UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

DIPLOMOVÁ PRÁCE

Statistické modely a neuronové sítě

Vedoucí diplomové práce
Mgr. Jana Vrbková
Rok odevzdání: 2012

Vypracovala
Bc. Martina Janáčková
AME, II. ročník

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracovala samostatně podvedením Mgr. Jany Vrbkové s pomoci uvedené literatury a ostatních informačních zdrojů.

V Olomouci dne _____

Bc. Martina Janáčková

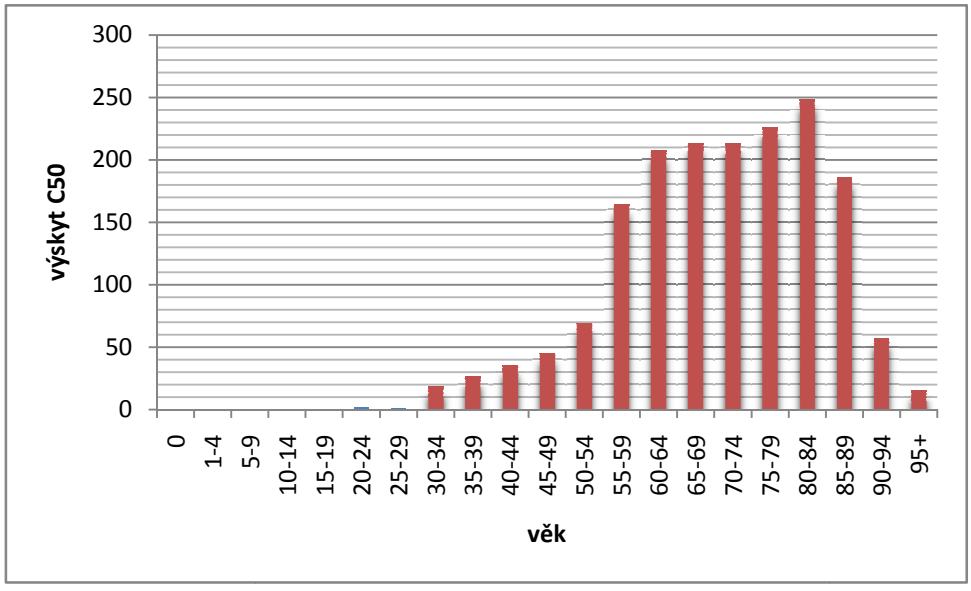
Poděkování

Ráda bych poděkovala mé vedoucí diplomové práce Mgr. Janě Vrbkové, Ph. D. za trpělivost, rady a čas, který mi věnovala při konzultacích. Dále bych chtěla poděkovat mé rodině, která mě podporovala po celou dobu studia.

Obsah

ÚVOD

V dnešní moderní době se může medicína pyšnit úspěchy v léčbě různých druhů rakoviny. Příkladem mohou být vakcíny Silgard a Cervarix, které chrání proti základním typům papilomavirům způsobující rakovinu děložního čípku. I přes to, tu stále existuje mnoho druhů rakovin, u nichž nejsou známy příčiny vzniku, například rakovina prsu. Karcinom prsu je zhoubné nádorové onemocnění, které vzniká ve tkáni prsu. Rakovina prsu se stala jednou z hlavní příčiny úmrtí na celém světě. Výzkum diagnózy rakoviny a následná léčba se stala důležitým tématem pro vědeckou komunitu. Prevence je stále záhada a nejlepší způsob, jak pomoci pacientkám, je včasná detekce. Nelze přesně předpovídat, koho konkrétně toto onemocnění postihne. Jsou známy pouze určité rizikové faktory, které zvyšují šance žen, že onemocní rakovinou prsu. Mezi tyto faktory patří např.: věk, genetické rizikové faktory, rodinná anamnéza a další. Stále se však přesně neví, které z těchto faktorů způsobí, že buňky se stanou zhoubnými (Marceno-Cedeño, 2011). Konkrétně v ČR podle údajů Českého statistického úřadu v roce 2011 zemřelo na diagnózu C50 (zhoubný novotvar prsu) 1725 žen. Dle dostupných statistických údajů se diagnóza C50 drží stále na prvním místě v příčinách úmrtí na zhoubné novotvary, těsně následována rakovinou plic (1675 případů v r. 2011).



Cílem této práce je srovnání třech statistických modelů, které by mohly napomáhat určení zda nález u pacientky je maligní či benigní. Statistické modely jsou použity na datovém souboru wisconsin, ve kterém jsou údaje od 699 žen, u nichž bylo naměřeno

9 hodnot týkající se novotvaru. Modely, které jsem si zvolila, jsou umělé neuronové sítě, stromové struktury a logistická regrese.

První kapitola seznámí čtenáře s umělými neuronovými sítěmi a se vším co s nimi souvisí. Svoji inspiraci našly u biologických neuronových sítí, které jsou zde popsány včetně základní jednotky neuronové sítě, neuronu. Nachází se zde stručná historie vývoje umělých neuronových sítí spolu s oblastmi využití sítí. Popisuji zde matematický neuron, struktury a učení samotných umělých neuronových sítí.

Druhá kapitola pojednává o stromových strukturách. Zabývám se rozhodovacím stromem, zejména klasifikačním stromem a regresním stromem. Je zde i popsán nejpoužívanější algoritmus pro růst stromu, CART. Nastíním nově používaný typ stromové struktury, a to soft tree.

Třetí kapitola stručně uvede čtenáře do logistické regrese.

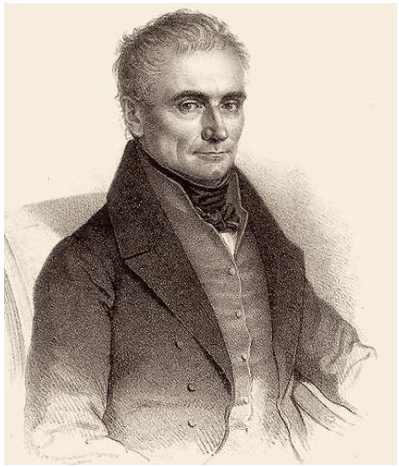
Poslední kapitola je praktická. Zde srovnávám všechny tři modely pomocí různých statistických testu a dívám se na vlastnosti daných modelů.

1. Umělé neuronové sítě (Artificial Neural Networks - ANN)

1.1 Historie problematiky neuronových sítí

Lidé si neustále kladou otázky, čím a jak myslí a co je podstatou myšlení. Již antičtí filosofové si tuto otázku pokládali a snažili se na ni nalézt odpověď. Nejstarší záznamy o poznatcích o mozku pocházejí ze 17. stol. před Kristem, kde slovo „mozek“ bylo zmíněno v souvislosti s popsáním symptomů, diagnózy a prognózy dalšího vývoje operace komplikované fraktury lebky dvou Egypťanů. Důvod proč se lidé neustále zajímají o strukturu a funkci mozku je jediný. Myšlení, resp. jeho úroveň, je hlavním, čím se lišíme od zvířat.

S vývojem dějin, odpovědi na tyto otázky prošly mnoha obměnami. Vyčerpávající odpovědi neznáme a ještě dlouho znát nebudeme, i přes veškeré úsilí, které je této tématice věnováno. V počátečních snahách o poznání podstaty myšlení byl využíván přístup zvnějšku, tzn., že se soudilo převážně podle vnějších projevů. Díky rozvoji fyziologie a neurologie se poznání více přiblížilo struktuře mozku a funkční podstatě jevů myšlení. Dnes se využívá kombinace obou přístupů zvnějšku (tzv. „black box“) i zvnitřku. Pohled zevnitř se více rozvíjel v druhé polovině 20. století., kdy se začaly prohlubovat znalosti o buňkách jako základních složkách živých organismů. Významné poznatky v této oblasti přinesl Jan Evangelista Purkyně (spoluzakladatel cytologie – buněčné biologie) a také Španěl Ramona y Cajala (Novák, 1988).



Obrázek č. 1. Jan Evangelista Purkyně, převzato z *http://3pol.cz/600-jan-evangelista-purkyne*.

V posledních desetiletích se začaly vytvářet matematické modely neuronu. Za průkopníka je považován Američan W. S. McCulloch, který spolu svým studentem W. Pittsem, je autorem prvního reálně použitelného matematického modelu neuronu, uveřejněného v roce 1943. I když jejich model byl několikrát přepracován, doplňován či zlepšován, tvoří základ pro většinu umělých neuronových sítí (Novák, 1998).

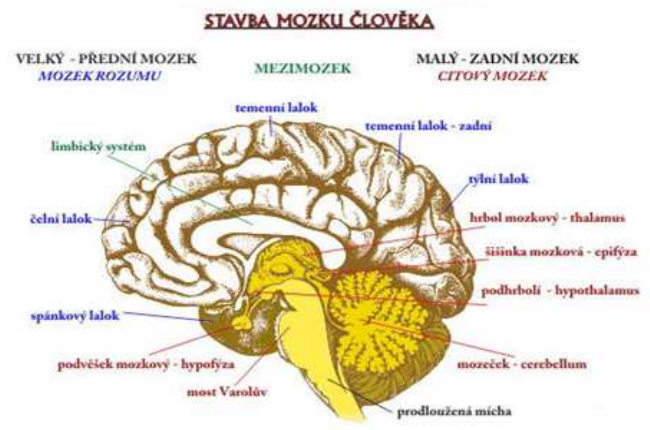
1.2 Biologická inspirace umělých neuronových sítí

Schopnost vnímání podnětů z okolí světa i o svém vlastním stavu a vhodně na tyto podněty reagovat je jednou ze základních vlastností živých organismů. Toto vše zajišťuje informační systém daného organismu. Existence informačního systému je základní podmínka pro přežití a rozvoj všech systémů vůbec (biologických, technických, ekonomických i společenských). U člověka a všech obratlovců je informační systém zajištěn nervovým systémem a mozkem (Novák, 1998).

NERVOVÝ SYSTÉM

Nervová soustava slouží ke kontaktu mezi vnějším prostředním a organismem. Je řídícím a spojovacím systémem uvnitř organismu. Jednotlivé části nervové soustavy jsou navzájem propojeny, aby si pomocí signálů (vzruchů) předávaly zprávy o procesech, které v ní probíhají (Dylevský, 2000). Je nutný pro smyslové vnímání, pro vnímání bolesti, ale i příjemných pocitů. Nervový systém kontroluje pohyb a reguluje tělesné funkce, např. dýchání. Můžeme jej považovat za nejsložitější síť v našem těle, jež má významný vliv na rozvoj řeči, myšlení a paměti (Weston, 2003). Řídícím orgánem nervového systému je mozek a mícha.

