzorek E_V. V případě, že během iteračního algoritmu (v iteraci i) E_V^i roste zatímco E_L^i klesá, je algoritmus zastaven a za odhad parametrů $\widehat{\boldsymbol{\beta}}$ jsou považovány takové hodnoty, kde E_V^i je minimální. Po skončení procesu učení je spočtena chyba i pro testovací vzorek E_T. Testovací vzorek je plně nezávislý na učícím procesu. Slouží jako kontrola pro případ, kdy je algoritmus několikrát pro stejný učící a validační vzorek restartován. Opakované spouštění algoritmu a výběr na základě validačního vzorku po čase vede totiž opět k přeučení modelu, tentokrát i vzhledem k validačnímu vzorku. Proto je použití všech vzorků velmi žádoucí.



Obrázek 3.2: Optimální bod iteračního algoritmu v případě postupu *rozdělení vzorku*.

Křížová validace

Jiný přístup kontroly přeučení představuje metoda křížové validace (cross-validation). Tato metoda opět spočívá v rozdělení datového vzorku - a to na K nezávislých částí (často $K = 10$). Učící algoritmus je proběhnut K-krát se vždy jednou vynechanou částí použitou pro validaci. Za odhad parametrů modelu je

prohlášen například průměr hodnot těchto odhadnutých parametrů ze všech K běhů učícího algoritmu. Tento přístup může být nebezpečný, a to díky obecně různým startovacím hodnotám pro každý z K výpočtů. Stejné neurony totiž pro každý z K výpočtů mohou asociovat různé informace a následné průměrování hodnot poté může vést k nesmyslným hodnotám vah. Jiný přístup je průměrování jednotlivých predikcí, ale tato technika je lépe rozpracována v další popsané technice v kapitole 4.4.

V této práci bylo použito první zmíněné techniky - rozdělení vzorku. Metoda křížové validace je totiž z principu i K násobně pomalejší.

3.3 Pokročilejší techniky

V této kapitole krátce představíme některé další techniky, které mohou zvýšit prediktivní sílu konceptu neuronových sítí (ve smyslu dosažení lepších hodnot optimalizované účelové funkce). I když literatura a odborné články kolem neuronových sítí představující širokou oblast, omezíme se jen na nejznámější techniky.

3.3.1 Boosting

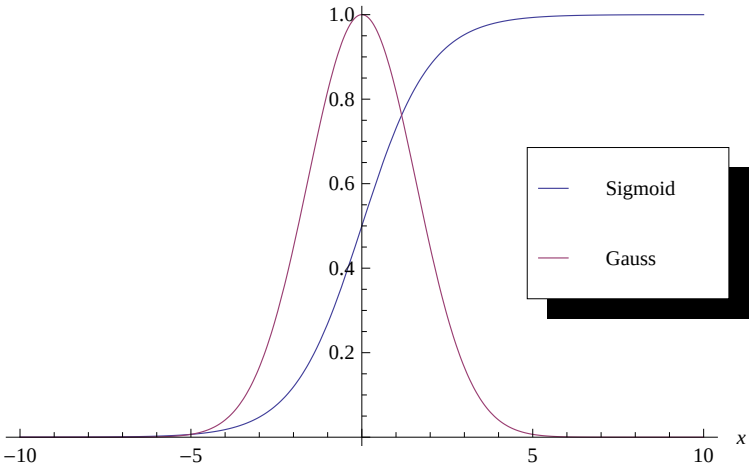
Metoda *boosting* představuje vylepšení algoritmů strojového učení, kdy je špatně naučeným vzorům zvýšena váha v rámci minimalizované chybové funkci a učící algoritmus je proběhnut opakovaně. Existuje řada algoritmů a jednotlivých implementací. Nejznámější je algoritmus *ADAboost* poprvé představen ve [4]. Většina nových implementací z *Adaboost*-u vychází. Algoritmus není v knihovně *Neural Network* implementován. Metoda boosting nebyla v této práci testována.

3.3.2 Neuronové sítě s radiálními jednotkami

Neuronové sítě s radiálními (nebo také lokálními) (například [1, strana 117]) jednotkami představují mladší model. Jde opět o dopřednou topologii s jednou skrytou vrstvou. Zatímco se již představená dopředná sít snaží rozdělit výstupní prostor na dva podprostory - viz možné tvary aktivačních funkcí na obrázku 1.3, radiální jednotky lokalizují výstup do okolí bodu určeného svými parametry - například Gaussova funkce

$$f(z) = e^{-\left(\frac{z}{a}\right)^2}, \quad a \geq 0. \tag{3.8}$$

Rozdíl proti dopředné síti je tedy především v použití jiné aktivační funkce uvnitř neuronů, a tedy výpočtu přenášeného signálu, který je dán tvarem aktivační funkce. Tento model je ovšem méně používán ve prospěch algoritmu *Support vector machine* (SVM) ([20], [21]), jenž se mnohem lépe vypořádává s problémy lokálních minim a při použití vhodné jádrové funkce (například již zmíněná Gaussova funkce) se neuronové síti s radiálními jednotkami velmi podobá. Pro SVM rovněž existují dvě velmi kvalitní a rychlé implementace libSVM ([18]) a lightSVM ([19]).



Obrázek 3.3: Aktivační funkce sigmoid a gaussova funkce.

3.3.3 Modulární neuronové sítě

Koncept modulárních neuronových sítí představuje další oblíbenou techniku. Využívá opět biologické předlohy, kdy se jednotlivé části lidského mozku zaměřují na různé úkony. Modulární sítě představují koncept, kdy je postaveno několik málo sítí, které řeší různé úkoly. Taková specializace má za úkol zvýšit extrapolační schopnost celkového modelu. Koncept je podobný *Lesům neuronových sítí*, popisovaných v kapitole 4.4. Na rozdíl od lesů sítí, kde jsou jednotlivé modely trénovány bez ohledu na ostatní, jsou jednotlivé sítě v rámci jednoho modulárního modelu na sobě přímo závislé. Obě techniky lze kombinovat - postavením lesu modulárních neuronových sítí. Takový koncept představuje velmi robustní systém. Počet neuronů u modulárních sítí, na rozdíl od jedné dopředné sítě, může dosahovat stovek až tisíců - v případě lesů modulárních sítí až stovek tisíců.

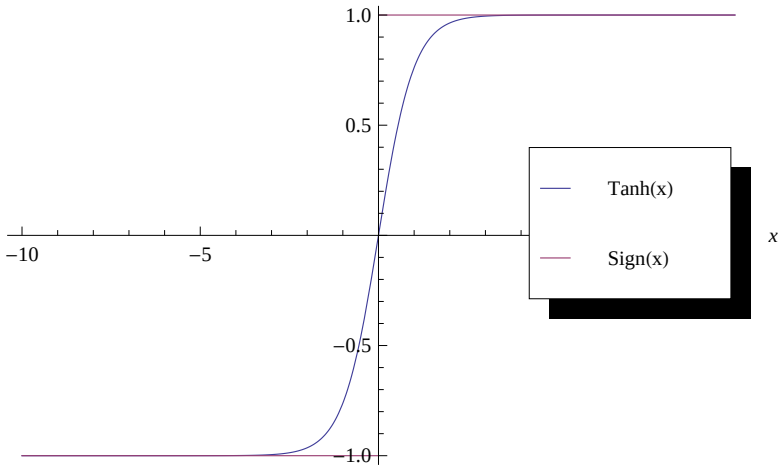
3.3.4 Použití genetických algoritmů

Použití genetických algoritmů pro modely neuronových sítí má nejčastěji dvojí použití. První případ je použití jakožto učící algoritmus pro předem zvolenou topologii. Druhý případ představuje použití pro optimalizaci samotné topologie neuronové sítě.

GA jako učící proces

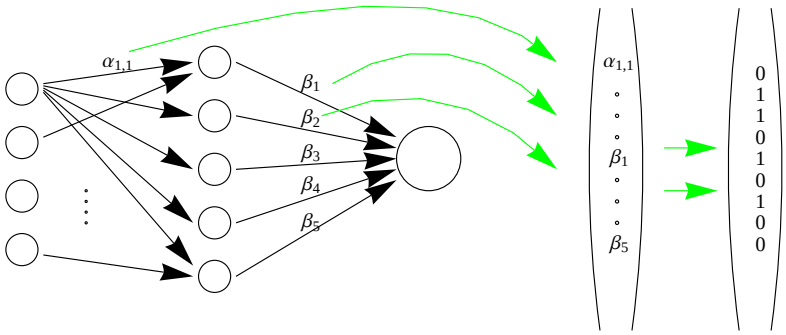
GA jakožto učící proces představuje alternativu pro algoritmy představené v kapitole 3.2. Jeho výhodou je možnost použít nespojitou, a tedy nediferencovatelnou topologii, která lépe vyhovuje biologické předloze. Umožňuje totiž použít nespojitou aktivační funkci - například Sign(x), která lépe odpovídá vlastnosti excitace (vybuzení) informace jako u biologického neuronu. Účelovou funkci $u(\boldsymbol{\beta})$ optimalizační úlohy volíme často podobně jako u tradičních algoritmů - minimalizace střední čtvercové chyby

$$\min_{\boldsymbol{\beta}} u(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{\beta}, \mathbf{x_i}))^2. \tag{3.9}$$



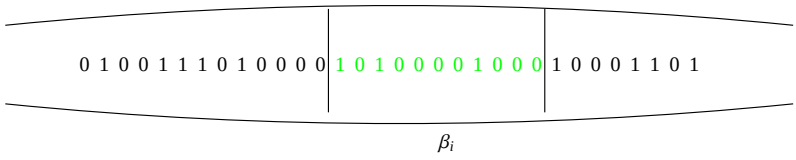
Obrázek 3.4: Srovnání aktivační funkce Tanh(x) a Sign(x).

Jednotlivec populace v každém iteračním kroku reprezentuje celkovou parametrizaci neuronové sítě. Všechny parametry neuronové sítě jsou tedy převedeny na zvolenou reprezentaci jedinců v rámci populace - například vektor nul a jedniček.



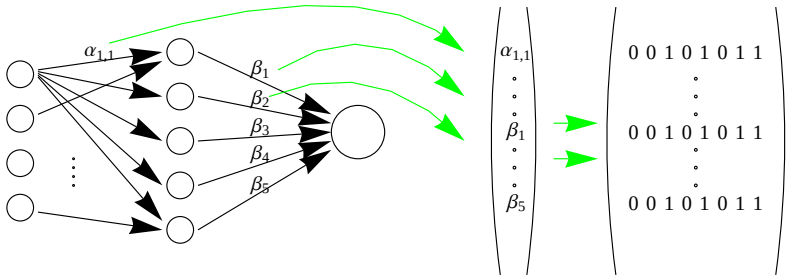
Obrázek 3.5: Transformace parametrů sítě na jedince populace.

Jednotlivé sekvence v rámci tohoto vektoru pak reprezentují samotné parametry sítě.



Obrázek 3.6: Část sekvence, reprezentující hodnotu parametru β_i.

Vzhledem k vysokému počtu vah uvnitř neuronových sítí lze použít například reprezentace kódování jedinců pomocí matic.
Po zvolení reprezentace je nutné zvolit způsob kódování a zpětného překladu. V praxi se nejčastěji používají dvě metody



Obrázek 3.7: Reprezentace pomocí matice.

- Jednoduché binární kódování
- Grayovo binární kódování

Jednoduché binární kódování představuje použití překladu hodnot do binární soustavy, kde například hodnoty před a za desetinou čárkou mají přesně určené sekvence v rámci zvolené reprezentace. Grayovo binární kódování představuje číselný systém, kde dvě po sobě následující hodnoty se liší právě o jeden bit.

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad \text{x} \quad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Obrázek 3.8: Grayovo kódování pro 2bitové, 3bitové a 4bitové řetězce a jednoduché binární kódování pro 4bitové řetězce

Grayovo kódování umožňuje v rámci GA algoritmů citlivější, a tedy můžeme předpokládat rychlejší konvergenci optimalizační úlohy. Nicméně pro úlohy s velkým počtem lokálních extrémů může Grayovo kódování konvergovat naopak pomaleji. Srovnání rychlosti konvergence obou metod ověříme v simulační úloze 5.5. Použití genetických algoritmů jakožto algoritmu učení neuronové sítě ovšem představuje časově extrémně náročnou úlohu.

Pro ilustraci - pokud bychom v obrázku 3.8 v případech matic 4bitových řetězců očíslovali jednotlivé řádky od hodnot 0 až po 15, pak by se v případě Grayova

kódování lišil řetězec reprezentující číslo 7 od řetězce čísla 8 na právě jedné pozici: $7 \rightarrow \{0,1,0,0\}$ a $8 \rightarrow \{1,1,0,0\}$. Zatímco v případě jednoduchého binárního kódování by se lišil hned ve všech čtyřech pozicích: $7 \rightarrow \{0,1,1,1\}$ a $8 \rightarrow \{1,0,0,0\}$.