LIDSKÝ MOZEK



Obrázek 2. Lidský mozek**,** převzato z *http://comenius2000.webnode.cz/news/lidsky-mozek/*

Mozek se vyvíjí a roste do 12 let života člověka. V porovnání s jinými tvory je lidský mozek poměrně velký, váží 1330 g až 1400 g a spotřebuje 20% veškerého kyslíku, které tělo přijme. Z pohledu umístění je na náročném místě, neboť po celý den musí být krev do mozku přiváděna proti směru gravitace. Mozek je tvořen miliony neurony, které by stačily na několik životů. Život člověka od dob, co jsme se vzpřímili, se prodloužil až na trojnásobek. Délka života po staletí byla nejvýše 35 let, nyní se běžně dožíváme 75 let či více. Díky prodloužení života jsme schopni využívat neurony intenzivněji a po delší dobu (Pfeiffer, 2007). Mozek je banka informací. Jako hlavní a řídící orgán celého těla má za úkol přijímat a ukládat nové poznatky. Tato vlastnost mozku je u všech jedinců spontánní. Základem mozku je hlavní větev, která je křižovatkou pro všechny přicházející a zpracovávané informace. Tuto větev nazýváme *mozkovým kmenem* (thalamus).

Mozek je tvořen specifickými shluky buněk, *nervovými buňkami* čili *neurony*. Jejich počet se odhaduje na 100 miliard. Mají svoji specifickou stavbu těla, která je popsána v následujícím odstavci (tělo, axony, dendrity, atd.). Buňky mozku se od ostatních odlišují enormní vzrušivostí a reaktivitou. Proto má tento systém prvotřídní předpoklady k bleskovému přenosu informací mezi všemi svými částmi. Tyto informace jsou přenášeny formou elektrických impulsů s rychlostí více než 500 km/hod. po výběžcích nervových buněk. Každá nervová buňka má až tisíc těchto výběžků a přenosy na nich trvají pouze milisekundy. V polovině dvacátého století byly objeveny látky, které se vyskytují pouze v mozku a to ve velmi malém množství. Nazývají se *neurotransmitery* (přenašeče). Bez nich by přenos informací v mozku nebyl možný. Jejich prostřednictvím spolu nervové buňky komunikují (Prinke, 2003).

Spolupráce levé a pravé hemisféry je umožněna díky bílé hmotě, která je tvořena nashromážděnými axony. Axony a těla neuronů jsou v mozku uspořádány, nikoliv pomíchány. Buněčná těla neuronu jsou shromážděna na některých místech, stejně jako axony. Myelin obklopující axony má bílou barvu, proto „bílá hmota“. Shluky neuronových buněk (těl bez myelionových pochev) mají naopak šedou barvu (Godaux, 2007).

1.2.1 Biologický neuron

Jak již bylo uvedeno výše, základní jednotkou mozku je nervová buňka neboli neuron. Neuron je vysoce specializovaná živočišná buňka. Vysoká specializace je příčinou, že nervová buňka v dospělosti není schopna se dělit a rozmnožovat (Jelínek, Zicháček, 1996). Tato buňka je schopna přijmout určité formy signálů, odpovědět speciálními

signály, vést je a vytvářet specifické funkční kontakty (synapse) s ostatními neurony a s *receptory* (místa, která reagují na změny z vnějšího nebo vnitřního prostředí převádějí na impulsy) nebo *efektory* (výkonné orgány reagující na příslušný receptor). Neurony se nacházejí vždy v centrální nervové soustavě (CNS – mozek+mícha) nebo v jejím blízkém okolí. Výběžky neuronu leží značně daleko, mimo centrální nervstvo (Pfeiffer, 2007; Jelínek, Zicháček, 1996).

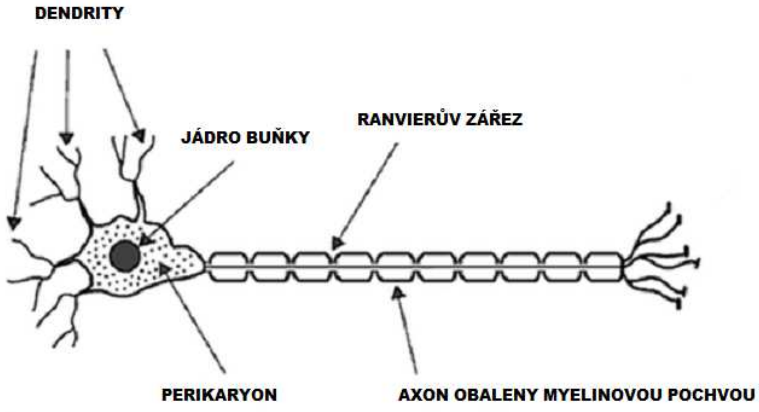
STAVBA NEURONU

Neuron se skládá z buněčného těla, jež může být různého tvaru a velikosti, a výběžků různých délek, nazývaných nervová vlákna - *dendrity* a *axony* (Kopáčová, 1996).

Buněčné tělo (též známe pod názvem perikaryon) je část nervové buňky, která obsahuje jádro.

Dendrity jsou krátké a tenké stromečkovitě rozvětvené výběžky, které vedou vzruch směrem k buňce (dostředivě, aferentně) a představují její vstup (nervový vzruch). V místě odstupu od těla jsou tlusté a postupně se větví. Na povrchu dendritů bývají přítomné dendritické trny, na kterých jsou synapse (Novák, 1988).

Axon neboli neurit je silnější než dendrit a vybíhá z těla neuronu. Axon je hladký a má stejný průměr po celé své délce. U člověka je nejdelší axon asi 1 m dlouhý. Představuje funkční výstup z neuronu. Vede vzruch směrem od těla neuronu (eferentně). Axon obsahuje axoplazmu, která je obalena tenkou membránou (Bocáková, 2004). Povrch některých axonů je chráněn dvojitou pochvou. Vnitřní nesouvislou pochvu, přerušovanou Ranvierovými zářezy, tvoří tukovitá látka (myelin) a zevní pochvu vytvářejí ploché Schwannovy buňky (Jelínek, Zicháček, 1996).



Obrázek č. 3. Biologický neuron, převzato z (Pfeiffer, 2007).

FUNKCE NEURONU

Podle současných poznatků má biologický neuron dvě základní funkce:

- *trofickou funkci* související s tvorbou a odbouráváním některých látek, ze kterých se skládá především tělo neuronu.
- *specifickou funkci*, která značí schopnost vytvářet a přenášet nervové vzruchy pomocí dendritů i axonů a též buněčné membrány neuronu, tj. fyzikálně - chemické změny, přenášené vnitřním informačním systémem neuronu mezi postsynaptickými částmi na jeho dendritech a jeho výstupními synapsemi.

Další funkcí biologického neuronu je shromažďování základních informací ze vstupů, neboli z výstupů neuronu s ním spojeného, zpracovat je a poslat je na výstup. Obecně neuron může mít více výstupů, ale informace procházející neuronem se nedělí, tzn., že každý výstup přenáší stejnou informaci (Novák, 1998).

SYNAPSE

Větve neuritu jsou zakončeny útvary zvanými *boutony*, které se spojí s tělem nebo dendritem jiného neuronu či s jinou buňkou obecně nazývanou *efektor*. Toto spojení nazýváme *synapse*. Synapse jsou velmi složité útvary zprostředkovávající informační styk mezi navzájem spolupracujícími neurony nebo spojení smyslové buňky s neuronem. Každá synapse je tvořena presynaptickým úsekem (zakončení příslušného výběžku axonu), synaptickou štěrbinou (mezera mezi oběma buňkami) a postsynaptickým úsekem (příslušná část navazujícího dendritu, somatu nebo axonu). Jeden neuron může mít několik synapsí. Předpokládá se, že množství synapsí je ovlivněno vnějším prostředím (Kopáčová, 1996).

Další vlastností synapse, kterou popisuje Tučková (2003) je *plasticita*. Souvisí s průchodností synapsí, jež není konstantní. Průchodnost se mění v procesu učení. Existují tři druhy synapsí: chemické, elektrické a mechanické.

- Chemické synapse se dále dělí na
    - excitační - způsobují rozšíření vzruchu v nervové soustavě,
    - inhibiční - naopak zapříčiňují útlum vzruchu.
- Elektrické synapse přenášejí signál průchodem iontů (aktivně nabité částice).
- Mechanické synapse přenášejí signál prostřednictvím styku membrán.

U složitějších synapsí schéma signálových procesů probíhajících uvnitř může být znázorněna jednak graficky, nebo do matice. Matice je tzv. *řídká*. V této matici je většina členů nulových, protože některé přenosy neexistují.

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{bmatrix}$$

Synapse neslouží jen pro přenos signálů mezi neurony, ale slouží pro vznik paměťových stop. V lidském mozku se uplatňují tři hlavní druhy paměťových mechanismů:

- krátkodobá paměť - uskutečňuje se díky cyklickému oběhu vzruchu v uzavřeném okruhu neuronu.
- střednědobá paměť - je založena na změně váhových činitelů synapsí neuronu, jež působí v daném obvodu a opakovaným průchodem vzruchu přes tyto synapse. V synapsích vzniká ribonukleová kyselina, která představuje dlouhodobější charakter změn. Může to být několik hodin, dokonce i dnů.
- dlouhodobá paměť - v období spánku se informace ze střednědobého paměťového mechanismu převádějí na paměť dlouhodobou. Děje se to kvůli otisknutí struktur molekul kyselin do bílkovin působících v jádře neuronu. Některé informace se uchovávají i po celou dobu života a může docházet k předávání těchto informací dalším generacím (Novák, 1998).

PŘENOS SIGNÁLU

Přenos signálů či informací z neuronu do jiných neuronů je složitý elektrochemický proces. Každý neuron je obklopen rozpustnými chemickými ionty vápníku, sodíku, draslíku a chlóru. Sodík a vápník vytvářejí aktivní neuronové odezvy, které se nazývají *akční potenciály* nebo nervové impulzy. Záměna iontů draslíku a vápníku vyvolaná změnou propustností buněčné membrány, způsobí vznik a přenos akčního potenciálu. Pokud nervové impulzy v podobě akčních potenciálů jsou dopraveny do synapsí, propustí jej do dendritů, což má za následek vznik elektrické reakce. Elektrická reakce je buď zesilující (excitační) anebo utlumující (inhibiční) v závislosti na povaze membrán dendritů. Akční potenciál je snižován vstupy z dendritů, které vznikly v inhibičních synapsích.

Efekt zvyšování nebo snižování elektrického potenciálu se realizuje uvnitř těla přijímací buňky. Jestliže tento potenciál dosáhne určitě úrovně, buňka se vybudí a vypudí

signál do axonu, který se v něm rozvětví a po určité době dosáhne synaptického spojení jiných buněk (Míša, 2006).

1.3 Vývoj umělých neuronových sítí

Umělé neuronové sítě (ANN) nacházejí svou inspiraci u biologických neuronových sítí, které byly detailně popsány výše. Předpokládá se tedy, že by se ANN měly chovat stejně nebo podobně jako biologické neuronové sítě. Lidský mozek se svoji vysokou složitostí, propojením neuronů a jejich množstvím, je (alespoň prozatím) velmi těžko napodobitelný, můžeme ovšem simulovat alespoň určité jeho funkce. Hlavní úlohou umělé neuronové sítě je tedy modelování struktury a činnosti biologických neuronových sítí.

Ukládání, zpracování a předávání informací probíhá prostřednictvím celé umělé neuronové sítě, paměť a zpracování informace v neuronové síti je tedy ve své přirozené podstatě spíše globální než lokální (Vondrák, 2001). Mezi základní vlastnosti umělé neuronové sítě patří schopnost učit se - nacházet závislosti v tzv. *trénovacích datech*, ty reprezentovat pomocí synaptických vah - a naučené vědomosti zevšeobecňovat. Zevšeobecňovat znamená správně reagovat na neznámé vstupy, na které nebyla neuronová síť naučena.

O neuronové sítě roste zájem zejména díky možnostem výpočetní techniky, paralelních výpočetních systémů a rozšiřování paměťových možností. ANN se využívají tam, kde nám nevadí případná chyba a kde provádění přesného algoritmu je velice náročné jak po finanční, tak časové stránce. Své opodstatnění mají v případech, kdy není matematicky možno popsat všechny vztahy a souvislosti, které ovlivňují sledovaný proces nebo v případech, kdy jsme schopni sestavit matematický model, ale je příliš složitý pro případnou algoritmizaci (Olej, Hájek, 2010).

- ***1943***

  Američtí vědci Warren McCulloch a Walter Pitts jsou považováni za zakladatele oboru neuronových sítí. Ve svém článku s názvem „*A Logical Calculus of the Ideas Immanent in Nervous Activity*“ uveřejněném v *Bulletin of Mathematical Biophysics* jako první definovali matematický model neuronu. Číselné hodnoty parametrů v tomto modelu byly převážně bipolární. Naznačili zde také, že i ty nejjednodušší typy neuronových sítí by mohly být schopny počítat libovolnou aritmetickou nebo

logickou funkci (Šíma, Neruda, 1996). Typ matematického neuronu zavedený v tomto článku je využívá prakticky dodnes. I když McCulloch a Pitts nepočítali s bezprostředním rozšířením své teorie, natož s praktickým využitím umělého neuronu

- ***1946***

  John von Neumann, autor projektu elektronických počítačů, se nechal do určité míry inspirovat článkem McCullocha a Pittse, při konstrukci svého počítače ENIAC (Olej, Hájek, 2010).

- ***1948***

  Norbert Wiener, jenž je považován za zakladatele kybernetiky, naznačil ve své knize „*Cyberneticsor the Control and Communication in the Animal and the Machine“* rovněž určité koncepty neuronových sítí.

- ***1949***

  Kanadský psycholog Donald Olding Hebb, který se zabýval neuropsychologií, ve které se snažil pochopit souvislost neuronu s psychologickými procesy (zejména s učením), ve své knize „*The Organization of Behavior“* navrhl učící pravidlo pro synapse neuronů. Pravidlo je inspirováno myšlenkou, že podmíněné reflexy, jež můžeme pozorovat u všech živočichů, jsou již vlastností jednotlivých neuronů. Jakmile jsou neurony (neuronová síť) podmíněným reflexům naučeny, nevyžadují „řízení“, tj. na určitou „akci“ přichází téměř samovolně jako odpověď určitá „reakce“.

- ***1954***

  Marvin Minsky, americký kognitivní vědec v oblasti umělé inteligence, stál u zrodu prvního neuropočítače Snark. Snark byl velice úspěšný z technického hlediska, poněvadž dokázal automaticky adaptovat váhy (přizpůsobovat toky signálů umělou neuronovou sítí), ale nikdy nebyl využit pro prakticky. Architektura Snarku byla inspirací pro další vývoj neuropočítačů (Šíma, Neruda, 1996).

- ***1956***

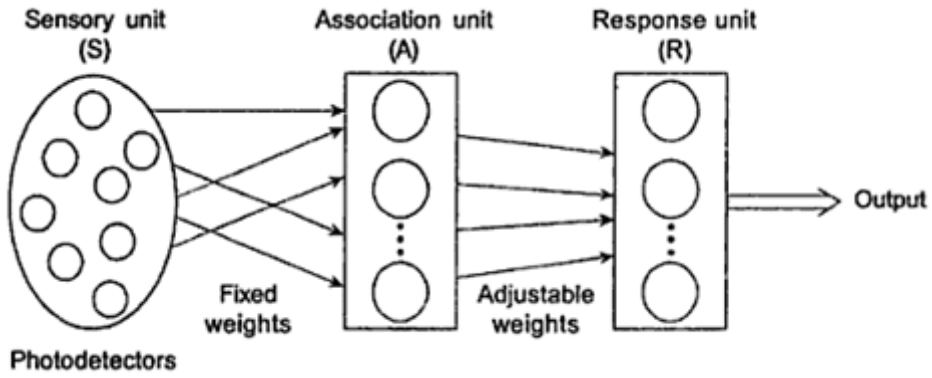
  Rochester, Holland, Habit a Duda se poprvé pokusili o počítačovou simulaci neuronové sítě.

- ***1957***

  V tomto roce Frank Rosenblatt vynalézá tzv. *perceptron*, který je zobecněním McCullochova a Pittsova modelu neuronu pro reálný číselný obor parametrů. Navrhl učící algoritmus, jenž matematicky dokazuje, že pro daná tréninková data nalezne po konečném počtu kroků odpovídající váhový vektor parametrů nezávisle na jeho počátečním nastavení.

ROSENBLANTTŮV PERCEPTRON

Pojem perceptron označuje buď jediný výkonný prvek – matematický neuron nebo síť složenou z těchto prvků. Rosenblantt se nechal inspirovat lidským okem. Perceptron měl napodobovat sítnici oka, tzn. rozpoznávat různé vzory, např. písmena psaná na ploše. Pro zjednodušení byly vzory vepsány do mřížky, kde každé políčko bylo černé nebo bílé, tj. mělo hodnotu 1 nebo 0. Každé políčko mělo svůj čtecí element. Podle fyziologického vzoru je perceptronová síť dvouvrstvá síť. Vstupní vrstva je (množina čtecích elementů) složena z fotodetektorů a tvoří senzorickou jednotku S. Fotodetektory jsou náhodně a neúplně napojeny na prvky vnitřní vrstvy - asociační jednotky A. Neúplné napojení znamená, že každý prvek z vstupní vrstvy nemusí být propojen s každým prvkem z vnitřní vrstvy. Výstupní vrstva (reakční jednotka R) je složena z perceptronů neboli prvků rozpoznávající vzory (Rajasekaran, 2004).



Obrázek č. 4. Roseblanttův perceptron, převzato z Rajasekaran (2004).

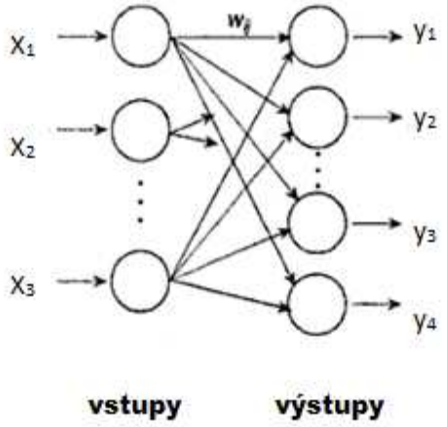
Váhy vstupů jsou konstantní. Adaptace vah nastává až u rozpoznávací jednotky. Pokud je vážená suma vstupů rozpoznávací jednotky menší nebo rovna 0, výstup ze sítě (z jednotky R) je 0, jinak je výstupem samotná vážená suma. Výstup může být určen i pomocí skokové funkce s binárními hodnotami (0/1), popř. bipolární skokové funkce (-1/1). V případě skokové funkce s binárním výstupem tedy

$$y_j = F\left(\rho_j\right) = 1, \qquad pro\ \rho_j > 0$$
$$= 0, \qquad jinak$$

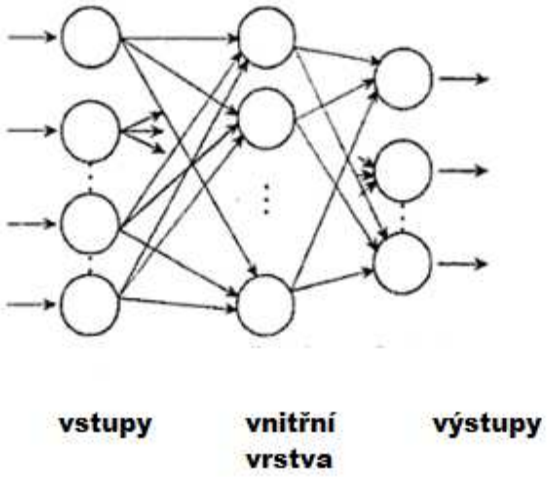
$$\rho_j = \sum_{i=1}^{n} x_i w_{ij},$$

kde x_i je vstup vrstvy R, w_{ij} jsou váhy vedoucí k prvkům výstupní jednotky (R) a y_j je výstup.

Učící algoritmus je typu učení s učitelem. Váhy jsou adaptovány tak, aby minimalizovaly chybu – neshodu vypočteného výstupu s cílovým (požadovaným) výstupem. Na obr. 5 vidíme jednoduchý perceptron a obr. 6 znázorňuje zobecněnou vícevrstvou dopřednou síť typu perceptron.



Obrázek č. 5. Jednoduchý perceptron, převzato z Rajasekaran (2004).



Obrázek č. 6. Model MLP, převzato z Rajasekaran (2004).

Algoritmus učení:

1. krok - inicializace vah w_i , i = 0, …, n a prahu w_0 náhodnými malými čísly

2. krok - předložení vstupu a požadovaného výstupu z trénovací množiny

$$(x_0\ x_1\ \ldots x_n) \rightarrow d$$

3. krok - stanovení skutečné hodnoty výstupu

$$y = F(\sum_{i=0}^{n} w_i x_i)$$

4. krok – adaptace dle následujících pravidel

správný výstup $w_i^{(new)} = w_i^{(old)}$,

výstup 0 a měl být 1 $w_i^{(new)} = w_i^{(old)} + x_i$,

výstup 1 a měl být 0 $w_i^{(new)} = w_i^{(old)} - x_i$,

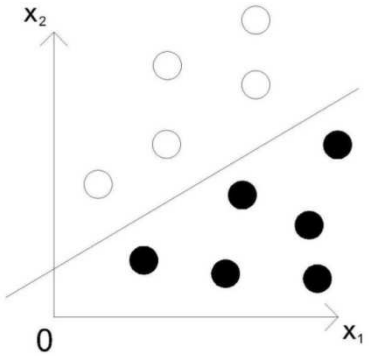
Tuto adaptaci můžeme upravit pomocí *koeficientu učení* (learning rate) η . Tento koeficient ovlivňuje proces adaptace a platí $0 \leq \eta \leq 1$, přičemž pravidla adaptace vah se změní následujícím způsobem:

správný výstup $w_i^{(new)} = w_i^{(old)}$,

výstup 0 a měl být 1 $w_i^{(new)} = w_i^{(old)} + \eta x_i$,

výstup 1 a měl být 0 $w_i^{(new)} = w_i^{(old)} - \eta x_i$,

kde $w_i^{(new)}$ je upravená váha, $w_i^{(old)}$ je stará váha a x_i je vstup. Pokud je koeficient učení η malý, vede to k pomalému učení, velké η vede k rychlému učení (Vondrák, 2001). Samotný perceptron umožňuje pouze klasifikaci do dvou tříd (tj. oddělí pouze lineárně separabilní množiny).