Optimalizace topologie použitím GA

Jinou možností použití GA při modelování neuronových sítí je optimalizace struktury neuronové sítě. Často se zároveň řeší úloha výběru vhodných regresorů do modelu. V takovém případě je nutné zkonstruovat účelovou funkci, která dokáže zohlednit všechny faktory a přitom se dokáže vypořádat s problémem přefitování. Postupovat můžeme podobně jako v případě učení neuronové sítě. Datový vzorek rozdělíme na učící a monitorovací vzorek. Učící část je v každém kroku algoritmu vyhodnocována účelovou funkcí a evoluce populace probíhá fitováním na tento vzorek. Monitorovací vzorek má roli validačního vzorku z učení neuronových sítí. Je rovněž pravidelně vyhodnocován a zastaví nám GA v případě, že hodnoty účelové funkce na tomto vzorku začínají růst. Protože učící proces neuronových sítí naráží na problém lokálních minim, každý prvek populace z nové počáteční inicializace neuronové sítě naučíme několikrát - například 50x.

Simulaci tohoto způsobu použití GA představíme v příkladu 5.6. Účelovou funkci $v(\boldsymbol{\beta}, \boldsymbol{\pi})$ tedy můžeme konstruovat analogicky jako v předchozím případě

$$\min_{\boldsymbol{\pi}} v(\boldsymbol{\beta}, \boldsymbol{\pi}) = \min_{\boldsymbol{\pi}} \frac{1}{J} \sum_{j=1}^{J} \frac{1}{n^*} \sum_{i=1}^{n^*} [y_i - f^*(\boldsymbol{\beta_j}, \mathbf{x_i}, \boldsymbol{\pi})]^2, \tag{3.10}$$

kde $f^*(x)$ představuje zobecnění modelu 2.19 o parametrizaci $\pi \in \Pi$, kde Π představuje celkovou možnou množinu parametrizací popsanou níže. Index i, kde $i = 1, \ldots, n^*$ je označení pozorování v rámci konstrukční matice učícího vzorku. Pro každého člena populace je učící algoritmus opakován J-krát. Indexujeme pomocí j, kde $j = 1, \ldots, J$. V každém opakování j odhadneme obecně jiné hodnoty parametrů $\boldsymbol{\beta_j}$.

Jednotlivci v rámci populací nyní reprezentují parametry topologie a výběr regresorů. Jako reprezentaci může použít vektor nul a jedniček. Kódování je nutné nyní volit takové, aby každý jedinec vzniklý genetickými operátory opět představoval možné řešení. U neuronových sítí můžeme uvažovat následující možná nastavení (při zafixování počtu skrytých vrstev na jednu - což obecně není nutné).

- Počet neuronů ve skryté vrstvě
- Počet vstupních regresorů
- Tvar aktivační funkce ve skrytých vrstvách
- Tvar aktivační funkce výstupního neuronu (a tedy transformace targetu)
- Výběr konkrétních regresorů
- Parametry optimalizačních učících algoritmů

- Použití dalších technik a jejich parametrů - boosting, centrování dat, prořezávání sítě
- a další..

Použití genetických algoritmů v sobě implicitně nese spoustu práce, kterou analytik poté nemusí provádět. GA vyzkouší velkou řadu kombinací, na které by se v rámci vývoje modelu vůbec nemuselo přijít. Manuální zkoumání jednotlivých nastavení samostatně nám sice může dát heuristický odhad jak model postavit, nicméně nám nic neříká o vzájemné interakci jednotlivých nastavení. Proto jsou genetické algoritmy často používaným trikem na odhalení správných kombinací nastavení.

4. Rozšiřující techniky

V této kapitole představíme a navrhneme techniky, které mají ambici lépe vysvětlit nebo vylepšit představený model neuronové sítě. Jednotlivé techniky budeme simulačně testovat v následující kapitole. Jednotlivé techniky mají vnést pohled především na přenos informace ze skryté vrstvy na výstupní neuron.

4.1 Dvoustupňový odhad parametrů

Přenos signálů ze skryté vrstvy na výstupní neuron představuje model lineární v parametrech

$$y_i = \xi_v \left(\theta_v + \sum_{k=1}^{Q} \beta_k s_{i,k} \right) + \varepsilon_i, \tag{4.1}$$

kde

$$s_{i,k} = \varphi_k \left(\theta_k + \sum_{j=1}^{P} \alpha_{j,k} x_{i,j} \right). \tag{4.2}$$

Můžeme tak zkoumat vlastnosti submodelu (4.1). Označme matici

$$\mathbf{S} = \{s_{i,k}\}, \quad i = 1, \dots n, \quad k = 1, \dots Q, \tag{4.3}$$

kde jednotlivé složky $s_{i,k}$ jsou šířící se signály ze skryté do výstupní vrstvy. Skrytá vrstva neuronové sítě představuje zobrazení z $\mathbf{R^P} \rightarrow \mathbf{R^Q}$, které nelineárně transformuje původní matici regresorů na matici signálů $\mathbf{X} \rightarrow \mathbf{S}$. Na rozdíl od faktorové analýzy nebo metody hlavních komponent je transformace jednak nelineární a nové sloupce matice jsou konstruovány s respektem na závisle proměnnou. Navíc se předpokládá, že volená dimenze matice $\mathbf{S}$ je vyšší než dimenze původní matice regresorů $\mathbf{X}$.

Díky nelineárním transformacím ve skryté vrstvě je učení neuronové sítě netriviální optimalizační proces. I když učící algoritmus zpracovává signály ze skryté vrstvy na výstupním neuronu lineárně v parametrech, simulace 5.2 ukazuje, že ne zcela optimálně. Tento *defekt* v rámci učícího procesu vysvětlujeme nastavením optimalizační procedury, která primárně extrahuje nelineární souvislosti ve skryté vrstvě. Dalším důvodem může být volba účelové funkce v úlohách klasifikace, kdy místo metody nejmenších čtverců může lepších výsledků dosáhnout účelová funkce metody maximální věrohodnosti. Proto je navržen dvoukrokový odhad parametrů, který tuto vadu odstraňuje.

4.1.1 Algoritmus TwoStepNNL

Krok 1. Odhad parametrů $\{\alpha_{j,k}, \beta_k, \theta_k, \theta_v\}$ v modelu (2.19) výchozím učícím algoritmem - procedurou Levenberg-Marquardt (3.2). Pro připomenutí uvedeme rovnici modelu (2.19)

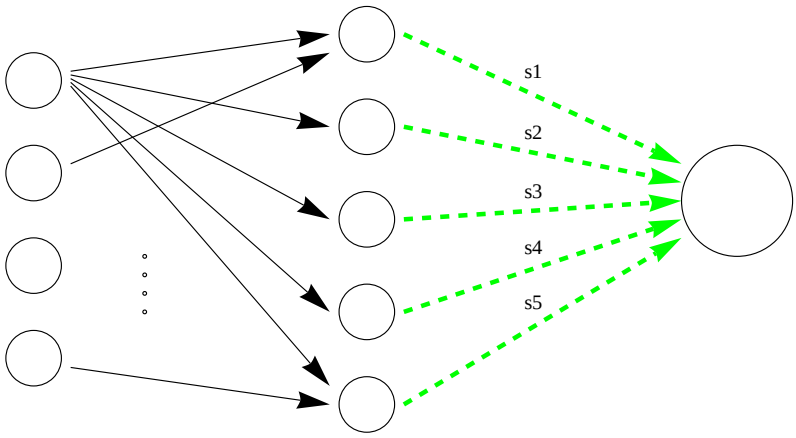
$$y_i = \xi_v \left[\theta_v + \sum_{k=1}^{Q} \beta_k \cdot \varphi_k(\theta_k + \sum_{j=1}^{P} \alpha_{j,k} \cdot x_{i,j}) \right] + \varepsilon_i.$$

Krok 2. Odhad parametrů θ_v, β_k, $k = 1, \ldots, Q$ příslušným lineárním modelem v závislosti na zvolenou aktivační funkci výstupního neuronu $\xi_v(x)$. Například použití logistické regrese, model (2.5), kdy namísto původní matice regresorů $\mathbf{X}$ použijeme matici šířících se signálů $\mathbf{S}$. První krok tedy použijeme na vytvoření nových regresorů - matice $\mathbf{S}$.

Krok 3. Nahrazení odhadů parametrů θ_v, β_k, $k = 1, \ldots, Q$ v neuronové síti hodnotami nalezenými v kroku 2.
V rámci simulace 5.2 ověřujeme oprávněnost tohoto postupu.

4.2 Multikolinearita signálů

Na každý neuron ve skryté vrstvě přichází stejné vstupní informace. V případě volby počtu neuronů ve skryté vrstvě vyšší než počtu regresorů vstupujících do modelu předpokládáme nelineární nebo skryté závislosti ve vstupních datech. V případě neplatnosti výskytu takových závislostí nebo zvolenému příliš vysokému počtu neuronů ve skryté vrstvě narazíme na problém multikolinearity signálů, vstupujících do výstupního neuronu.



Obrázek 4.1: Šíření signálu ze skryté vrstvy.

Multikolinearita představuje vzájemnou vysokou korelovanost regresorů - v našem případě signálů. Vysoká míra vzájemné korelace signálů způsobuje problémy při odhadu parametrů a následné interpretaci odhadnutých parametrů modelu. Zkoumání multikolinearity můžeme založit na výběrové korelační matici corr(S) matice **S** (4.3). V případě, že maximum absolutních hodnot nediagonálních prvků výběrové korelační matice přesahuje 0,8 můžeme předpokládat výskyt multikolinearity.

V simulaci 5.3 zkoumáme míru multikolinearity mezi skrytou a výstupní vrstvou založenou na metodě VIF (Variance Inflation Factor) [26]. VIF definujeme jako

$$\mathrm{VIF_j} = \frac{1}{1 - R_j^2}, \tag{4.4}$$

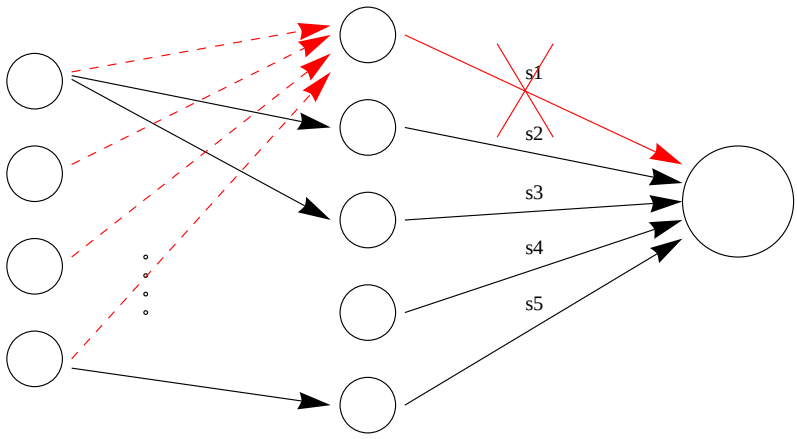
kde R_j^2 je koeficient determinace regrese j-tého regresoru na zbylých regresorech

$$X_j = a_0 + a_1 X_1 + \cdots + a_n X_n + \varepsilon. \tag{4.5}$$

I když neexistuje přesná hodnota VIF, při které bychom mohli identifikovat multikolinearitu v datech, řada autorů za multikolinearitu prohlašuje případ, kdy $\mathrm{VIF_j} > 10$ pro nějaké j. I když [26] ukazuje možné problémy takového přístupu, v simulaci 5.3 tento postup použijeme.

4.3 Prořezávání sítě

Dvoustupňový odhad parametrů představený v kapitole 4.1 nám rovněž umožňuje statisticky ověřit, zda některé signály ze skryté vrstvy nejsou zbytečné a jejich případné vynechání z modelu. Mluvíme o tzv. prořezávání neuronové sítě (*pruning*). Při prořezávání parametrů (spojů v topologii) se díky dopřednosti můžeme omezit na prořezávání signálů mezi skrytou a výstupní vrstvou. Vynecháním některého ze signálů (a tedy jednoho skrytého neuronu) rovněž rušíme vliv všech parametrů směřujících do vynechaného neuronu.



Obrázek 4.2: Prořezání neuronové sítě.

Prořezávání založíme na testu významnosti parametrů submodelu 4.1. Postupujeme jako v případě sestupné varianty metody Stepwise regrese. V simulaci 5.4 ověříme takto navržené prořezávání.

4.4 Lesy neuronových sítí

Lesy neuronových sítí představují koncept často využívaný u jiné techniky strojového učení - rozhodovacích stromů - proto terminologie *lesy* (například [5]). Les modelů představuje množinu nezávisle stavěných modelů (prediktorů) - v našem případě neuronových sítí. Jednotlivé sítě jsou trénovány bez vzájemné interakce s jinými sítěmi a poté vhodně kombinovány - nejčastěji se vezme průměr predikcí (například v případě, že modelovaná závislá proměnná má spojité rozdělení). Jinou možností je například většinové hlasování (klasifikační úlohy). Kombinování prediktorů byla například jedna z vítězných technik známé úlohy zveřejněné společností *Netflix* v roce 2006 [6]. Cílem soutěže bylo najít model, který bude převyšovat prediktivní sílu modelu společnosti Netflix o 10%.