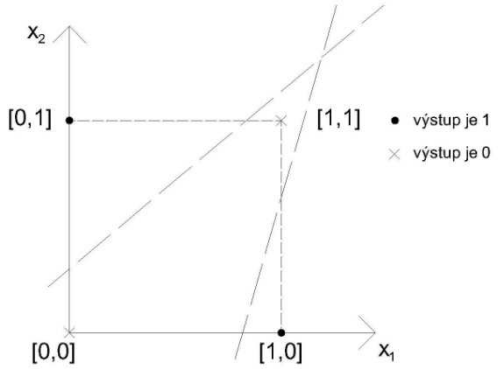
Obrázek č. 7. Klasifikace do dvou tříd.

Nicméně jediný perceptron nevyřeší všechny problémy. Klasickým problémem logické funkce, kterou nelze počítat pomocí jednoho neuronu, je vylučovací funkce XOR. Uvažujeme jen dva binární vstupy a jeden binární výstup, jehož hodnota je 1, pokud právě jedna hodnota je rovna 1. Hodnoty výstupu lze vidět v tab. 1.

Vstupy	Vstupy	Výstup
0	0	0
1	1	0
0	1	1
1	0	1

Tab. 1. Pravděpodobnostní hodnoty XOR.

Z obr. 8 vidíme, že neexistuje přímka (nadrovina), která by oddělila body odpovídající výstupu 1 od bodů odpovídající výstupu 0.



Obrázek č. 8. Funkce XOR.

- ***1957- 1958***

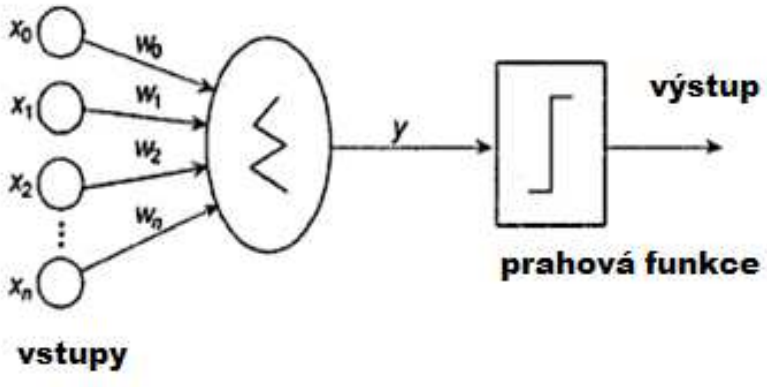
  Rosenblatt spolu s Wightmanem a dalšími spolupracovníky sestrojili první výkonný neuropočítač Mark I Perceptron, který sloužil na rozpoznávání obrazů.

- ***1960***

  Byl vytvořen Adaline (Adaptive Linear neuron), jehož autory jsou Bernard Widrow a jeho studenti. V témže roce Widrow založil první firmu zaměřenou na aplikace neuropočítačů, která vyráběla a prodávala neuropočítače a jejich komponenty.

ADALINE

Jednoduchá síť Adaline má pouze jeden výstupní neuron a hodnota výstupu je bipolární (tj. 1 nebo -1). Vstupy mohou být binární, bipolární nebo reálné. Pokud vážený součet vstupů je větší nebo roven 0, výstup je 1.



Obrázek č. 9. Jednoduchá síť typu Adaline, převzato z Rajasekaran (2004).

Učící algoritmus u Adaline je podobný jako u sítě typu perceptron. Pravidlo je dáno

$$w_i^{(new)} = w_i^{(old)} + \eta(d - y)x_i,$$

kde η je koeficient učení, *d* je cílený výstup, *y* je vypočítaný výstup a x_i jsou vstupy. Adaline se používá například ve vysokorychlostních modemech. Dále se využívá k odstranění ozvěny při telefonních spojeních na dlouhou vzdálenost.

- ***1965***

  Nilsson definuje tzv. učící se stroje (learning machines).

Nastává „doba temna“ pro výzkum ANN. Obor umělých neuronových sítí se potýkal se dvěma problémy. Jedním z problému byl, že mnoho badatelů přistupovali k neuronovým sítím z experimentálního hlediska a nevěnovali pozornost analytickému výzkumu umělých neuronových sítí, které se tak stávaly pouhými „černými skříňkami“. Druhý problém spočíval ve vyřčení příliš silných prohlášení houževnatými výzkumníky, např. že v příštích několika letech bude vyvinut umělý mozek. Nejlepší odborníci se raději přesunuli do jiné oblasti výzkumu umělé inteligence.

Kniha *Perceptrons* od autorů Minského a Paperta se nemalou části podílela na tomto negativním přístupu odborné veřejnosti. V knize popisovali známý fakt, že jeden

perceptron nemůže počítat jednoduchou logickou funkci. Řešením je dvouvrstvá síť se třemi neurony, ale pro vícevrstvý perceptron nebyl znám učící algoritmus. Autoři se domnívali, že algoritmus, který by byl potřeba k vyřešení, je příliš složitý, snad není ani možný. Jejich názor byl odbornou veřejností přijímán a považoval se za matematicky dokázaný. Díky tomuto se vývoj neuronových sítí pozastavil na dalších několik let. Stále tu však byli nadšenci, kteří se ANN věnovali, např. Stephen Grossberg, Teuvo Kohonen či Kunihiko Fukushima.

- ***počátek 80. let***
  Hopfield znovu nastartoval zájem o neuronové sítě. Pomalu vzniká třída tzv. *Hopfieldových sítí*. Později je zformulován princip simulace paměti, neboli uchování informace v dynamickém systému (Olej, Hájek, 2010).

- ***1986***
  Vycházejí články o učícím algoritmu *backpropagation*. V článku autorů Rumelharta, Hintona a Williamse je popisován učící algoritmus zpětného šíření chyb pro vícevrstvou síť. Díky tomuto algoritmu se vyřešil problém Minského z 60. let, který byl popsán výše. Později se ukázalo, že algoritmus backpropagation byl v podstatě znovuobjeven, poněvadž byl využíván již v „období nepopularity“ neuronových sítí jinými vědci, např. Arthurem Brysonem a Yu-Chi Hoem.

- ***1987***
  V tomto roce se konala první velká konference zaměřena na neuronové sítě v San Diegu, které se zúčastnilo 1700 odborníků. Téhož roku byla založena i mezinárodní společnost pro výzkum neuronových sítí INNS (International Neural Network Society, viz http://www.inns.org/). V roce 1988 INNS začala vydávat svůj časopis Neural Networks.

Zájem o neuronové sítě neustále roste. Příkladem je např. diplomová práce z roku 2006 (Görlichová, 2006), ve které autorka uvádí mnoho odkazů na články využívající ANN v medicíně. V dalším textu se pokusím alespoň stručně představit jednotlivé oblasti využití ANN

1.4 Oblasti využití ANN

Umělé neuronové sítě již nejsou jen experimentálními modely napodobujícími fungování lidského mozku, ale slouží k mnoha konkrétním účelům, mezi něž patří následující oblasti.

- Problémy aproximace funkcí:
    - složitý systém lze modelovat pomocí vstupních a výstupních dat.
- Klasifikace do tříd, klasifikace situací:
    - predikce bankrotu podniků či firem, finanční analýzy podniku, modelování ratingu států nebo firem, klasifikace žadatelů o úvěr, atd.
- Rozpoznávání a případná rekonstrukce obrazů (v této oblasti byly neuronové sítě poprvé prakticky využity):
    - většinou dvourozměrné obrazce, vytvořené na dvourozměrné mřížce – *rastru*, jednotlivé pixely rastru jsou navzájem spojeny se vstupními neurony sítě,
    - předpokládáme, že obrazce patří do jedné nebo více kategorií a neuronová síť má za úkol určit, do které kategorie patří obrazec,
    - např. rozpoznávání tištěného nebo rukou psaného písma (čtení směrových čísel na dopisních obálkách, ověřování správnosti podpisu na šecích atd.),
    - rozpoznávání snímků z radaru či sonaru (armádní využití),
    - čtení snímku z RTG, tomografie (zdravotnictví).
- Syntéza řeči (první oblast použití algoritmus zpětného šíření chyb):
    - NETtalk→ neuronová síť je vícevrstvá s jedinou skrytou vrstvou a adaptována algoritmem zpětného šíření chyby (backpropagation),
    - síť je naučena, aby se naučila číst psaný anglický text, tj. musí přiřadit k písmenu foném → v angličtině docela obtížné, poněvadž v určitých slovech se stejné písmeno vyslovuje jinak, např. hat [hæt] a hate [heɪt],
    - po naučení sítě autoři získali přesnost 95% při vyslovování znaků na učebním souboru a 80% úspěšnost při použití na testovaném souboru,
    - firma DEC vyvinula konkurenční systém syntézy řeči, jež nebyl založen na neuronové síti (používal velmi složitý produkční systém s velkým počtem pravidlem), jehož vývoj trval několik let (ve spolupráci

s předními lingvisty), oproti systému s využitím neuronových sítí, kde naučení trvá několik hodin.

- Zdravotnictví:
    - předpovídající modely, tj. např. systémy, jenž na základě křivky EKG předurčí výskyt nemoci,
    - podávání léčiv, kdy neuronová síť pracuje čtyřikrát přesněji než lékaři (automaticky nepodávala antibiotika, na něž byl pacient alergický).
- Sociálně-právní oblast:
    - Spreacher publikoval článek, ve kterém popisuje neuronovou síť, která identifikuje oběti partnerského násilí. Neuronová síť identifikovala 231 z 297 známých obětí domácího násilí, tj. pracovala s přesností téměř 78% (Spreacher, 2004).
- Ekonomická oblast:
    - zvláště tam, kde se vyskytují proměnné v nelineárním vztahu (finanční data),
    - předpověď vývoje jednotlivých ekonomických jevů, např. směnných kurzů, výnosů, cen cenných papírů, inflace či vývoje HDP,
    - modelování reakce spotřebitelů na určitou nabídku (marketing).
- Technologie a výroba:
    - nejčastěji jsou různé systémy rozpoznávání obrazů,
    - prognostické modely, např. předpověď poruch, např. leteckých motorů,
    - kontrolní systémy (kvalita potravin, vůně parfémů apod.).
- Meteorologie:
    - předpovídání počasí (Zvárová, 2009).

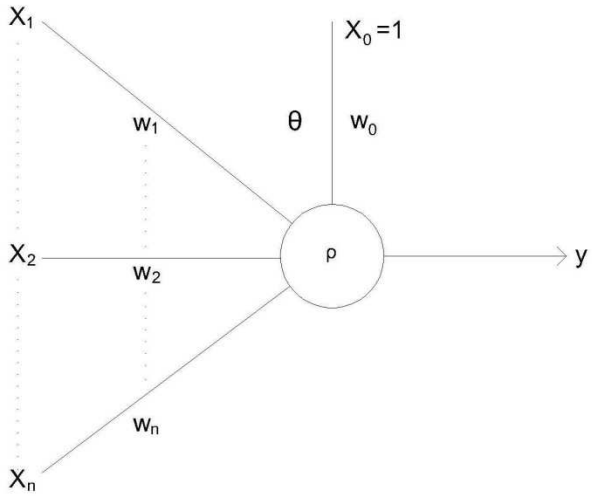
1.5 Matematický model neuronu

Základem matematického pojetí umělé neuronové sítě je *matematický neuronu*. Rozdílnost mezi jednotlivými modely je dána použitím odlišných matematických funkcí nebo různou topologií daného modelu. Matematické modely můžeme dělit (Tučková, 2003):

- podle složitosti
    a) modely první generace – popisují jednodušší biologické neurony,

b) modely druhé generace – popisují složitější chování neuronů.

- podle povahy vstupních dat
    a) binární – nespojité přenosové funkce,
    b) spojité – spojité přenosové funkce.



Obrázek č. 10. Matematický model neuronu.

Matematický neuron (viz obr. č. 10) má konečný počet vstupů (*n* reálných vstupů) $x_1, x_2, x_3, \ldots, x_n$ .Vstupy jsou ohodnoceny reálnými synaptickými vahami $w_1, w_2, w_3, \ldots, w_n$, které určují propustnost neuronu. V podstatě odrážejí uložení zkušeností do neuronu (u pevných vah odborníkem v dané oblasti, u proměnných vah v procesu učení neuronové sítě na tréninkových datech). Čím vyšší hodnota váhy, tím je vstup důležitější (Bíla, 1998). Váhy mohou být i záporné, čímž se vyjadřuje inhibiční charakter vstupu do neuronu, stejně jako v neurofyziologii (viz výše). Kladné váhy tedy budou, pokud jsou synapse excitačního charakteru a záporné, pokud jsou inhibičního charakteru.
Symbol ρ značí *vnitřní potenciál neuronu*:

$$\rho = \sum_{i=0}^{n} w_i x_i .$$

Je to vlastně vážená suma vstupních hodnot, kde w_0 je tzv. *absolutní člen* neboli *bias* a $x_0 = 1$. Symbol θ označuje tzv. *práh neuronu*. Hodnota prahu určuje, kdy je neuron aktivní resp. neaktivní v případě, že neuron funguje na principu „všechno nebo nic“. Jestliže dojde k překročení prahové hodnoty, tj. $\rho > \theta$, neuron se stává aktivním, přenáší signál na výstup (popř. vstup dalšího neuronu v případě neuronů zapojených do sítě).

Obecně je výstup neuronu *y* hodnotou přenosové funkce *F* pro danou hodnotu vnitřního potenciálu ρ. Vstupy matematického modelu neuronu modelují dendrity reálného neuronu a váhy určují synapse. Výstup *y* pak modeluje axon. Vstupní hodnoty $x_1, x_2, x_3, \ldots, x_n$ se tedy v neuronu transformují na výstup pomocí dvou výpočetních postupů, výpočtu vnitřního potenciálu ρ a přenosové funkce $y = F(\rho)$.

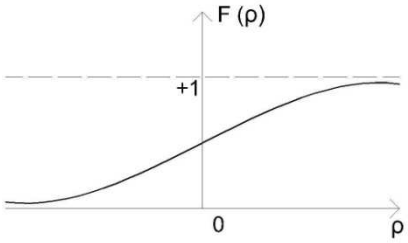
Přenosová funkce má za úkol přeměnit vstupní signál na výstupní signál, obvykle v intervalech hodnot 0 až 1, popř. -1 až 1. Jejím úkolem je rozeznat, zda daná kombinace vstupních signálů je dostatečně významná. Přenosové funkce se obecně dělí na lineární a nelineární, případně spojité a nespojité. Jaký typ funkce zvolíme, závisí na náročnosti výpočtu, času trénování ANN a konkrétním řešeném příkladě. V následující tabulce jsou uvedeny nejpoužívanější přenosové funkce.

Skoková funkce:

$$F(\rho) = \begin{cases} 1, & \text{pro } \rho \geq \theta \\ 0, & \text{pro } \rho < \theta \end{cases}$$

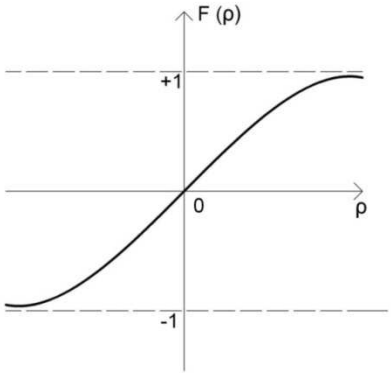
Sigmoidální (logistická) funkce:

$$F(\rho) = \frac{1}{1 + e^{-\rho}}$$



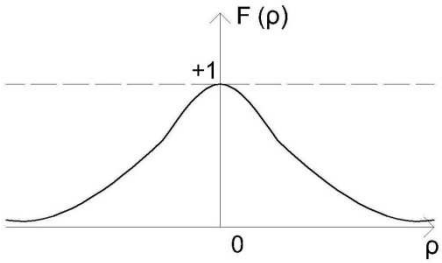
Hyperbolický tangens:

$$F(\rho) = \frac{1 - e^{-\rho}}{1 + e^{-\rho}}$$



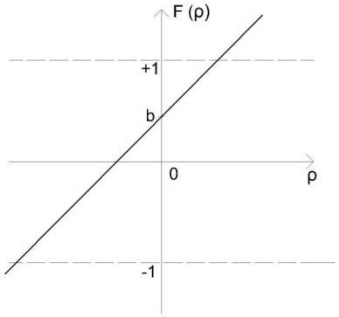
Gaussova funkce:

$$F(\rho) = exp[(-\rho^2)]$$



Lineární funkce:

$$F(\rho) = a\rho + b, \;\; a, b \in R$$



Tab. 2. Přehled základních používaných přenosových funkcí.

Skoková funkce je uplatňována pro neuronové sítě využívané např. pro klasifikaci do dvou tříd, lineární funkce používáme při lineárních aproximačních úlohách. Nelineární spojité přenosové funkce, jako jsou sigmoidální (logistická) funkce nebo hyperbolický tangent, se využívají při modelování nějaké jiné funkční závislosti. Gaussovská přenosová funkce se využívá například v neuronových sítích typu RBF (Radial Basis Function). Vyjadřuje zde míru příslušnosti vzoru k prototypu. Je-li výstup neuronu blízký jedničce, tak také vzor

bude velmi podobný prototypu. Podobnost se uvažuje z pohledu euklidovské metriky, která se používá pro RBF neurony (na rozdíl od perceptronů a MLP sítí popsaných dále, kde je metrikou skalární součin). Více se lze o umělých neuronových sítích typu RBF dočíst například na stránkách Davida Klímy (Klíma, 2002).

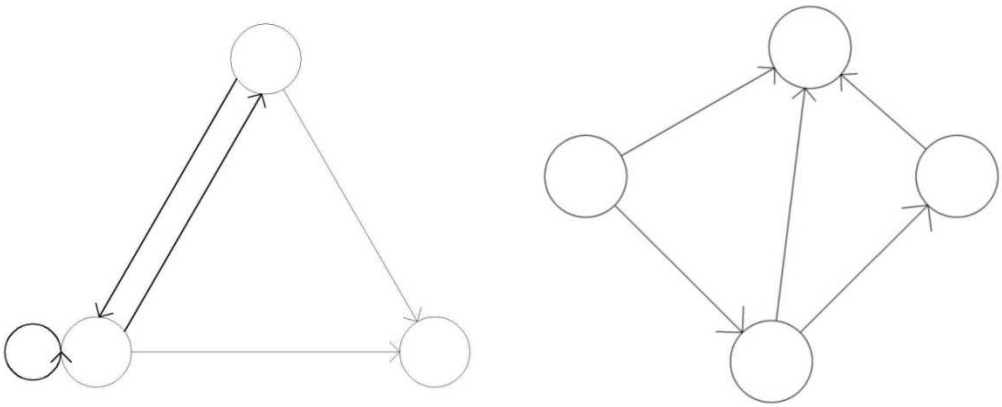
1.6 Charakteristika umělé neuronové sítě

Propojením dvou nebo více matematických neuronů vznikne neuronová síť. Neurony jsou mezi sebou propojeny tak, že výstup jednoho neuronu je spojen se vstupem stejného nebo jiného neuronu. Počet neuronů v sítí a způsob propojení mezi neurony určuje *topologie sítě*. Topologii neboli architekturu neuronové sítě lze graficky vyjádřit orientovaným grafem. Uzly reprezentují neurony a hrany vzájemné propojení neuronů. Životní cyklus každé neuronové sítě můžeme rozdělit na tři etapy.

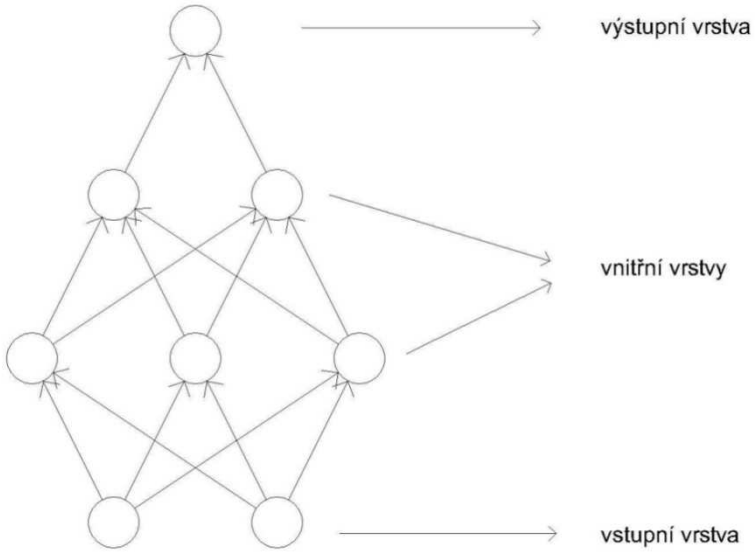
Organizační etapa (změna stavu neuronu).
V této etapě je stanovena architektura sítě a její případná změna (počet neuronů v síti a jejich uspořádání). Existují dva základní druhy architektury sítě:

- cyklická (rekurentní) síť,
- acyklická (dopředná) síť.