Tato podkapitola je inspirována především vystoupením J.Howarda (dále J.H.) se svým příspěvkem na konferenci v Melbourne [17]. J.H. zde prezentoval algoritmus, se kterým se mu podařilo zvítězit v několik data-miningových soutěží pořádaných internetovou službou *Kaggle*. Tato služba umožňuje subjektům zadat data-miningovou úlohu a nechat zájemce hledat co nejlepší řešení. Autor nejlepšího řešení je poté zadavatelem odměněn. J.H. se jako řešitel účastnil několika takových soutěží, poté přešel do řad pořadatelů a správců této internetové služby.

Jeho algoritmus vychází z metod *RandomForest* a *Random Subspace* s použitím klasifikátoru - neprořezaný rozhodovací strom (unprunning decision tree). V této práci vyzkoušíme použít jeho metodu s nahrazením rozhodovacích stromů zkoumanou metodou neuronových sítí.

4.4.1 J.Howard - RandomForest

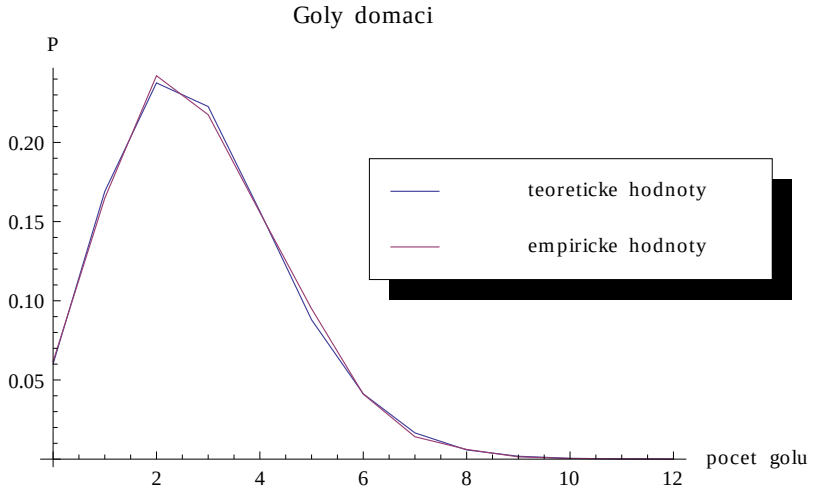
Základní myšlenka tohoto modelu vychází z práce [25], kdy generujeme velký počet vzájemně nezávislých modelů - neprořezaných rozhodovacích stromů (dále jen stromů) - J.H. uvádí řádově například $D = 1000$. Pokud má modelovaná náhodná veličina spojité rozdělení, pak jako její odhad volíme průměr jejich predikcí od jednotlivých stromů. Jedna z hlavních myšlenek tohoto modelu je co největší randomizace s cílem o co největší variabilitu (především na straně použitých dat). Proto při stavbě k-tého modelu nestavíme jednotlivé stromy vždy s použitím identické matici $\mathbf{X}$ (1.6), ale pro stavbu každého stromu generujeme jinou matici $\mathbf{X_k^L}$. Označme $\mathbf{X_k'}$ matici, která je tvořena pouze náhodně vybranou podmnožinou sloupců původní matice $\mathbf{X}$. Z matice $\mathbf{X_k'}$ nyní náhodně vybereme podmnožinu řádků, kterou označíme $\mathbf{X_k^L}$. Tato matice je poté použita pro učící proces zvoleného modelu. Doplněk matice $\mathbf{X_k^L}$ do matice $\mathbf{X_k'}$ označme $\mathbf{X_k^V}$. Tato matice hraje klíčovou roli při řešení problému přeučení. Jakmile je k-tý model postaven je aplikován právě na tuhle matici $\mathbf{X_k^V}$. Jelikož tato matice nevstupovala do procesu stavby modelu, jsou odhady závislých proměnných y_i^k příslušné k řádkům této matice implicitně nepřeučené. Stejný postup opakujeme D-krát. Pro každý strom, příslušnou matici $\mathbf{X_k^V}$ a odhady $\widehat{y_i^k}$ si vždy zapamatujeme pouze predikce pro tuto nezávislou (vzhledem k postavenému modelu v k-tém kroku) matici $\mathbf{X_k^V}$. Za výslednou predikci proměnné y_i volíme průměr dosažených predikcí (pouze vzhledem k maticím $\mathbf{X_k^V}$). Vzhledem ke konstrukci matic $\mathbf{X_k^V}$ může ke každému pozorování i příslušet jiný počet dílčích predikcí, což vzhledem k vyššímu počtu postavených modelů není problém.

5. Simulační příklady

Výchozí úlohou pro všechny simulace je úloha předpovědi počtu gólů v hokejových zápasech. Pro simulace byla použita data z nejvyšší Kanadsko-americké hokejové soutěže za posledních 10 sezón. K dispozici tak máme 14 654 odehraných zápasů. Pro každý zápas (pokud to je možné) bylo spočítáno 56 charakteristik - regresorů, popis jednotlivých regresorů najdeme v tabulkách 5.11 a 5.12. Tyto charakteristiky obsahují předem známé informace o jednotlivých týmech a další informace o modelovaném zápase. Příkladem může být

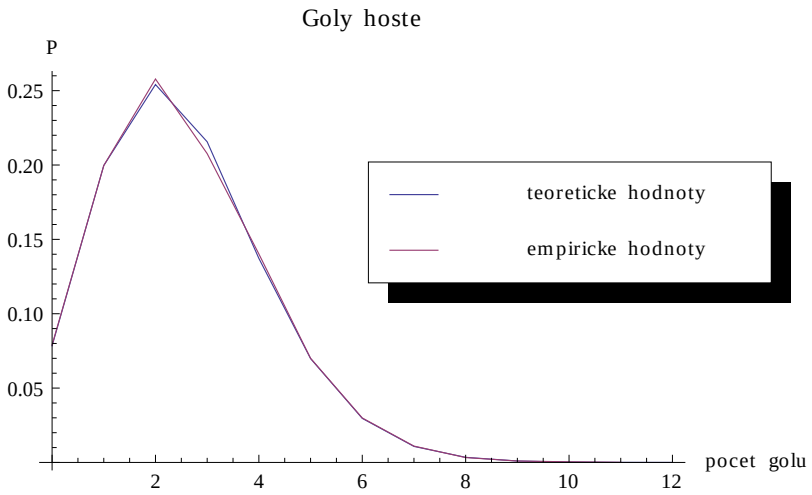
- průměrný počet gólů vstřelených branek domácího (nebo hostujícího) týmu v zápasech hraných během minulých 100 dní
- indikátor, zda modelovaný zápas je zápas nadstavbové části Play Off
- průměr bodů domácího (nebo hostujícího) týmu získaných během posledních 100 dní

Počet gólů v zápase sestává ze součtu náhodných veličin počtu gólů vstřelených domácím týmem a počtu gólů vstřelených hostujícím týmem. Jak ukazuje [11] a i grafické srovnání níže, budeme předpokládat, že tyto náhodné veličiny sledují Poissonovo rozdělení. Výběrovým průměrem odhadneme empirické hodnoty středních hodnot počtu gólů domácích týmů a hostujících týmů - λ_d a λ_h.

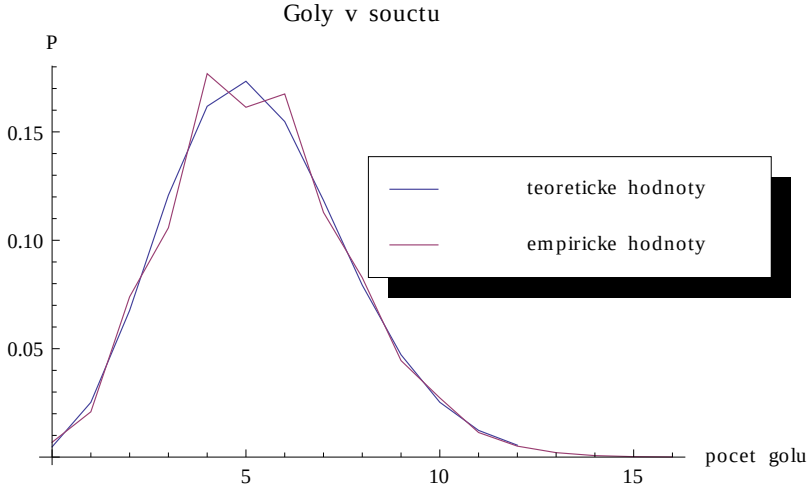


Obrázek 5.1: Srovnání empirických hodnot počtu gólů domácích týmů s Poissonovým rozdělením s $\widehat{\lambda_d} = 2{,}8111$.

Součet dvou nezávislých náhodných veličin s Poissonovými rozděleními má opět Poissonovo rozdělení. Zde je ovšem předpoklad nezávislosti porušen, což ukazuje i srovnání histogramu empirických hodnot proti teoretickým hodnotám Poissonova rozdělení s parametrem λ_s, který jsem odhadli opět jako výběrový průměr součtu gólů obou týmů.
Zuby v grafu 5.3 u empirických hodnot jsou způsobeny především specifickým chováním týmů při konci zápasu, kdy jeden z týmů vyhrává těsným rozdílem (nejčastěji právě o jeden gól) a druhý tým se odhodlá ke hře bez brankáře, čímž



Obrázek 5.2: Srovnání empirických hodnot počtu gólů hostujících týmů s Poissonovým rozdělením s $\widehat{\lambda_h} = 2{,}5468$.



Obrázek 5.3: Srovnání empirických hodnot vstřelených gólů s Poissonovým rozdělením s $\widehat{\lambda_s} = 5{,}3579$.

se zvýší pravděpodobnost jak vstřelené, tak obdržené branky.

I přesto například Ingolfsson v [11] modeluje součet gólů obou týmů jako veličinu s Poissonovým rozdělením. Karlis a Ntzoufras ve [10] navrhují alternativní metodu - modelování počtu branek pomocí dvourozměrného Poissonova rozdělení. Zde předpokládáme nezávislé náhodné veličiny X_1, X_2, X_3 s Poissonovými rozděleními s parametry λ_1, λ_2, λ_3. A předpokládejme, že počet gólů domácího týmů splňuje $X = X_1 + X_3$ a počet gólů hostujícího týmu $Y = X_2 + X_3$, potom pravděpodobnostní rozdělení jednotlivých přesných výsledků je

$$P(X = x, Y = y) = e^{-(\lambda_1+\lambda_2+\lambda_3)} \frac{\lambda_1^x}{x!} \frac{\lambda_2^y}{y!} \sum_{k=0}^{\min(x,y)} \binom{x}{k} \binom{y}{k} k! \left(\frac{\lambda_3}{\lambda_1\lambda_2}\right)^k. \qquad (5.1)$$

Parametr λ_3 má vlastnost míry závislosti mezi X a Y, přesněji platí $\lambda_3 = \text{cov}(X, Y)$. V případě $\lambda_3 = 0$ se redukuje pravděpodobnostní rozdělení na součin rozdělení dvou nezávislých náhodných veličin s Poissonovým rozdělením. Předpokladem tohoto modelu je $\lambda_3 \geq 0$. V našem datovém vzorku ovšem empirický odhad výběrové kovariance vychází $\widehat{\lambda_3} = -0{,}064$, a tak jako výchozí model pro modelování součtu budeme brát jednodušší variantu - součet dvou nezávislých náhodných veličin. Zápornou hodnotu $\widehat{\lambda_3} = -0{,}064$ potvrzuje i hlavní bookmaker společnosti Tipsport Ladislav Pavlis s fundamentálním vysvětlením: *"Zatímco v průběhu zápasu mají oba týmy společnou tendenci buď spíše bránit nebo spíše útočit, tak v případě powerplay je významná snaha jednoho týmu vstřelit branku, zatímco snaha druhého týmu je pouze bránit."*

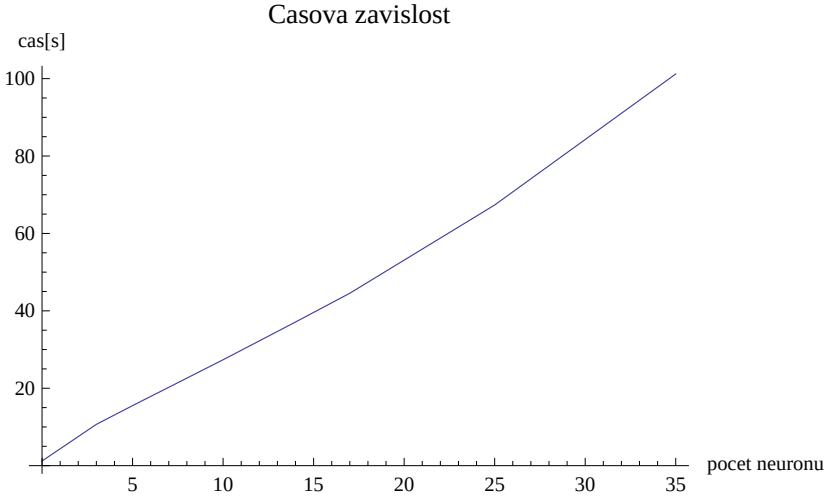
Předpokládejme tedy náhodnou veličinu $X^{(s)}$ vyjadřující celkový počet gólů, které padnou v hokejovém zápase v základní hrací době. Analogicky jako v případě poissonovské regrese modelujeme parametr λ - plně určující celé Poissonovo rozdělení. Odhad parametru λ vyjadřuje odhad střední hodnoty náhodné veličiny $X^{(s)}$.

5.1 Simulace rychlosti modelů

Tato simulace zkoumá rychlost učení neuronové sítě. Nejsou podstatné absolutní časy - tyto časy totiž přímo závisí na použitých komponentech počítače a způsobu implementace, které se liší u každého počítače a použitého software. Implementace v jazyku C++ optimalizovaná pro běh na grafických kartách bude mít pravděpodobně mnohem lepší časové výsledky než běh na notebooku s dvoujádrovým procesorem pouštěný v systémech vyšší úrovně jako je Mathematica. Smyslem této simulace je ukázat časovou náročnost výpočtu v závislosti na hodnotě počtu neuronů ve skryté vrstvě neuronové sítě. Pro každou takovou hodnotu jsme výpočet opakovali 100x. Hodnoty počtu neuronů ve skryté vrstvě jsme postupně brali z množiny

$$U = \{0, 3, 5, 7, 10, 14, 17, 25, 35\}. \tag{5.2}$$

Graf 5.4 zobrazuje závislost mediánu času učení neuronové sítě na počtu neuronů ve skryté vrstvě.



Obrázek 5.4: Čas učení neuronové sítě v závislosti na počtu neuronů ve skryté vrstvě.

Vyšší časové výpočetní nároky, proti například lineárním modelům, snižují možnost provádět větší řadu simulací.

5.2 Simulace dvoustupňového odhadu

V tomto příkladu ověříme oprávněnost navrženého dvoustupňového odhadu parametrů 4.1.1. Tento a dva následující příklady založíme na generování 3 000 simulací. Před začátkem simulací rozdělíme datový vzorek na vývojový a testovací soubor (náhodně bylo vybráno 35% dat). Pro každou takovou simulaci provedeme následující sekvenci procedur

- náhodné rozdělení vývojového vzorku na učící a validační soubory
- naučení neuronové sítě výchozím učícím algoritmem a spočtení střední čtvercové chyby na testovacím vzorku
- spočtení a uložení VIF signálů ze skryté do výstupní vrstvy
- úprava vah pomocí TwoStepNNL s použitím vývojového vzorku a spočtení střední čtvercové chyby na testovacím vzorku
- test na prořezávání sítě na třech $(0{,}05; 0{,}15; 0{,}3)$ zvolených hladinách významnosti
- případně provedené prořezání sítě a spočtení střední čtvercové chyby na testovacím vzorku

Veškeré závěry provádíme na testovacím vzorku, který nebyl zapojen do vývoje modelu v žádné fázi simulace. V tomto příkladu použijeme ve skryté vrstvě aktivační funkci

$$\varphi_k = \arctan(x). \tag{5.3}$$

Na výstupním neuronu použijeme aktivační funkci

$$\xi_v = \text{Sigmoid(x)}, \tag{5.4}$$

tzn. target transformujeme na interval [0;1]. Počet neuronů ve skryté vrstvě je volen náhodně z množiny

$$U = \{3, 5, 7, 9\}. \tag{5.5}$$

Počet vstupujících regresorů je generován náhodně z množiny

$$V = \{5, 6, 7, 8, 9, 10\}. \tag{5.6}$$

Poté je příslušný počet regresorů náhodně vybrán.
Tabulka 5.1 demonstruje výsledky příkladu. Tento příklad ukazuje oprávněnost použití metody TwoStepNNL proti výchozímu algoritmu Levenberg - Marquardt. Jak ukazuje tabulka 5.1, navržený algoritmus ve přibližně 67 procentech případů dosáhl lepších výsledků.

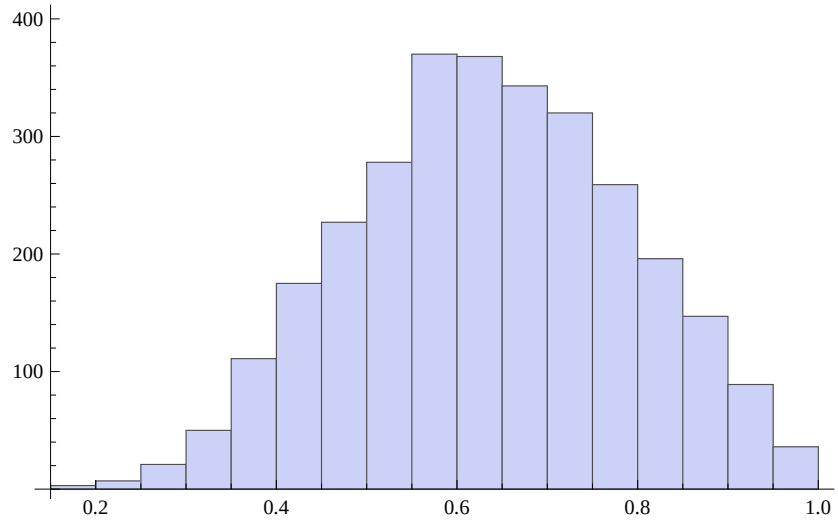
metoda	počet lepších příkladů	%
výchozí učící algoritmus - LM	973	32,43 %
TwoStepNNL	2 027	67,57 %
celkem	3 000	100,0 %

Tabulka 5.1: Srovnání výchozího učícího algoritmu Levenberg - Marquardt s navrženým algoritmem TwoStepNNL.

5.3 Simulace multikolinearity signálů na výstupním neuronu

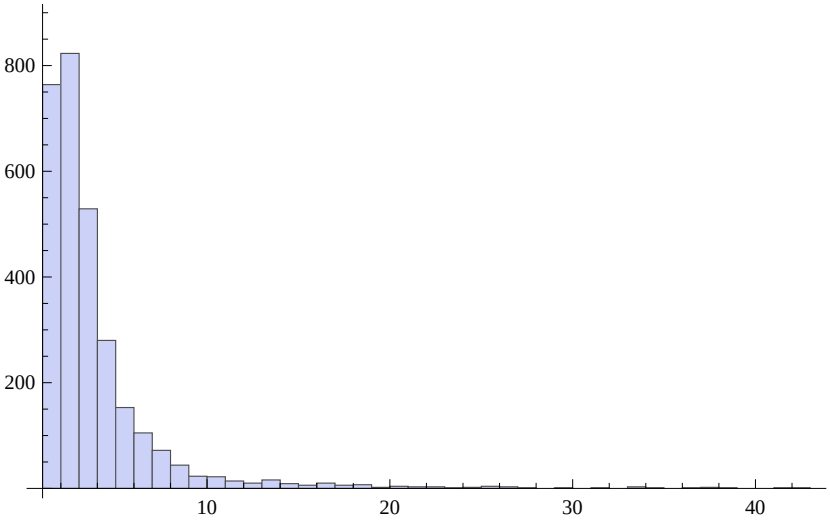
Tento příklad zkoumá, nakolik jsou signály předávané ze skryté vrstvy do vrstvy výstupní korelované. Učící algoritmus z povahy minimalizované účelové funkce nijak nezohledňuje problém multikolinearity. To může mít za následek to, že signály šířící se dál ze skryté vrstvy do vrstvy výstupní mohou obsahovat velmi podobné informace (v dopředné topologii do všech neuronů ve skryté vrstvě přichází ze vstupní vrstvy stejná data). Proto především při snaze interpretovat parametry mezi skrytou a výstupní vrstvou je nutné studovat problém multikolinearity šířících se signálů.

V předchozím příkladu (3 000 simulací) jsme v pro každou simulaci zkoumali maximální absolutní hodnotu nediagonálních prvků korelační matice signálů šířících se ze skryté vrstvy do vrstvy výstupní. Obrázek 5.5 ukazuje histogram těchto hodnot a potvrzuje, že v některých simulacích může nastat problém multikolinearity. V přibližně v 15-ti % simulací přesáhla zkoumaná veličina hodnotu 0,8 a ve 35-ti % simulací hodnotu 0,7.



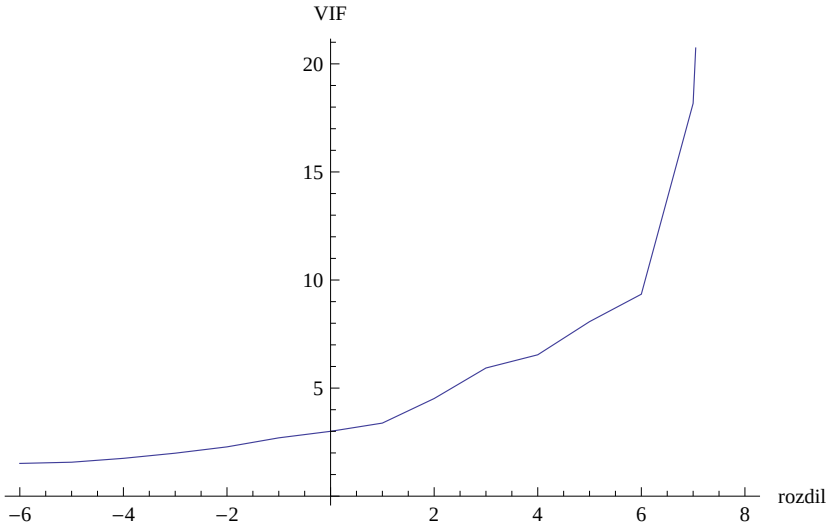
Obrázek 5.5: Histogram maximálních absolutních hodnot nediagonálních prvků korelační matice šířících se signálů.

Pro každou simulaci jsme rovněž zkoumali maximální hodnotu faktoru VIF jednotlivých regresorů pro každou simulaci. Obrázek 5.6 ukazuje histogram těchto maxim a i tato míra potvrzuje riziko na výskyt multikolinearity.



Obrázek 5.6: Histogram maxim faktorů VIF ze všech simulací.

Překročení referenční hodnoty 10 nastalo ale pouze v cca 5-ti % případů, překročení hodnoty 5 pak v přibližně 18-ti % případů. Obě metody pro takto navrženou úlohu detekovaly multikolinearitu šířících se signálů přibližně v 5-ti až 10-ti procentech případů. Lepší vysvětlení tohoto problému ukazuje obrázek 5.7, který zobrazuje závislost průměrných hodnot faktorů VIF na hodnotě rozdílu počtu neuronů ve skryté vrstvě a počtu regresorů vstupujích do vstupní vrstvy.



Obrázek 5.7: Závislost průměrné hodnoty faktorů VIF na hodnotě rozdílu počtu neuronů ve skryté vrstvě a počtu regresorů vstupujících do vstupní vrstvy.

Tuto závislost interpretujeme jako snahu učícího algoritmu konstruovat signifikantní signály mezi skrytou a výstupní vrstvou. Čím je větší rozdíl mezi počtem neuronů ve skryté vrstvě proti počtu regresorů vstupujících do vstupní vrstvy, tím je pro učící algoritmus obtížnější konstruovat navzájem nekorelované šířící se signály mezi skrytou a výstupní vrstvou. Takto zkonstruovanou závislost navíc můžeme použít jako heuristický odhad pro volbu počtu neuronů ve skryté vrstvě vzhledem k počtu vstupních regresorů.

5.4 Simulace prořezávání neuronové sítě

V tomto příkladu prozkoumáme možnost prořezání neuronové sítě (odstranění nadbytečných spojů v topologii). Opět vyjdeme z příkladu 5.2, kde pro každou z 3 000 simulací nad rámec aplikace algoritmu TwoStepNNL zkoumáme statistickou významnost signálů mezi skrytou a výstupní vrstvou dle příslušných p-hodnot (p-value). P-hodnoty odhadneme při stavbě modelu v rámci TwoStepNNL algoritmu. Prořezání provedeme, pokud maximum p-hodnot je vyšší než zvolená kritická hranice. Prořezáním rozumíme odstranění váhy příslušného signálu (= zafixování na hodnotu 0) - viz obrázek 4.2 a opětovné použití algoritmu TwoStepNNL. Za kritickou hranici jsme zvolili postupně hodnoty $\{0{,}05; 0{,}15; 0{,}3\}$. Pokud došlo k překročení kritické hranice, došlo k prořezání - odstranění právě jednoho nejhoršího signálu (dle p-hodnoty). Následně byla spočtena střední čtvercová chyba pro výchozí učící algoritmus, algoritmus TwoStepNNL a algoritmus TwoStepNNL doplněný o prořezání. Následující tabulky ukazují, že nezávisle na hladině prořezání, nedosahuje algoritmus doplnění o prořezání výrazně lepších výsledků než samotný algoritmus TwoStepNNL. Významně lepší hodnoty algoritmu TwoStepNNL doplněného o prořezání proti výchozímu učícímu algoritmu Levenberg-Marquardt proto přikládáme samotnému algoritmu TwoStepNNL nikoliv prořezávání.

Počet všech simulací	3 000
Počet simulací s prořezáváním na hladině 5%	2 857
Prořezaná síť převyšuje výchozí algoritmus	1 910 (66,85 %)
Prořezaná síť převyšuje algoritmus TwoStepNNL	1 477 (51,70 %)

Tabulka 5.2: Srovnání metod s rozšířením o prořezávání na hladině 0,05.

Počet všech simulací	3 000
Počet simulací s prořezáváním na hladině 15%	2 750
Prořezaná síť převyšuje výchozí algoritmus	1 838 (66,84 %)
Prořezaná síť převyšuje algoritmus TwoStepNNL	1 418 (51,56 %)

Tabulka 5.3: Srovnání metod s rozšířením o prořezávání na hladině 0,15.