Pokud máme cyklickou síť, v grafu topologie sítě se vyskytuje skupina neuronů tvořící cyklus, tj. v této skupině neuronů je výstup prvního neuronu vstupem druhého neuronu, jehož výstup je opět vstupem třetího neuronu atd. Příkladem cyklické architektury je *Hopfieldova síť*, ve které jsou neurony propojeny každý s každým, kromě samy se sebou. Matice vah je u této sítě čtvercová a diagonálně symetrická (na hlavní diagonále jsou 0). Váhy mezi dvěma neurony jsou stejné. Práh neuronu bývá obvykle roven 0 a používá se skoková přenosová funkce. Vstupy mohou nabývat hodnot +1 a -1, případně 0 a +1. Graf acyklické sítě neobsahuje cyklus. Acyklickou síť lze rozdělit do vrstev, které jsou uspořádány nad sebe a spoje vedou pouze z nižších vrstev do vyšších. Jedná se o tzv. *vícevrstvé neuronové sítě*. Například dvouvrstvá síť se skládá z jedné vstupní vrstvy, z jedné vnitřní vrstvy a jedné výstupní vrstvy. Vstupní vrstva se tedy považuje za nultou a započítávají se jen vrstvy vnitřní a vrstva výstupní. Lze se také setkat s označením vícevrstvá síť s *m* vnitřními vrstvami, tj. uvede se explicitně počet vnitřních vrstev.



Obrázek č. 11.: a) Cyklická síť b) Acyklická síť



Obrázek č. 12. Vícevrstvá neuronová síť.

Jednotlivé vrstvy jsou charakterizovány takto:

- vstupní vrstva
  - neurony nejsou nijak propojeny,
  - slouží jako vstupy do další vrstvy,
- vnitřní vrstvy
  - propojují vstupní vrstvu s výstupní vrstvou (v těchto vrstvách bývá více neuronů než ve vstupní a výstupní vrstvě),
  - zpracovávají signál,
- výstupní vrstva
  - převádí výsledky z vnitřní vrstvy na výstupy.

V neurofyziologické analogii vstupní neurony odpovídají receptorům, výstupní neurony efektorům a propojené vnitřní neurony mezi nimi vytvářejí dráhy, po kterých se šíří vzruch. Šíření a zpracování informace na cestě v síti je umožněno změnou stavů neuronů ležících na cestě. Signál mezi vrstvami se obecně může šířit dvěma směry, v přímém (dopředném) směru (feedforward) a ve zpětném směru (backpropagation).

Aktivní etapa (změna stavu neuronů).
V dané etapě se specifikuje nejprve počáteční stav sítě. Stavy vstupních neuronů se nastaví na tzv. vstup sítě. Ostatní neurony jsou ve svém počátečním stavu. Poté dojde v čase k postupným změnám stavů neuronů, provádí se tzv. *výpočet*, a to buď sekvenčně (v daném čase se mění stav jen 1 neuronu) nebo paralelně (svůj stav aktualizuje více neuronů v jednom okamžiku). Výsledek výpočtu sítě jsou hodnoty na výstupních neuronech (Zvárová, 2009; Šíma, Neruda, 1996).

Adaptivní etapa (změna konfigurace – nastavení vah).
Adaptivní etapa určuje konfiguraci sítě (počáteční stavy neuronů) použitou jako počáteční konfiguraci v aktivní etapě fungování sítě a také způsob, jakým se mění váhy v čase. Na počátku se nastaví váhy v sítí na počáteční konfiguraci (např. náhodně). Aktivní etapa fungování sítě (popsaná výše) se využívá k vlastnímu výpočtu fungování sítě pro daný konkrétní vstup, adaptivní etapa slouží k jejímu tzv. učení. Existuje několik učících algoritmů pro různé typy modelů neuronových sítí. Příkladem algoritmu učení je algoritmus backpropagation, který bude blíže popsán v kapitole 4. Tento algoritmus je příkladem tzv. učení s učitelem. Je definována tzv. tréninková množina, která se skládá z dvojic vstup a výstup sítě (tréninkový vzor), a na tyto data se síť postupně adaptuje (Šíma, Neruda, 1996).

1.7 Učení neuronové sítě

Učení je základní vlastností neuronové sítě. V oblasti umělých neuronových sítí pojem učení je synonymem *adaptace*. Jedná se o sbírání a ukládání znalostí. Učení lze definovat jako změna synaptických vah a prahů dle zvoleného algoritmu učení. Činnost neuronových sítí se po stanovení topologie zpravidla dělí na dvě fáze, fáze učení a fáze života.

Fáze učení – je to fáze, kdy se znalosti ukládají do synaptických vah neuronové sítě. V průběhu učení se synaptické váhy mění. Po ukončení fáze učení bude neuronová síť držitelkou znalostí získaných v průběhu učení. Tato fáze odpovídá adaptivní etapě fungování sítě.

Fáze života – v této fázi získané znalosti neuronová síť využije při řešení různých příkladů. Váhy se nemění, mění se pouze stavy neuronů. Tato fáze odpovídá aktivní etapě fungování sítě.

Vlastnost učení odlišuje neuronovou síť od známého používání počítače, kdy musí být vytvořen algoritmus (program), podle kterého přesně krok za krokem probíhá výpočet. Cílem procesu učení je tedy nastavení vah a prahů tak, aby buď odchylka mezi požadovaným a skutečným výstupem při odezvě na soubor trénovacích vzorů byla minimální (učení s učitelem) nebo aby síť byla schopna rozpoznávat vzory, tj. sobě podobné vstupy (učení bez učitele). V procesu učení s učitelem je množina vzorů obvykle náhodně rozdělena na trénovací množinu a testovací množinu. Trénovací množina se používá pouze ve fázi učení. Testovací množina je uplatňována až ve fázi života umělé neuronové sítě. Velmi důležitá je v tomto případě reprezentativnost trénovacích dat. Pod pojmem reprezentativnost je myšleno přibližně stejné množství vzorů ve všech třídách. Třídy jsou vlastnosti, které jsou požadovány na výstupech z natrénované sítě. Pokud jsou trénovací data nevhodně vybrána, negativně to ovlivní kvalitu učení (Olej, Hájek, 2010; Tučková, 2003).

1.7.1 Způsoby učení

Algoritmů učení umělé neuronové sítě je celá řada, proto je dobré si je nějakým způsobem rozdělit. Následující přehled ukazuje jednu z možností, jak takové dělení provést.

1. Dělení dle asociativity:

a) neasociativní učení – neuronové síti je jednou nebo opakovaně předkládán vzor nebo více vzorů, aniž by se předpokládala souvislost mezi vzory. Cílem tohoto učení je zachování vzorů v paměti a jejich následné vybavení,

b) asociativní učení – cílem je vymezení vzájemných vztahů mezi vzory nebo skupinami vzorů.

2. Rozdělení dle přítomnosti „učitele“:

a) učení s učitelem – tento algoritmus učení se vyznačuje tím, že používá srovnání daného výstupu s prediktivním výstupem a přizpůsobuje všechny parametry tomuto srovnání, tzn. prahy a váhy, aby se co nejvíce přiblížily k požadovaným hodnotám. Stanoví se chyba a na jejím základě se budou měnit výše uvedené parametry tak, aby chyba byla co nejmenší. Je nutno stanovit trénovací množinu.

b) učení bez učitele (samoorganizace) – je založeno na schopnosti rozpoznávání vstupních vzorů, které mají stejnou či podobnou vlastnost. Potom neuronové sítě zvládnou třídit vstupy dle daných vlastností. Trénovací množina vzorů není k dispozici. Příkladem sítí s tímto typem učení jsou *Kohonenovy samoorganizující se mapy*, jejichž funkci lze do jisté míry srovnat se shlukováním metodou *k*-průměrů, viz například (Vít, 2012).

3. Dělení dle počtu opakování:

a) jednorázové – v rámci jednoho učícího cyklu dochází k nejlepším výsledkům,

b) opakované učení – neuronové síti jsou opakovaně předkládány vzory a chceme, aby bylo dosaženo minimální odchylky od žádoucího výstupu, přičemž existují různé varianty předkládání vzorů, např. posilované učení, kdy se zasahuje i do průběhu učebního procesu na základě průběžného vyhodnocování (Tučková, 2003).

HEBBŮV ZÁKON

Mezi nejdůležitější principy učení umělých neuronových sítí patří tzv. *Hebbův zákon*, který definoval kanadský psycholog Donald O. Hebb, považovaný za otce neuropsychologie. Tento zákon je základem po všechny typy učení (nejen neuronových sítí) a je aplikován již v prvotních modelech umělého neuronu. Je založen na myšlence, že váhy mezi jednotlivými neurony, které jsou oba v excitačním stavu, budou narůstat.

2. Stromové struktury

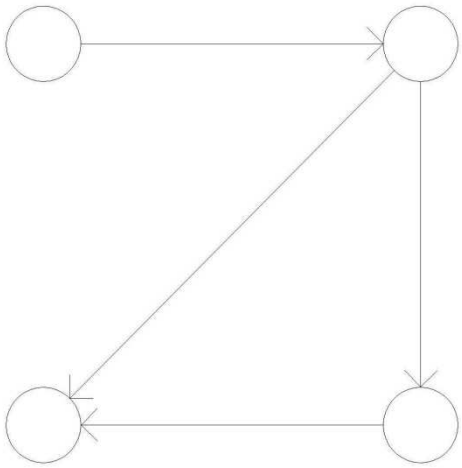
Nejprve uvedu potřebné základními pojmy teorie grafů tak, jak je uvádí Foltýnek (2011). Poté se budu věnovat rozhodovacím stromům, protože je budu využívat v praktické části této práce.

Definice 3.1 *Obecný graf je uspořádaná trojice G = (U, H, f), kde*

- *U je množina uzlů, zpravidla neprázdná a konečná,*
- *H je množina hran, zpravidla je konečná,*
- *f je incidenční zobrazení f: H je incidenční zobrazení f: H→ U^2, které každé hraně h ∈ H uspořádanou dvojici uzlů (x, y) ∈ U^2.*

Definice 3. 2 *Orientovaná hran je uspořádaná dvojice uzlů, pro které platí E ⊆ V x V. E je množina hran V je množina uzlů.*

Orientovaný graf je zadán dvěma konečnými množinami. První z nich je U – množina uzlů a druhá je H – množina orientovaných hran (šipek). Orientovaný graf je určen pravidlem, které stanoví, odkud kam, tedy ze kterého uzlu do kterého daná orientovaná hrana směřuje.



Obrázek č. 13. Orientovaný graf.

Definice 3.3 *Sled délky k v grafu G = (U, H, f) z uzlu v_0 do uzlu v_k je konečná posloupnost tvaru v_0, h_0, v_1, h_1, …., v_k, h_k, kde k ≥ 0, ve které se vždy střídají uzly a hrany (v_i ∈ V, h_i ∈ H pro ∀ i = 0, ..., k) a pro každé j = 1, …, k platí, že f (h_j) = (v_{j-1}, v_j) tedy, že hrana h_j spojuje uzel v_{j-1} a v_j.*

O uzavřený sled se jedná, pokud počáteční uzel sledu je shodný s koncovým.