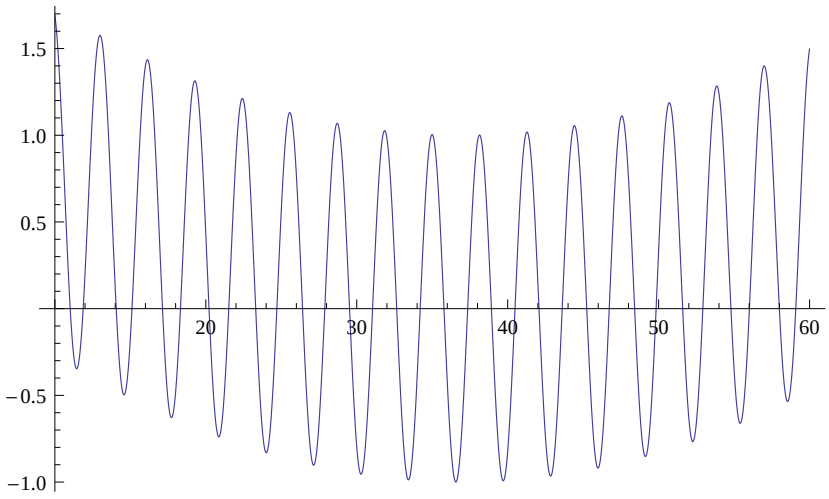
Počet všech simulací	3 000
Počet simulací s prořezáváním na hladině 30%	2 577
Prořezaná síť převyšuje výchozí algoritmus	1 712 (66,43 %)
Prořezaná síť převyšuje algoritmus TwoStepNNL	1 322 (51,30 %)

Tabulka 5.4: Srovnání metod s rozšířením o prořezávání na hladině 0,30.

5.5 Srovnání jednoduchého binárního a Grayova binárního kódování

Pro srovnání obou způsobů kódování použijeme úlohu hledání minima funkce s velkým počtem lokálních extrému. Za takovou funkci si volíme funkci

$$f(x) = \sin(2x - 37) + 0{,}001(x - 37)^2, \tag{5.7}$$



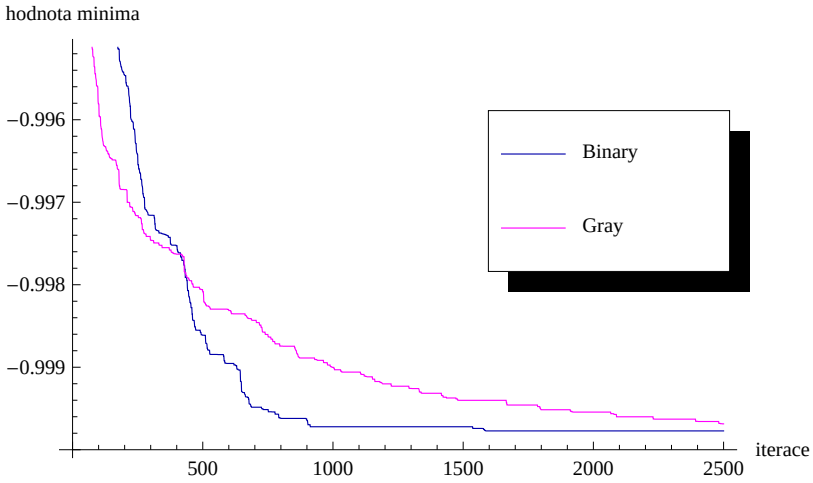
Obrázek 5.8: Funkce $f(x) = \sin(2x - 37) + 0{,}001(x - 37)^2$

Dvě často používané funkce *Mathematic-y* pro řešení minimalizačních úloh jsou funkce FindMinimum a NMinimize, které navíc umožňují volbu různých optimalizačních algoritmů. Pro srovnání budeme hledat minimum zvolené funkce i pomocí všech těchto algoritmů. Tabulka 5.5 ukazuje výsledky jednotlivých funkcí s příslušnými algoritmy a výsledky genetických algoritmů s oběma typy kódování. Jak ukazuje tabulka 5.5, globální minimum v okolí bodu 36,5 nalezly pouze genetické algoritmy v obou variantách kódování a dvě varianty funkce *NMinimize*. Z toho jedna metoda *DifferentialEvolution* hledá argument minima velmi podobně jako genetické algoritmy [22].

Pro srovnání rychlosti konvergence obou variant kódování zopakujeme tento pokus 500x vždy pro novou počáteční populaci jedinců. Přes všechny tyto pokusy spočítáme pro každý *čas* iterace průměrnou hodnotu aktuálně nejlepších jedinců v populaci. Graf 5.9 nám poté porovnává obě varianty kódování.
Grayovo kódování v prvotních iteracích dosahuje lepšího hodnot nalezeného minima - ze své podstaty snadněji konverguje v rámci aktuálně nalezeného lokálního

algoritmus	specifikace	hodnota x	dosažené minimum
FindMinimum	Automatic	−1,11589	0,453548
FindMinimum	ConjugateGradient	8,30417	−0,176138
FindMinimum	PrincipalAxis	2,02413	0,223923
FindMinimum	Newton	2,02413	0,223923
FindMinimum	QuasiNewton	5,16415	0,014028
FindMinimum	InteriorPoint	−1,11589	0,453548
NMinimize	Automatic	36,5644	−0,999810
NMinimize	NelderMead	33,4244	−0,987208
NMinimize	DifferentialEvolution	36,5644	−0,999810
NMinimize	SimulatedAnnealing	20,8643	−0,739508
NMinimize	RandomSearch	27,1443	−0,902817
GA	Binary	36,5600	−0,999772
GA	Gray	36,5600	−0,999772

Tabulka 5.5: Srovnání metod a jejich specifikací při hledání minima u zvolené funkce.



Obrázek 5.9: Srovnání obou variant kódování.

minima. Zatímco variabilnější jednoduché binární kódování (mutace jednoho bitu může výrazně změnit hodnotu řešení - na rozdíl of Grayova kódování) rychleji konverguje ke řešení globálnímu.

V případě úlohy s očekávaným velkým počtem lokálních extrémů se proto lépe jeví použití jednoduchého binárního kódování. Naopak u úloh s menším počtem těchto lokálních extrémů může Grayovo kódování konvergovat rychleji.

5.6 Použití genetických algoritmů pro optimalizaci neuronové sítě

Pro příklad použití kombinace genetického algoritmu a neuronové sítě vyjdeme z kapitoly 3.3.4. V tomto příkladu použijeme GA pro šlechtění topologie a učící proces samotný necháme na výchozím učícím algoritmu Levenberg-Marquardt. Vzhledem k časové náročnosti výpočtu necháme GA hledat

- počet vstupních regresorů
- volba vstupních regresorů
- počet neuronů ve skryté vrstvě

Ostatní nastavení v rámci topologie sítě necháme pro srovnaní stejné jako v příkladu 5.2. Účelovou funkci

$$\min_{\boldsymbol{\pi}} v(\boldsymbol{\beta}, \boldsymbol{\pi}) = \min_{\boldsymbol{\pi}} \frac{1}{J} \sum_{j=1}^{J} \frac{1}{n} \sum_{i=1}^{n} [y_i - f^*(\boldsymbol{\beta_j}, \mathbf{x_i}, \boldsymbol{\pi})]^2 \tag{5.8}$$

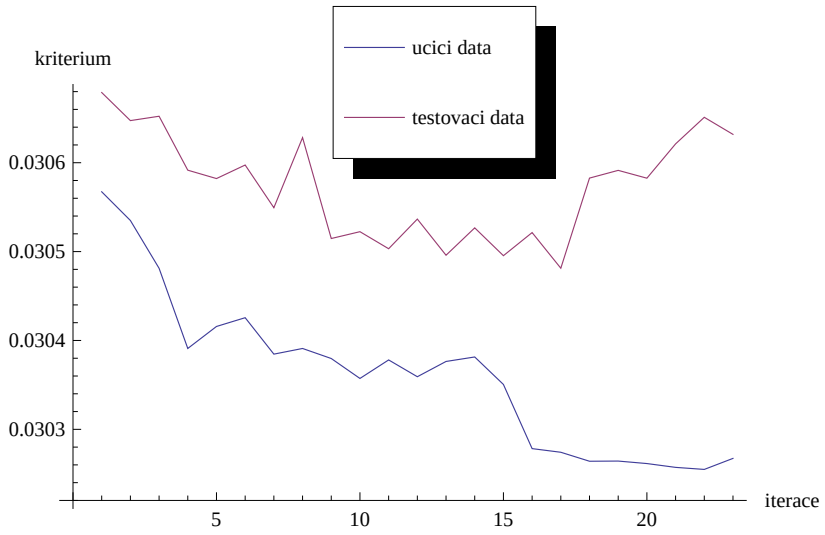
opět pro velkou časovou náročnost výpočtu použijeme s volbou $J = 25$ a počtem jedinců v populaci $K = 50$. Jedince populace reprezentujeme jako řetězec nul a jedniček délky $R = 58$, kde prvních 56 pozic vyjadřuje zařazení regresorů do modelu (1 = ano, 0 = ne) - používáme všech 56 regresorů. Zbylé dvě pozice reprezentují počet neuronů ve skryté vrstvě se stavy (použití Grayova kódování)

- $\{0, 0\} \rightarrow 3$
- $\{0, 1\} \rightarrow 5$
- $\{1, 1\} \rightarrow 7$
- $\{1, 0\} \rightarrow 9$.

I zde se jeví jako velkým problém riziko přeučení. Po pár iteracích se začíná GA učící data interpolovat - stále klesá hodnota minimalizovaného kritéria na učících datech, zatímco na testovacích datech chyba roste viz 5.10. Algoritmus můžeme tedy zastavit a za minimum vzít populaci (a jejího nejlepšího jedince) dle testovacího vzorku - v tomto případu populace ze 17-té iterace. GA (s přihlédnutím k testovacímu vzorku) vybral 10 regresorů

$$J_{GA} = (6, 8, 9, 10, 11, 25, 32, 38, 42, 49). \tag{5.9}$$

Jako vhodný počet neuronů byl zvolen počet 9.



Obrázek 5.10: Průběh hodnot minimalizovaného kritéria na učících a testovacích datech při použití GA pro hledání vhodných regresorů a počtu neuronů ve skryté vrstvě.

Tento příklad demonstruje použití GA při hledání vhodných regresorů a počtu neuronů ve skryté vrstvě. Jako důležité se ukazuje studium průběhu hodnot minimalizovaného kritéria na testovacích datech.

5.7 Aplikace modelů pro modelování počtu gólů v zápase

V tomto příkladu aplikujeme představené techniky na predikci pravděpodobnosti, že v hokejových zápasech padne 6 a více branek (tzv. over 5,5). Opět použijeme všech 56 regresorů - tabulka 5.11. Závislá proměnná Y_i vyjadřuje dle použitého modelu buďto náhodnou veličinu celkového počtu gólů vstřelených v zápase i, kde $i = 1, \ldots, T$ v základní hrací době (to znamená bez případného prodloužení) anebo přímo pravděpodobnost, že v zápase i padne 6 a více branek. Budeme předpokládat, že celkový počet branek v zápase má Poissonovo rozdělení s parametrem λ_i, který vyjadřuje rovněž střední hodnotu a rozptyl náhodné veličiny celkového počtu branek v zápase. Modelujeme tedy podmíněnou střední hodnotu náhodné veličiny Y_i za podmínky známých hodnot regresorů $\mathbf{x}_i$

$$E(Y_i|\mathbf{x_i}, \boldsymbol{\beta}) = f(\mathbf{x_i}, \boldsymbol{\beta}), \tag{5.10}$$

kde $f(\mathbf{x_i}, \boldsymbol{\beta})$ představuje zvolený model s příslušnými parametry $\boldsymbol{\beta}$. Pro otestování síly modelu použijeme porovnání proti kurzům světových sázkových kanceláří na již odehraných zápasech. Pro srovnání budou použity kurzy sázkových kanceláří, které jsou vypsány pro náhodnou událost, zda v daném zápase padne více či méně než 5,5 branek. Použité kurzy reprezentují průměrné uzavírací hodnoty největších světových sázkových kanceláří a představují tedy odhad určený *trhem*. Příklad hodnot ukazuje obrázek 5.16 (v části příloh).

Necelá hranice 5,5 představuje používanou konvenci jasně určující všechny možné stavy výsledku. Historie kurzů sázkových kanceláří pro jednotlivé zápasy jsou rovněž na webové stránce viz tabulka 5.10. Jiné dostupné zdroje kurzů, výsledků či dalších informací jsou rovněž v tabulce 5.10. Pro srovnání provedeme odhad počtu gólů pomocí poissonovské a logistické regrese s použitím metody Stepwise, jednoduchou dopřednou neuronovou sítí a na závěr použijeme metodu lesů neuronových sítí. Ve všech případech použijeme rozdělení datového souboru na vývojovou a testovací část. Testovací část není v žádné fázi použita pro vývoj modelu a představuje tak nezávislý datový soubor a je pro všechny modely identická. Jako metodu srovnání pro historické kurzy z testovacího vzorku použijeme fiktivní sázení částky 1 jednotka. Sázku pro daný zápas provedeme v případě, kdy převrácená hodnota odhadu pravděpodobnosti události, že padne méně 5,5 gólů je nižší než kurz na tuto událost. Stejně postupujeme i pro doplňkovou událost výskytu více něž 5,5 gólů v zápase. V tomto případě použijeme doplňkovou pravděpodobnost, že padne více než 5,5 gólů a kurz na tuto událost. Zaveďme následující značení

- $p_{i,m}$ pravděpodobnost, že v zápase i padne méně než 5,5 gólů
- $p_{i,v} = 1 - p_{i,m}$ pravděpodobnost, že v zápase i padne více než 5,5 gólů
- $k_{i,m}$ medián kurzů sázkových kanceláří pro událost, že v zápase i padne méně než 5,5 gólů
- $k_{i,v}$ medián kurzů sázkových kanceláří pro událost, že v zápase i padne více než 5,5 gólů

- $v_{i,m}$ vklad na událost, že v zápase i padne méně než 5,5 branek
- $v_{i,v}$ vklad na událost, že v zápase i padne více než 5,5 branek
- r_i výsledek zápasu i ve formě celkového počtu gólů obou týmů v základní hrací době
- $z_{i,m}$ zisk sázky v případě vkladu na událost méně než 5,5 gólů u zápasu i při kurzu $k_{i,m}$ a nastalém výsledku r_i
- $z_{i,v}$ zisk sázky v případě vkladu na událost více než 5,5 gólů u zápasu i při kurzu $k_{i,v}$ a nastalém výsledku r_i
- $z_i = z_{i,m} + z_{i,v}$ celkový zisk u zápasu i

Kurzem zde rozumíme hodnotu, kterou se násobí vklad v v případě, že sázka je správná - například pří správném uhádnutí, že v zápase týmu A proti týmu B padne méně než 5,5 branek a kurzu na tuto událost ve výši $k_v = 1{,}80$, je zisk z_m při vkladu $v_m = 1$ jednotka roven $z_m = (k_v - 1)v_m = (1{,}80 - 1) \cdot 1 = 0{,}8$ jednotky. V případě neuhádnutí je zisk $z_m = -1$ jednotka. Celkový zisk při sázení v případě této strategie můžeme tedy psát jako

$$z_i = z_{i,m} + z_{i,v}, \tag{5.11}$$

kde

$$z_{i,m} = \mathrm{I}_{(p_{i,m} \cdot k_{i,m} > 1{,}0)} \cdot (-v_{i,m} + \mathrm{I}_{(r_i < 5{,}5)} \cdot v_{i,m} \cdot k_{i,m}) \tag{5.12}$$

a

$$z_{i,v} = \mathrm{I}_{(p_{i,v} \cdot k_{i,v} > 1{,}0)} \cdot (-v_{i,v} + \mathrm{I}_{(r_i > 5{,}5)} \cdot v_{i,v} \cdot k_{i,v}). \tag{5.13}$$

Výraz $\mathrm{I}_{(p_{i,m} \cdot k_{i,m} > 1{,}0)}$ vyjadřuje indikátor události, že kurz sázkové kanceláře je vzhledem k odhadnuté pravděpodobnosti je příliš vysoký. Výraz $\mathrm{I}_{(r_i < 5{,}5)}$ je indikátor události, že padne v zápase i méně než 5,5 gólů. Úlohu tedy můžeme formulovat jako hledání nejpřesnějších odhadů jednotlivých pravděpodobností $p_{i,m}$. Nejjednodušší přístup je odhad $p_{i,m}$ konstantní hodnotou empirického výběrového průměru, tedy

$$\widehat{p_{i,m}} = \widehat{p_{k,m}} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{I}_{(r_i < 5{,}5)}, \quad i = 1, \ldots, n. \tag{5.14}$$

V našem datovém vzorku takto odhadneme $\widehat{p_{k,m}} = 0{,}547124$. Nyní provedeme fiktivní sázení podle nadefinované strategie 5.12 a 5.13 a napočítáme celkový zisk dle 5.11. Výsledky této strategie najdeme v tabulce 5.9.

Poissonovská regrese

Další použitou metodou pro srovnání je model poissonovské regrese. Jak ukazuje obrázek 5.3, náhodná veličina Y_i má přibližně Poissonovo rozdělení. Pravděpodobnost $p_{i,m}$ tedy budeme modelovat jako

$$p_{i,m} = \sum_{k=0}^{5} \frac{\lambda_i^k e^{-\lambda_i}}{k!}, \tag{5.15}$$

kde λ_i modelujeme jako

$$\mathrm{E}(Y_i|\mathbf{x_i}) = e^{\mathbf{x_i}\boldsymbol{\beta}}. \tag{5.16}$$

Pravděpodobnosti $\widehat{p_{i,m}}$ pak odhadneme

$$\widehat{p_{i,m}} = \sum_{k=0}^{5} \frac{\widehat{\lambda_i}^k e^{-\widehat{\lambda_i}}}{k!} = \sum_{k=0}^{5} \frac{e^{(\mathbf{x_i}\widehat{\boldsymbol{\beta}})^k} e^{-e^{\mathbf{x_i}\widehat{\boldsymbol{\beta}}}}}{k!}. \tag{5.17}$$

Metoda Stepwise pro tento model vybrala regresory s indexy (řazeno dle pořadí vybrání regresorů)

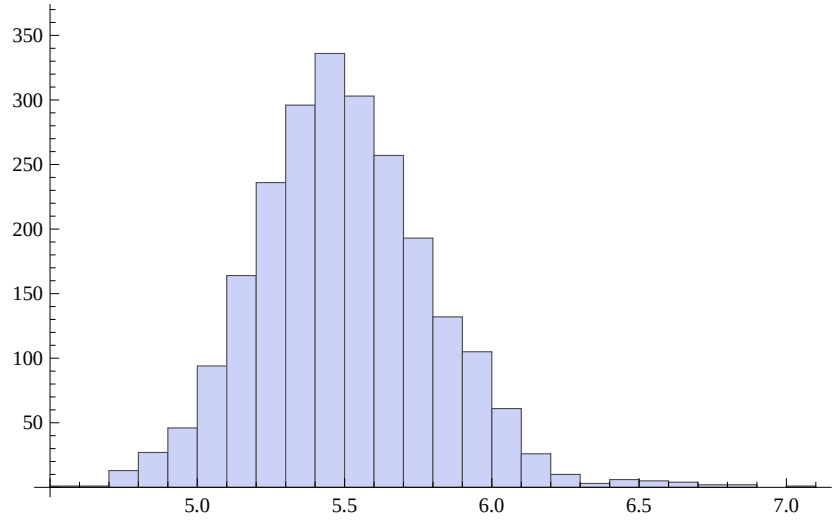
$$J_p = (10, 8, 7, 9, 49, 11, 35, 12, 21, 41). \tag{5.18}$$

Odhad parametrů a p-hodnoty regresorů demonstruje tabulka 5.6

regresor	odhad parametru	p-hodnota
konst.	0,938888	$< 0{,}001$
regresor_{10}	0,063443	$< 0{,}001$
regresor_{8}	0,062045	$< 0{,}001$
regresor_{7}	0,056876	$< 0{,}001$
regresor_{9}	0,041944	$< 0{,}001$
regresor_{49}	$-0{,}063717$	$< 0{,}001$
regresor_{11}	0,020028	0,005
regresor_{35}	$-0{,}007876$	0,046
regresor_{12}	0,015175	0,033
regresor_{21}	0,235971	0,007
regresor_{41}	$-0{,}241675$	0,045

Tabulka 5.6: Tabulka odhadů parametrů pro regresory vybrané technikou Stepwise regression na hladině významnosti 5%.

Histogram odhadů hodnot $\widehat{\lambda_{i,m}}$ demonstruje obrázek 5.11



Obrázek 5.11: Histogram odhadnutých $\widehat{\lambda_{i,m}}$ metodou poissonovské regrese.

Obrázek 5.11 ukazuje variabilitu predikcí, především rozsah hodnot predikcí. Minimální a maximální predikovaná pravděpodobnost u tohoto modelu vychází $\{0{,}307; 0{,}705\}$. Hodnoty srovnání simulovaného sázení proti kurzům sázkových kanceláří najdeme v tabulce 5.9.

Logistická regrese

Další použitou metodou pro srovnání je model logistické regrese. Pravděpodobnost $p_{i,m}$ tentokrát budeme modelovat přímo

$$p_{i,m} = \mathrm{E}(Z_i|\mathbf{x_i}) = \frac{e^{\mathbf{x_i}\boldsymbol{\beta}}}{1 + e^{\mathbf{x_i}\boldsymbol{\beta}}}, \tag{5.19}$$

kde Z_i představuje binární náhodnou veličinu indikující, zda v zápase i padlo méně než 5,5 branek $Z_i = 1$, nebo zda padlo více než 5,5 branek $Z_i = 0$. Pravděpodobnosti $\widehat{p_{i,m}}$ pak odhadneme

$$\widehat{p_{i,m}} = \frac{e^{\mathbf{x_i}\widehat{\boldsymbol{\beta}}}}{1 + e^{\mathbf{x_i}\widehat{\boldsymbol{\beta}}}} \tag{5.20}$$

Metoda Stepwise pro tento model vybrala regresory s indexy (řazeno dle pořadí vybrání regresorů)

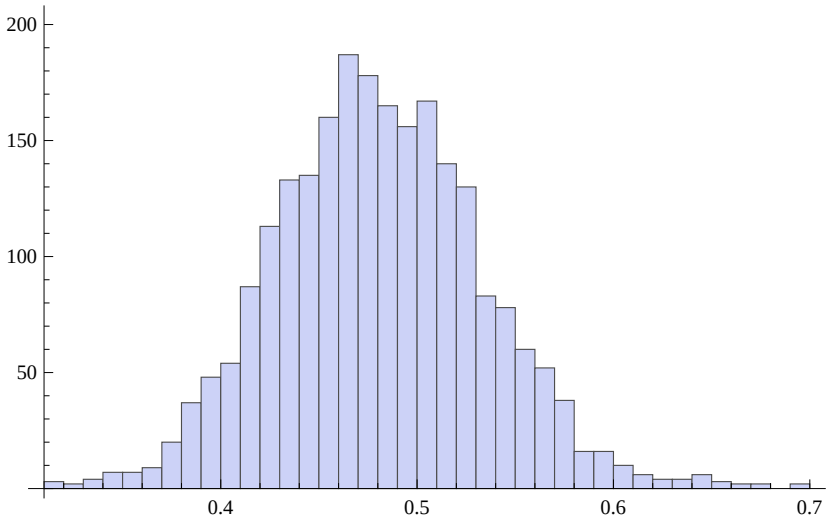
$$J_L = (10, 8, 9, 7, 49, 30, 1, 54) \tag{5.21}$$

Odhad parametrů a p-hodnoty regresorů demonstruje tabulka 5.7

Histogram odhadů hodnot $\widehat{p_{i,m}}$ demonstruje obrázek 5.12.
Hodnoty minimální a maximální predikované pravděpodobnosti jsou $\{0{,}313; 0{,}696\}$.
Hodnoty srovnání proti kurzům sázkových kanceláří najdeme v tabulce 5.9.

regresor	Odhad parametru	p-hodnota
konst.	$-2{,}62448$	$< 0{,}001$
regresor_{10}	$0{,}26809$	$< 0{,}001$
regresor_{8}	$0{,}24505$	$< 0{,}001$
regresor_{9}	$0{,}16547$	$< 0{,}001$
regresor_{7}	$0{,}19502$	$< 0{,}001$
regresor_{49}	$-0{,}23213$	$0{,}006$
regresor_{30}	$0{,}04589$	$0{,}001$
regresor_{1}	$-0{,}02283$	$0{,}013$
regresor_{54}	$-0{,}00165$	$0{,}021$

Tabulka 5.7: Tabulka odhadů parametrů pro regresory vybrané technikou Stepwise regression na hladině významnosti 5%.



Obrázek 5.12: Histogram odhadnutých $\widehat{p_{i,m}}$ metodou logistické regrese.