Definice 3.4 *Tah je sled, ve kterém se neopakuje žádná hrana.*

V tahu jsou všechny hrany vzájemně různé, ale některým uzly může tah procházet opakovaně.

Definice 3.5 *Cesta v grafu je sled, ve kterém se neopakují žádné uzly.*

Cesta obsahuje pouze vzájemně různé uzly, tím pádem se nemůžou opakovat ani hrany a nemůže jít ani o uzavřený sled.

Definice 3.6 *Kružnice je neorientovaný uzavřený tah, ve kterém jsou všechny uzly s výjimkou počátečního a koncového vzájemné různé.*

Definice 3.7 *Orientovaný uzavřený tah, ve kterém jsou všechny uzly s výjimkou počátečního a koncového vzájemně různé, se nazývá cyklus*.

Pokud orientovaný graf neobsahuje cyklus, nazývá se acyklický.

Definice 3.8 *Neorientovaný graf G se nazývá souvislý, jestliže pro každé dva uzly x a y existuje v grafu G cesta začínající v uzlu x a končící v uzlu y.*

Věta 3.1 Pro libovolný graf G = (U, H, f) jsou následující podmínky ekvivalentní:

- pro libovolné dva uzly u, v patřící do U existuje jediná cesta v G z u do v,
- graf G je souvislý a neobsahuje žádnou kružnici,
- graf G je souvislý a platí, že |U| = |H|+1,

kde |U| je počet uzlů a |H| je počet hran.

Definice 3.9 *Graf G splňující ekvivalentní podmínky z věty 3.1 se nazývá strom.*

Definice 3.10 *Kořenový strom je orientovaný graf, ve kterém je vyznačen jeden uzel nazvaný kořen, a platí pro něj následující podmínky:*

- *do kořene nevede žádná hrana,*
- *do každého dalšího uzlu vede právě jedna hrana*
- *všechny uzly jsou z kořene orientovaně dostupné.*

ROZHODOVACÍ STROMY

Pro modelování reálných situací lze využít různé typy *rozhodovacích stromů*. Při tvorbě těchto stromů se používá metoda „rozděl a panuj“ (divide and concquer). Rozhodovací

stromy rekurzivně rozdělují zkoumaný prostor dat dle určitých rozhodovacích pravidel (sadou hierarchicky uspořádaných pravidel), tj. provádějí rozklad tohoto datového prostoru. Rozhodovací stromy převzaly terminologii částečně od živých stromů. Používají se pojmy jako růst stromu, větvení či prořezávání stromu. Na vrcholu rozhodovacího stromu je *kořen*, který představuje celý soubor dat (datový prostor) a postupně probíhá větvení (dělení datového prostoru) do dalších uzlů, tzv. *neterminálních uzlů*. Uzly, které se dále nedělí, se označují jako *terminální uzly* nebo také *listy*. Údaje, jež se týkají vysvětlované proměnné (výstupu), jsou obsaženy v listech. Vnitřní uzly odpovídají podmnožině vzorku dat z uzlu přímo nad ním a reprezentují vždy jedno kritérium pro další dělení dat do skupin. Do grafického zobrazení stromové struktury jsou obvykle přímo vepsány popisy jednotlivých uzlů a hran.

Proces rekurzívního dělení je zastaven, pokud bude splněno kritérium pro zastavení (tzv. stopping rule). Rozhodovací stromy se mohou lišit dle různých charakteristik.

- Podle způsobu větvení:
    - binární – každý uzel se větví pouze na dvě větve (nejčastější),
    - k-nární (k > 2) – uzly se dělí na *k* větví.
- Podle typu výsledného stromu:
    - klasifikační strom – každému listu je přiřazena jedna třída rozkladu datového prostoru,
    - regresní strom – každému listu je přiřazena konstanta (tj. odhad hodnoty závislé proměnné, výstupu).

2.1 Klasifikační stromy

Jednou z oblastí, kde se využívá rozhodovacích stromů je problém *klasifikace.* Jsou to úlohy, v nichž hledáme model, kde závislou proměnnou zařazujeme do disjunktních tříd na základě hodnot nezávisle proměnných (vstupů). Třídy vlastně tvoří rozklad datového prostoru. Ve statistice se tyto úlohy řeší prostřednictvím diskriminační analýzy dat nebo shlukové analýzy dat.

V praxi se klasifikační stromy využívají například v bankovnictví. Banky rozdělují své zákazníky na důvěryhodné a nedůvěryhodné zájemce o úvěr.

Na základě předchozího popisu můžeme tedy říci, že po průchodu klasifikačním stromem budou data vždy zařazena do jedné z klasifikačních tříd

$$C_1, \ldots, C_K, \text{ kde } K \geq 2.$$

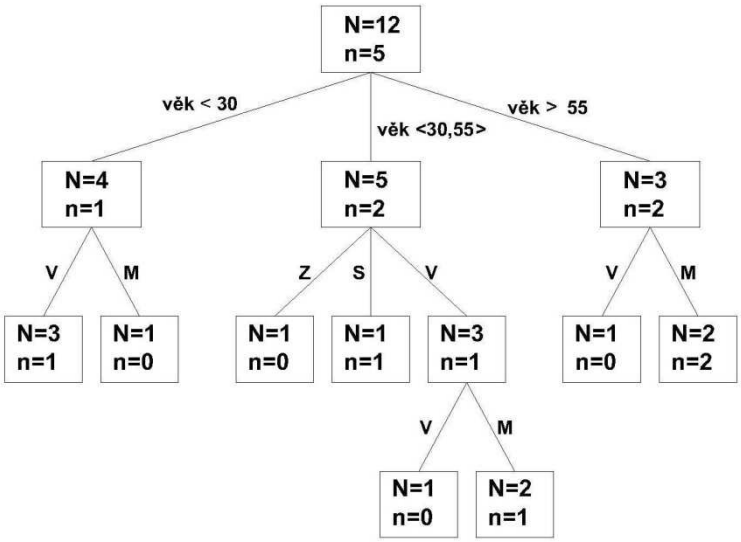
Každé pozorování proměnné Y je charakterizováno vektorem $\boldsymbol{x} = (x_{1,\ldots,} x_m)$ hodnot vysvětlujících proměnných – prediktorů $X_1, \ldots, X_m$. Strom se v každém neterminálním uzlu větví. Nejčastěji jsou používány binarní stromy - binární větvení spočívající v odpovědi na otázku „$x_i < c$?“ pro kvantitativní prediktor X_i , resp. otázku „$x_i \in S$?“, kde S je neprázdná vlastní podmnožina množiny všech hodnot veličiny X_i, pro kvalitativní prediktor. Jedné větvi je přiřazena kladná odpovědi a druhé záporná odpověď. K danému větvení se využívá vždy jen jednoho z prediktorů. Ale stejný prediktor může být použit v jiném větvení. Obdobným způsobem bychom postupovali v případě k-nárních stromů, tj. více variant odpovědí na otázku v uzlu. U klasifikačního stromu patří pozorování y_i vždy do jednoho terminálního uzlu a je mu přiřazena kategorie. Každé pozorování v závislosti na hodnotách prediktorů postupuje od kořenového uzlu přes větvení v neterminálních uzlech až k některému z listů. Množina všech listů určuje disjunktní rozklad prostoru hodnot prediktorů X. Klasifikační strom T s uzly $t = (t_1, t_2, \ldots, t_n)$ určuje klasifikační funkci d_T definovanou na χ s hodnotami v množině $\{C_1, \ldots, C_K\}$.

V následujícím příkladě je ukázáno využití rozhodovacího stromu v klasifikaci. Úkolem je rozdělit respondenty do dvou tříd, zda mají zájem o koupi nabízeného produktu či nikoliv. Údaje, jež jsou k dispozici, se nacházejí v tabulce.

	věk (X_1)	vzdělání (X_2)	bydliště (X_3)	zájem (Y)
1	19	S	V	Z
2	52	V	M	N
3	23	Z	V	N
4	47	S	V	Z
5	38	V	V	N
6	28	V	M	N
7	62	Z	M	Z
8	70	S	V	N
9	33	V	M	Z
10	27	S	V	N
11	59	Z	M	Z
12	31	Z	M	N

Tab. 3. Nasbírané údaje.

Jsou zde tři prediktory, a to věk, vzdělání (Z – základní, S – střední, V - vysoké), bydliště (V – vesnice, M - město) a výstupem Y je klasifikace do tříd: „má zájem o koupi produktu“ (Z), „nemá zájem o koupi produktu“ (N).



Obrázek č. 14. Klasifikační strom.

Symbol *N* označuje počet respondentů v daném uzlu, *n* je počet respondentů mající zájem o koupi nabízeného produktu.

Nejčastěji používaným stromem je binární strom a tedy binární dělení v každém uzlu. Ačkoli se může zdát, že je v mnoha případech lepší dělení na více podskupin, není to obecně dobrá strategie, poněvadž dochází k příliš rychlé fragmentaci dat. Rozdělení do více skupin lze vždy realizovat sérií binárních dělení (Hastie, 2001).

2.2 Regresní stromy

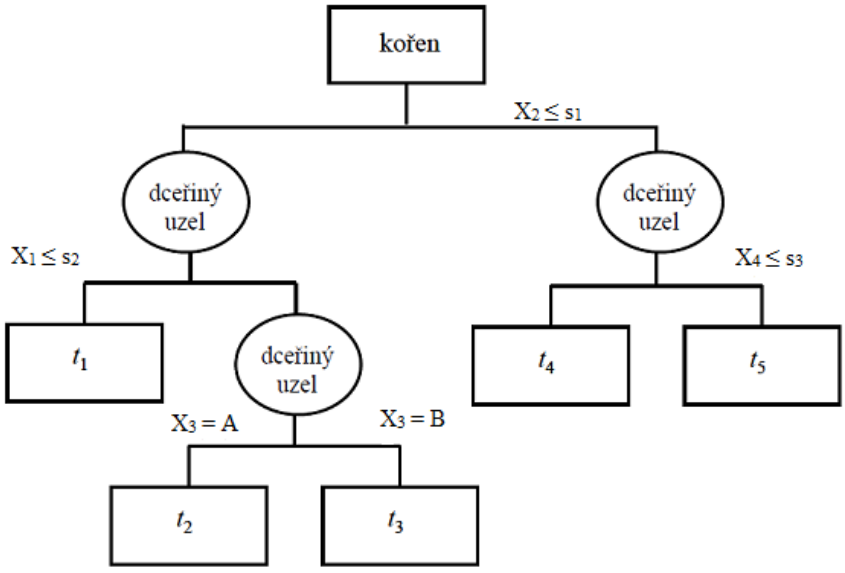
Další způsob využití rozhodovacího stromu je odhad hodnot závislé (vysvětlované) proměnné. Pokud je závisle proměnná spojitá, k vytváření modelu se používají *regresní stromy*. Tyto stromy se využívají například v marketingovém rozhodování při určení potencionálních zákazníků, u kterých je největší šance nákupu výrobku. Z celkové databáze potenciálních zákazníků se náhodně osloví s nabídkou vybraná skupina zákazníků. Výsledky nabídky se analyticky zpracují a vytipují se charakteristické vlastnosti osob, které kladně reagovaly na nabídku. V reklamní kampani se osloví osoby právě s těmito vlastnostmi.