Neuronová síť s jednou skrytou vrstvou

Další použitou metodou pro srovnání je model dopředné neuronové sítě s jednou skrytou vrstvou s učícím algoritmem Levenberg-Marquardt. Použijeme stejný přístup jako u modelu logistické regrese. Modelovat budeme binární náhodnou veličinu Z_i. Aktivační funkci výstupního neuronu zvolíme logistický sigmoid, ve skryté vrstvě jako aktivační neuron zvolíme funkci $\arctan(x)$. Pro volbu regresní matice použijeme následující jednoduchou heuristiku: vezmeme prvních 8 (= stejný počet jako logistická regrese) nejvíce absolutně korelovaných sloupců z regresní matice (pouze s ohledem na učící vzorek). Takto jsme vybrali regresory s indexy

$$J_N = (10, 8, 26, 28, 9, 30, 7, 6). \tag{5.22}$$

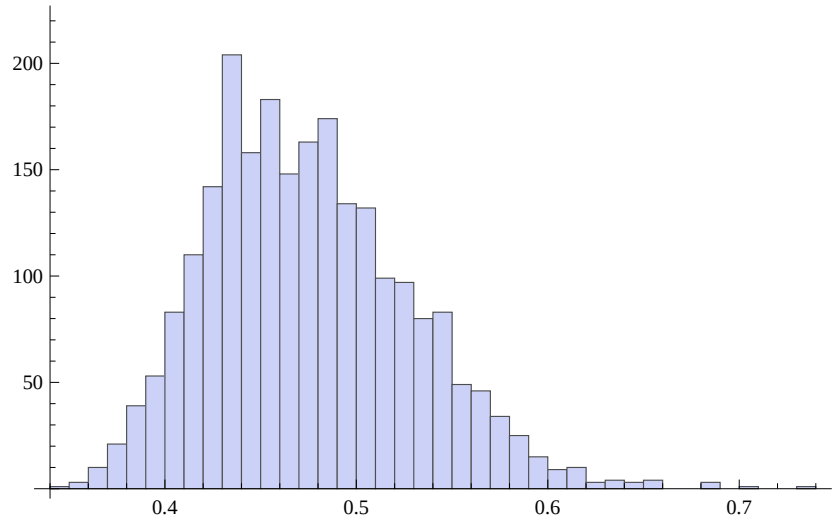
Počet neuronů ve skryté vrstvě volíme o 3 větší než je počet regresorů, tedy $Q = 11$. Při učení neuronové sítě bylo provedeno zastavení učícího procesu pomocí validačního vzorku. Pro modelování $p_{i,m}$ vyjdeme ze vzorce 2.19 a můžeme tedy psát

$$p_{i,m} = \mathrm{E}(Z_i|\mathbf{x_i}) = \xi_v \left[\theta_v + \sum_{k=1}^{Q} \beta_k \cdot \varphi_k(\theta_k + \sum_{j=1}^{P} \alpha_{j,k} \cdot x_{i,j}) \right]. \tag{5.23}$$

Pravděpodobnosti $\widehat{p_{i,m}}$ pak odhadneme

$$\widehat{p_{i,m}} = \xi_v \left[\widehat{\theta_v} + \sum_{k=1}^{Q} \widehat{\beta_k} \cdot \varphi_k(\widehat{\theta_k} + \sum_{j=1}^{P} \widehat{\alpha_{j,k}} \cdot x_{i,j}) \right]. \tag{5.24}$$

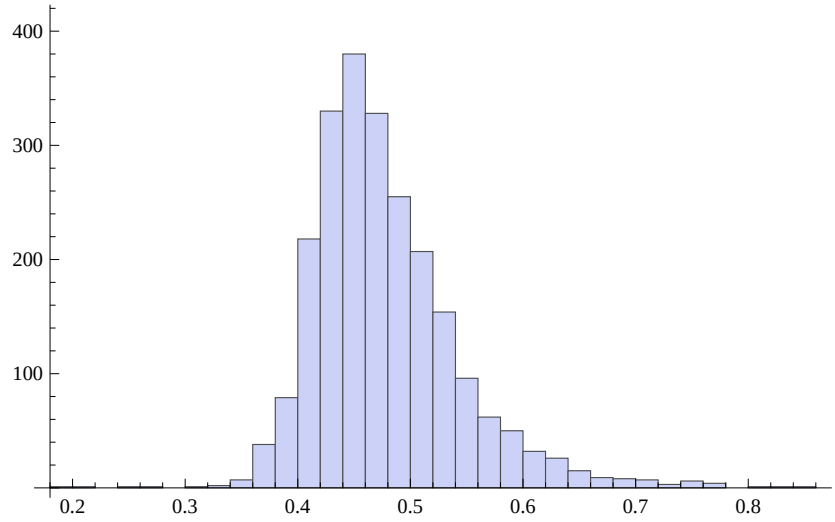
Odhady parametrů $\widehat{\theta_v}$ A $\widehat{\beta_k}$ u šířících se signálů ze skryté vrstvy do výstupní popisuje tabulka 5.7. Histogram odhadů hodnot $\widehat{p_{i,m}}$ demonstruje obrázek 5.13.



Obrázek 5.13: Histogram odhadnutých $\widehat{p_{i,m}}$ metodou dopředné neuronové sítě.

Hodnoty minimální a maximální predikované pravděpodobnosti jsou $\{0{,}358, 0{,}738\}$. Hodnoty srovnání proti kurzům sázkových kanceláří najdeme v tabulce 5.9.

Další zkoušenou variantou je navrhnutý dvoustupňový odhad TwoStepNN a jeho varianta s prořezáním nadbytečných signálů. Obě tyto varianty vychází z předchozí naučené neuronové sítě. Protože Z_i je binární náhodná veličina použijeme v rámci metody TwoStepNNL model logistické regrese. Provedené prořezání spočívá ve vynulování právě jednoho parametru, a to u nejvíce zbytečného signálu. Struktura modelu pro odhad $p_{i,m}$ je tedy v obou případech stejná jako u naučené neuronové sítě - změna spočívá pouze v hodnotách odhadnutých parametrů.
Histogram odhadů hodnot $\widehat{p_{i,m}}$ metodou dvoustupňového odhadu TwoStepNN s variantou obsahující prořezání demonstruje obrázek 5.14
V tabulce 5.7 první sloupec reprezentuje jednotlivé odhady parametrů $\widehat{\beta_k}$ u příslušných šířících se signálů, v druhém sloupci najdeme hodnoty odhadů parametrů pomocí výchozí metody Levenberg-Marquardt (LB). Ve třetím sloupci najdeme hodnoty odhadů stejných parametrů pomocí navrženého algoritmu TwoStepNNL, ve čtvrtém sloupci jsou p-hodnoty významnosti parametrů při použití algoritmu TwoStepNNL. V pátém sloupci máme odhady opět stejných parametrů při použití varianty s prořezáním jednoho nejvíce nadbytečného signálu a ve šestém sloupci k nim příslušné p-hodnoty.



Obrázek 5.14: Histogram odhadnutých $\widehat{p_{i,m}}$ metodou dopředné neuronové sítě s odhadem parametrů pomocí TwoStepNN s variantou obsahující prořezání.

parametr	Odhad LB	2step	p-hodn.	2stepP	p-hodn.
$\widehat{\theta_v}$	1,16334	−11,9214	0,404	−10,9992	0,432
$\widehat{\beta_1}$	0,12860	0,94225	0,003	0,93721	0,003
$\widehat{\beta_2}$	−0,07268	−0,06976	0,708	−0,06065	0,741
$\widehat{\beta_3}$	−0,29311	−7,87787	0,001	−7,76675	0,001
$\widehat{\beta_4}$	0,19280	0,64748	0,003	0,64596	0,004
$\widehat{\beta_5}$	0,15704	−0,16326	0,746	0,00000	-
$\widehat{\beta_6}$	−0,05614	−0,79013	0,003	−0,77958	0,004
$\widehat{\beta_7}$	−0,38212	−1,34991	$< 0{,}001$	−1,34480	$< 0{,}001$
$\widehat{\beta_8}$	0,01206	0,20187	0,304	0,19601	0,316
$\widehat{\beta_9}$	−0,13193	−0,35855	0,102	−0,29718	0,008
$\widehat{\beta_{10}}$	0,02879	−18,5821	0,118	−17,9506	0,126
$\widehat{\beta_{11}}$	−0,09795	−3,70152	0,070	−3,73161	0,067
$\widehat{\beta_{12}}$	−1,07177	−1,03767	$< 0{,}001$	−1,03372	$< 0{,}001$
$\widehat{\beta_{13}}$	−0,01619	−0,30673	0,369	−0,29054	0,389

Tabulka 5.8: Odhady parametrů u šířících se signálů jednotlivými metodami.

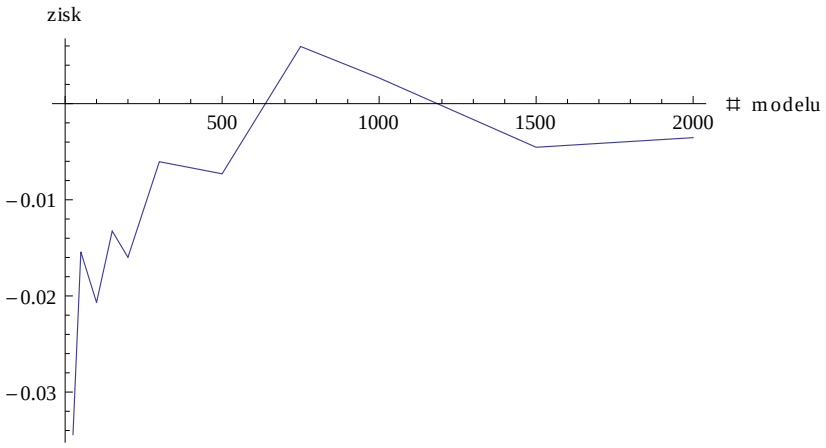
Hodnoty srovnání proti kurzům sázkových kanceláří najdeme opět v tabulce 5.9.

Lesy neuronových sítí

Závěrečnou testovanou metodou je metoda lesů neuronových sítí. Modelovat budeme opět binární náhodnou veličinu Z_i. Aktivační funkci výstupního neuronu všech jednotlivých neuronových sítí zvolíme logistický sigmoid, ve skryté vrstvě jako aktivační neuron volíme vždy funkci $\arctan(x)$. Jednotlivá nastavení pro každou neuronovou síť a volbu vstupních regresorů generujeme náhodně následovně

- hodnotu počet regresorů generujeme náhodně z rozsahu 5 až 15
- hodnotu počtu řádků konstrukční matice pro učící proces generujeme náhodně z intervalu (0,35 ; 0,65)
- hodnotu počtu neuronů ve skryté vrstvě generujeme náhodně z rozsahu 3 až 15

Za výslednou predikci $\widehat{y_i}$ pro jednotlivá pozorování z testovacího vzorku volíme aritmetický průměr jednotlivých dosažených predikcí. Vzhledem k implicitní vlastnosti vypořádat se s problémem přeučení, byl testovací vzorek použit při stavbě jednotlivých neuronových sítí. Takto stavíme 2000 modelů. Graf 5.15 ukazuje dosažený zisk vzhledem k počtu použitých modelů.



Obrázek 5.15: Závislost dosaženého zisku sázecí strategie vzhledem k počtu generovaných modelů.

Jak ukazuje obrázek 5.15, dosažený zisk roste vzhledem k počtu použitých modelů. Nicméně tento růst se zastavuje (pro náš datový soubor) při použití přibližně 800 modelů a poté mírně klesá a zůstává přibližně konstantní. Za ideální počet tedy volíme hodnotu 800 modelů, což interpretujeme jako optimální hodnotu vzhledem k velikosti použitého datového souboru (ve smyslu počtu pozorování i počtu regresorů). Srovnání proti ostatním metodám najdeme v tabulce 5.9.

Výsledky

Dosažené výsledky potvrzují, že nelineární přístup pomocí neuronových sítí dává lepší výsledky než metody lineární. Základní odhad pomocí konstanty potvrzuje, že sázkové kanceláře stanovují kurzy celkem dobře. Použitím lineárních modelů se dokážeme přiblížit až na takovou hranici, kdy sázení by jak pro sázejícího, tak sázkovou kancelář představoval hru s nulovým součtem. Vzhledem k maržím sázkových kanceláří tento výsledek demonstruje, že i lineární metody dosahují přesnějšího odhadu jednotlivých pravděpodobností, než jsou odhady sázkových kanceláří. Koncept neuronových sítí, pak i přes marže sázkových kanceláří, dosahuje kladných hodnot vzhledem k navržené strategii pro fiktivní sázení.

metoda	Počet sázených zápasů	zisk	%
konstantní $p_{k,m}$	1 481	−70,32	−4,748%
Poisson reg.	1 061	−0,54	−0,051%
Logit	1 018	4,61	0,005%
NN	1 231	14,98	1,216%
TwoStepNNL	1 213	43,10	3,553%
TwoStepNNL+prořezání	1 217	46,59	3,828%
Lesy NN	1 215	7,23	0,595%

Tabulka 5.9: Výsledky jednotlivých metod vzhledem k navržené strategii fiktivního sázení.

Různý počet vsázených zápasů pro jednotlivé metody je dán různými hodnotami predikcí jednotlivých metod pro každý zápas. Vzhledem k dostupným kurzům sázkových kanceláří může každá metoda právě vzhledem k dosaženým predikcím označit jinou množinu zápasů, u které se provede fiktivní sázka.

5.8 Použitý software

Pro práci s daty a simulace byl použit matematický systém od společnosti *Wolfram* ***Mathematica*** a rozšiřující knihovna *Neural Package*. Provedené simulace nicméně lze provést ve většině matematických programů podporujících modelování v oblasti neuronových sítí. Pro stahování respektive parsování všech dat z webu [23] byla rovněž použita Mathematica, které tak představuje ucelený nástroj pro veškeré moderní modelování. Jinou možností je například použití software WEKA, Matlab, R a mnoha dalších. Pro výpočet pomocí grafických karet lze například použít software Encog. Především výpočetní systém WEKA, který lze integrovat do systému Mathematica, jej doplňuje o téměř všechny algoritmy a metody (například RandomForest, Adaboost, rozhodovací stromy, integruje například i výpočetní knihovnu libSVM). Zdrojové kódy všech výpočtů, včetně použitých dat jsou na přiloženém datovém nosiči CD (pro spuštění některých výpočtů je nutná knihovna Neural Network).

5.9 Další možná rozšíření

Další možné rozšíření představuje studium konstrukce samotných signálů na neuronech ve skryté vrstvě - tedy popis vztahů mezi vstupní a skrytou vrstvou. V této práce jsme používali pouze jednoduchou dopřednou topologii, další rozšíření může být studium a srovnání proti neuronům a radiálními jednotkami a příbuzné metodě Support Vector Machine nebo struktur obsahující cykly. Jinou oblastí dalšího zkoumání může být přímé srovnání proti metodě RandomForest. V hlavní simulaci 5.7 můžeme dále postupovat například přidáním dalších relevantních regresorů (zranění klíčových hráčů, stav soupisky, motivace týmů vzhledem k pohárovým soutěžím, atd.). Jiné vylepšení může představoval použití kopulí při modelování závislosti mezi náhodnými veličinami počtu vstřelených branek jednoho a počtu vstřelených branek druhého týmu.

Závěr

V této práci jsme se zabývali problematikou umělých neuronových sítí. Tato metoda představuje jednu z mnoha technik strojového učení. V rámci této práce jsme popsali především jednoduchou dopřednou síť. Věnovali jsme se některým výpočetním aspektům při odhadu parametrů. Podařilo se nám vylepšit výchozí odhadovací algoritmus nově navrženým algoritmem TwoStepNNL, jehož oprávněnost se nám podařilo i simulačně ověřit. Pokusili jsme se vnést alespoň základní pohled na vnitřní strukturu umělé neuronové sítě - především přenos informací mezi skrytou a výstupní vrstvou.

V hlavním simulačním příkladu 5.7 jsme porovnali základní lineární modely proti konceptu neuronových sítí v úloze predikce počtu gólů v hokejových zápasech. I přes nezahrnutí mnoha relevantních informací, které mají bookmakeři sázkových kanceláří k dispozici, se nám podařilo postavit matematický model, který se jim v predikci pravděpodobnosti na zvolenou sázkovou událost může vyrovnat.

Představenou techniku lze snadno použít i v jiných podobných oblastech - například v bankovnictví v úlohách kreditního rizika, v pojišťovnictví při modelování storen nebo v mnoha dalších oblastech. Lze ji použít jakožto rozšíření nebo doplnění u nejčastěji používaných metod zobecněných lineárních modelů.

Seznam použité literatury

[1] Šíma J., Neruda R.: *Teoretické otázky neuronových sítí*, MATFYZPRESS, Praha, 1996.

[2] Cipra T.: *Finanční ekonometrie*, EKOPRESS, Praha, 2008.

[3] Wolfram Mathematica: *Documentation center*, [Internet], http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/2.5.3.html

[4] Freund Y., Schapire R.E.: *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences, no. 55. 1997

[5] Breiman L., Friedman J., Stone Ch.J., Olshen R.A.,*Classification and Regression Trees*, Chapman and Hall/CRC, Boca Raton, 1998

[6] Netflix [Internet] http://www.netflixprize.com/rules

[7] Azam F.: *Biologically Inspired Modular Neural Networks*, PhD Dissertation, Virginia Tech. 2000, http://scholar.lib.vt.edu/theses/available/etd-06092000-12150028/unrestricted/etd.pdf

[8] Wikipedie [Internet] http://en.wikipedia.org/wiki/Gray_coding

[9] Kopa M., Hanzák T.: *Materiál z přednášky Kreditní riziko v bankovnictví*

[10] Karlis D., Ntzoufras I.: *Bivariate Poisson and Diagonal Inflated Bivariate Poisson Regression Models in R*, Journal of Statistical Software, September 2005, Volume 14, Issue 10

[11] Ingolfsson A.: *Ice Hockey*, Wiley Encyclopedia of Operations Research and Management

[12] [Internet] http://www.mindcreators.com/NeuronBasics.htm

[13] [Internet] http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Print_Version

[14] [Internet] http://www.stumptown.com/diss/chapter2.html

[15] Allwright J.C.: *Conjugate gradient versus steepest descent*, Journal of Optimization Theory and Applications, volume 20, number 1, 1976

[16] Pradeep T. et al.: *Comparison of variable learning rate and Levenberg-Marquardt back-propagation training algorithms for detecting attacks in Intrusion Detection Systems*, International Journal on Computer Science and Engineering, 2011

[17] Howard J.: *Getting in shape for the sport of data science*, MelbURN Meetup, 2011
http://media.kaggle.com/MelbURN.html

[18] [Internet]
http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[19] [Internet]
http://svmlight.joachims.org/

[20] Cortes C., Vapnik V. N.: *Support-Vector Networks*, Machine Learning, 20, 1995.

[21] Schölkopf B., Smola A. J. *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[22] [Internet]
http://mathworld.wolfram.com/DifferentialEvolution.html

[23] [Internet]
http://www.liga.cz/hokej/usa/nhl/

[24] Branda M.: *Zobecněné lineární modely v pojišťovnictví*, SAV MFF, Praha, 2012
http://artax.karlin.mff.cuni.cz/ branm1am/presentation/Branda_GLM_2012.pdf

[25] Breiman L.: *Random Forests*, Machine Learning 45, 2001

[26] O´Brien R. M. *A Caution Regarding Rules of Thumb for Variance Inflation Factors*, Quality & Quantity, 2007.

[27] [Internet]
http://mathworld.wolfram.com/Ill-ConditionedMatrix.html

Přílohy

Hokej » USA » NHL 2010/2011 — 27.03.2011

Carolina - Tampa Bay
2 : 4
(2:2, 0:1, 0:1)

(přesilovka)	7.	Staal Eric	13. St.Louis Martin	
	19.	Cole Erik	17. Gagne Simon	
			37. Moore Dominic	(přesilovka)
			59. Gagne Simon	(do prázdné sítě)

1X2 (55) | H/A (48) | Over/Under (21) | AH (27) | DC (35)

Sázkové kanceláře: 2		Celkem	Over	Under
expekt	Expekt (www)	4.5	1.42	2.75
tipico	Tipico (www)	4.5	1.32	3.00
Sázkové kanceláře: 18		**Celkem**	**Over**	**Under**
BestBet	Bestbet (www)	5.5	1.88	1.92
bet-at-home	Bet-At-Home (www)	5.5	1.75	1.85
bet24	Bet24 (www)	5.5	1.93	1.87
BetCRIS	BetCRIS (www)	5.5	1.91	1.91
betfair	BETFAIR exchange (www)	5.5	1.95	1.87
RedKings	BetRedKings (www)	5.5	1.88	1.89
betsson	Betsson (www)	5.5	1.88	1.96
bwin	Bwin.com (www)	5.5	1.87	1.95
DOXX.com	DoxxBet (www)	5.5	1.89	1.85
expekt	Expekt (www)	5.5	1.95	1.80
interwetten	Interwetten (www)	5.5	1.85	1.85
JETBULL	Jetbull (www)	5.5	1.88	1.92
LEON BETS.COM	Leon (www)	5.5	1.90	1.86
paf	Paf (www)	5.5	1.75	1.95
Redbet	Redbet (www)	5.5	1.95	1.83
SBOBET	Sbobet (www)	5.5	1.94	1.92
tipico	Tipico (www)	5.5	1.85	1.85
youwin	youwin (www)	5.5	1.94	1.90
Sázkové kanceláře: 1		**Celkem**	**Over**	**Under**
expekt	Expekt (www)	6.5	2.80	1.40

Obrázek 5.16: Příklad kurzů světových kanceláří hraného 27.3.2011 mezi týmy Carolina a Tampa Bay.

www.liga.cz
www.covers.com
www.betexplorer.com

Tabulka 5.10: Webové stránky obsahující pársovaná data.

	Tabulka regresorů ze simulačních úloh
regresor_1	Počet souvislé skupiny minulých zápasů týmu A, u kterých došlo k překročení hranice počtu gólů $h = 5{,}5$ - načítáváme do kladných hodnot. V případě série zápasů nepřekročení hranice 5,5 gólů nasčítáváme do záporných hodnot.
regresor_2	Stejně jako regresor_1 pro tým B.
regresor_3	Podobně jako regresor_1, jen sledujeme sérii výher týmu A - načítávání do kladných hodnot a nevýher - do záporných hodnot.
regresor_4	Stejně jako regresor_3 pro tým B.
regresor_5	Robustnější varianta regresoru_1 - připouští jednu vadu v souvislé posloupnosti zápasů
regresor_6	Stejně jako regresor_5 pro tým B.
regresor_7	Průměr počtu vstřelených branek týmu A za posledních 100 dní.
regresor_8	Průměr počtu obdržených branek týmu A za posledních 100 dní.
regresor_9	Průměr počtu vstřelených branek týmu B za posledních 100 dní.
regresor_{10}	Průměr počtu obdržených branek týmu B za posledních 100 dní.
regresor_{11}	Průměr počtu vstřelených branek týmu A za posledních 100 dní na domácím hřišti.
regresor_{12}	Průměr počtu obdržených branek týmu A za posledních 100 dní na domácím hřišti.
regresor_{13}	Průměr počtu vstřelených branek týmu B za posledních 100 dní na hostujícím hřišti.
regresor_{14}	Průměr počtu obdržených branek týmu B za posledních 100 dní na hostujícím hřišti.
regresor_{15}	Průměr počtu vstřelených branek ve vzájemných zápasech týmů A-B týmem A
regresor_{16}	Průměr počtu vstřelených branek ve vzájemných zápasech týmů A-B týmem B
regresor_{17}	Počet vzájemných zápasů
regresor_{18}	Průměr počtu vstřelených branek ve vzájemných zápasech týmů A-B týmem A za posledních 700 dní
regresor_{19}	Průměr počtu vstřelených branek ve vzájemných zápasech týmů A-B týmem B za posledních 700 dní
regresor_{20}	Počet vzájemných zápasů za posledních 700 dní
regresor_{21}	Průměr získaných bodů (fotbalové počítání) týmem A za posledních 1 200 dní
regresor_{22}	Průměr získaných bodů (fotbalové počítání) týmem A za posledních 350 dní
regresor_{23}	Průměr získaných bodů (fotbalové počítání) týmem A za posledních 100 dní
regresor_{24}	Průměr získaných bodů (fotbalové počítání) týmem A za posledních 30 dní
regresor_{25}	Průměr získaných bodů (fotbalové počítání) týmem B za posledních 1 200 dní

Tabulka 5.11: Seznam regresorů - část 1.

	Tabulka regresorů ze simulačních úloh
regresor_{26}	Průměr získaných bodů (fotbalové počítání) týmem B za posledních 350 dní
regresor_{27}	Průměr získaných bodů (fotbalové počítání) týmem B za posledních 100 dní
regresor_{28}	Průměr získaných bodů (fotbalové počítání) týmem B za posledních 30 dní
regresor_{29}	Počet vstřelených gólů týmu A v jeho posledním hraném zápase
regresor_{30}	Počet obdržených gólů týmu A v jeho posledním hraném zápase
regresor_{31}	Počet vstřelených gólů týmu B v jeho posledním hraném zápase
regresor_{32}	Počet obdržených gólů týmu A v jeho posledním hraném zápase
regresor_{33}	Počet zápasů týmu A v posledních 15-ti dnech
regresor_{34}	Počet zápasů týmu B v posledních 15-ti dnech
regresor_{35}	Počet dní mezi posledním a nadcházejícím zápasem týmu A
regresor_{36}	Počet dní mezi posledním a nadcházejícím zápasem týmu B
regresor_{37}	Průměr získaných bodů (hokejové počítání) týmem A za posledních 1 200 dní
regresor_{38}	Průměr získaných bodů (hokejové počítání) týmem A za posledních 350 dní
regresor_{39}	Průměr získaných bodů (hokejové počítání) týmem A za posledních 100 dní
regresor_{40}	Průměr získaných bodů (hokejové počítání) týmem A za posledních 30 dní
regresor_{41}	Průměr získaných bodů (hokejové počítání) týmem B za posledních 1 200 dní
regresor_{42}	Průměr získaných bodů (hokejové počítání) týmem B za posledních 350 dní
regresor_{43}	Průměr získaných bodů (hokejové počítání) týmem B za posledních 100 dní
regresor_{44}	Průměr získaných bodů (hokejové počítání) týmem B za posledních 30 dní
regresor_{45}	Jednotkový vektor
regresor_{46}	Rok sezóny
regresor_{47}	Počet dní od prvního zápasu v datovém vzorku - časový index
regresor_{48}	Logaritmus předchozího regresoru
regresor_{49}	Indikátor, zda jde o zápas PlayOff nebo základní části
regresor_{50}	Geografická vzdálenost mezi týmy A a B
regresor_{51}	Odmocnina z předchozího regresoru
regresor_{52}	Indikátor, zda jde o derby = vzdálenost týmů do 10km
regresor_{53}	Součet geografických vzdáleností týmu A, nacestovaných za posledních 17 dni
regresor_{54}	Odmocnina předchozího regresoru
regresor_{55}	Součet geografických vzdáleností týmu B, nacestovaných za posledních 17 dni
regresor_{56}	Odmocnina předchozího regresoru

Tabulka 5.12: Seznam regresorů - část 2.