V případě regresního stromu se datový soubor v každém uzlu rozděluje pomocí hodnoty s daného prediktoru X. Pozorování (s hodnotou výstupní proměnné y_i) patří do prvního uzlu, pokud je $x_i \geq s$ (resp. $x_i > s$) a do druhého uzlu pokud je $x_i < s$ (resp. $x_i \leq s$). V průběhu dělení se pozorovaná data z trénovací množiny rozdělují do podskupin tak dlouho, dokud není splněno tzv. *zastavovací pravidlo* (stopping rule). Pro listy stromu (terminální uzly) se vypočte numerická hodnota rovná průměru hodnot výstupní veličiny z trénovacích dat v listu (Komprdová, 2012).

Existuje řada různých algoritmů pro vytváření klasikačních a regresních stromů, např. ID3, C4.5, C5.0, ale mezi nejvýznamnější můžeme řadit algoritmus CART.

2.3 CART

Stromy typu CART se mohou používat pro klasifikační i regresní úlohy. Tyto stromy rostou na základě rekurzivního binárního dělení. Na počátku tvorby stromu patří všechna pozorování do kořene (kořenového uzlu). Poté jsou pozorování rozdělena do dvou dceřiných uzlů, vždy v závislosti na určité hraniční hodnotě (prahu) s nějakého prediktoru X.



Obrázek č. 15. Strom typu CART. Prediktory X_1, X_2, X_4 jsou spojité a X_3 je kategoriální s kategoriemi A, B. Indexy u terminálních uzlů značí pořadí, v jakém došlo k oddělení jednotlivých terminálních uzlů. Převzato z Komprdová (2012).

Otázka nalezení správného rozdělení dat v uzlu prostřednictvím prediktoru *X* spočívá v tom, že se snažíme, aby hodnoty výstupní proměnné Y (příslušnost do třídy v případě klasifikačních stromů, výstupní závisle proměnná v případě regresních stromů) byly uvnitř uzlu co nejhomogennější a současně mezi uzly co nejrozdílnější. Řídíme se tzv. *kriteriálními statistikami* (Impurity measures). Tyto statistiky určují, který prediktor (nezávisle proměnná) zajistí nejlepší rozdělení ve smyslu uvedeném výše a také určí homogenitu uzlu.

KRITERIÁLNÍ STATISTIKY PRO KLASIFIKAČNÍ STROMY

U klasifikačního stromu jsou kriteriální statistiky založeny na poměru pozorování zařazených do dané kategorie závisle proměnné v potenciálních uzlech. Mezi nejznámější kritéria patří Gini index GI, entropie H a klasifikační chyba ME.

Gini index

$$GI = \sum_{C=1}^{K} p_{tC}\,(1 - p_{tC}) = 1 - \sum_{C=1}^{K} p_{tC}^2,$$

Entropie

$$H = -\sum_{C=1}^{K} p_{tC} \log_2 p_{tC},$$

Klasifikační chyba

$$ME = 1 - max\{p_{tC}\},$$

kde p_{tC} je podíl pozorování z kategorie (třídy) *C* v uzlu *t* z celkového počtu pozorování v tomto uzlu (tj. pravděpodobnost kategorie *C* v uzlu *t*), *K* je celkový počet tříd.

Gini index bude nulový, pokud je v konečném uzlu pouze jediná kategorie (jediná hodnota výstupní proměnné *Y*), a dosahuje maxima, jestliže je v terminálních uzlech stejný počet pozorování pro každou kategorii. Entropie a GI jsou jako kriteriální statistika vhodnější, neboť jsou citlivější na změny hodnot p_{tc} než ME.

KRITERIÁLNÍ STATISTIKAPRO REGRESNÍ STROM

U regresního stromu se jako kriteriální statistika používá minimum kvadratické chyby, tj.

$$Q_t(T) = \frac{1}{n_t}\sum_{i=1}^{n_t}\left(y_i - \overline{y_t}\right)^2,$$

$$\overline{y_t} = \frac{1}{n_t}\sum_{i=1}^{n_t} y_{i(t)},$$

kde n_t je počet pozorování v uzlu *t* a $y_{i(t)}$ jsou hodnoty závislé proměnné pozorování v uzlu t.

ALGORITMUS CART

Algoritmus spočívá ve dvou základních krocích. Jsou to: procedura růstu stromu a tzv. *prořezávání stromu*. Procedurou růstu stromu se vytvoří tzv. *maximální strom* T_{max} prostřednictvím rekurzivního dělení. Prořezávání stromu spočívá ve výběru takové posloupnosti podstromů T_{max}, která zahrnuje kompletní informaci obsaženou v původních datech.

Procedura růstu stromu T_{max}:

1. krok - všechna pozorování jsou přiřazena kořenovému uzlu stromu,

2. krok - nalezení nejlepšího rozdělení pro každý prediktor X_j, j=1,..,m, tj. hodnoty prahu (hraniční hodnoty) s. V případě regresního stromu jde o minimalizaci výrazu pro všechny možné hodnoty s.

3. krok - rozdělení souboru do dvou dceřiných uzlů t_1 a t_2, tj. volba prediktoru X_j dle hodnot kriteriální statistiky vypočtené pro dělení z kroku 2.

4. krok - opakování kroku 2 a 3 (rekurzivní dělení) dokud se růst stromu sám nezastaví nebo není splněno pravidlo pro zastavení růstu stromu.

PRAVIDLO PRO ZASTAVENÍ RŮSTU STROMU (STOPPING RULES)

Růst stromu se zastaví sám v následujících případech:

- terminální uzel obsahuje pouze jedno pozorování,
- všechna pozorování v uzlu mají stejnou hodnotu všech predátorů,
- všechna pozorování v uzlu mají stejnou hodnotu závisle proměnné,

nebo jej lze zastavit nastavením některých parametrů:

- maximální počet větvení,
- maximální počet pozorování v terminálním uzlu,

- frakce pozorování v uzlu, která již nemůže být oddělena,
- velikost chyby v potenciálních dceřiných uzlech - k dělení nedojde, pokud např. střední kvadratická chyba nebo podíl nesprávně klasifikovaných pozorování překročí danou hranici (Komprdová, 2012).

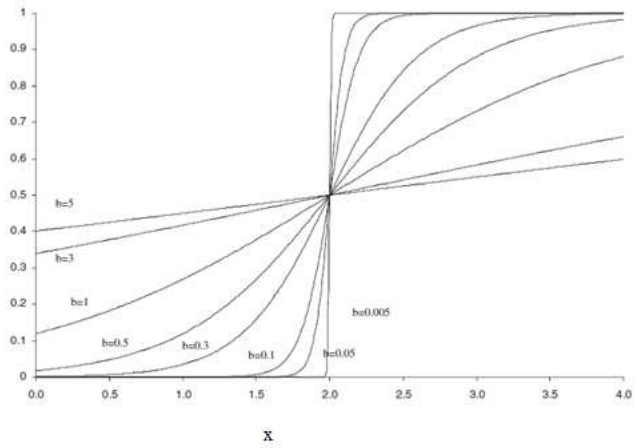
Nejlepší strategií je vytvoření maximálního stromu T_{max} a jeho následné prořezávání prostřednictvím některého z prořezávacích algoritmů, např. tzv. *cost-complexity pruning*, (Hastie, 2001).

2.4 Soft tree

Zvláštním případem klasifikačního stromu je tzv. *soft tree*. S touto strukturou a odpovídajícím algoritmem rozhodování se leze podrobněji seznámit v článku Antonia Ciampiho a Yvese Lechevalliera z knihy autora Jean-Louis Auget. Já se omezím jen na stručné představení toho, jak se *soft tree* liší od klasického rozhodovacího stromu. Rozdíl mezi klasickým rozhodovacím stromem a *soft tree* je v rozhodnutí, kterou větví uzlu se vydáme. Zatímco u klasického rozhodovacího stromu (v literatuře označovaném jako *hard tree*) je rozhodnutí vždy ve tvaru „ jít doleva, jestliže $x_i > c$ nebo $x_i \in S$“, resp. „jít doprava, jestliže $x_i \leq c$ nebo $x_i \notin S$“, tj. tzv. „tvrdé“ rozhodnutí, tak u *soft tree* je tzv. „měkké rozhodnutí“ ve formě „jít doleva s určitou pravděpodobností, jít doprava s doplňkovou pravděpodobností“. Graf stromu *soft tree* je podobný grafu klasického rozhodovacího stromu (viz příklad dále). Zásadním rozdílem je použití uzlové rozhodovací funkce (node decision function), která má obvykle sigmoidální charakter. Nejčastěji se používá logistická funkce

$$g(x) = \frac{1}{1 + \exp^{(-z)}},$$

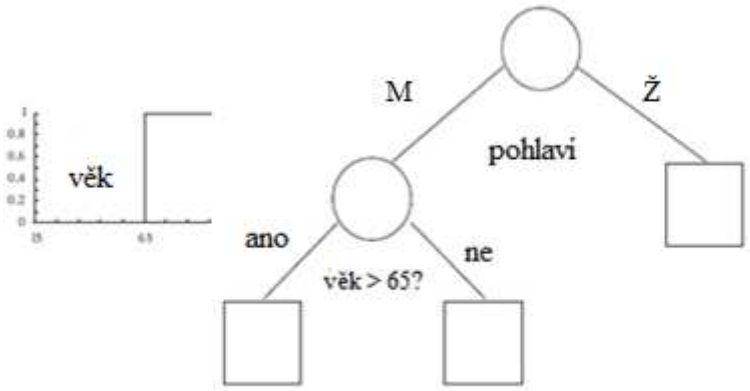
kterou lze pomocí parametru upravit tak, že poskytuje různě „ostrá“ rozhodnutí (viz obr. č. 16).



Obrázek č. 16. Logistická funkce.

Nutno poznamenat, že ačkoli modelujeme pravděpodobnost nastání určitého jevu na základě hodnot jistých vysvětlujících proměnných, a to pomocí stromové struktury, nejedná se v žádném případě o model logistické regrese. Výslednou pravděpodobnost P(Y=1|**X**) modelujeme v podstatě jako směs pravděpodobností, kde „lokální" logistická regrese ovlivňuje hodnotu koeficientů ve smíšeném modelu. Každá taková „lokální" logistická regrese je založena na jediném faktoru, který je vybrán na základě algoritmu konstrukce stromu. Existují i další modely, které rozhodování v každém uzlu modelují např. pomocí neuronových sítí či jiných složitějších postupů. Tyto modely sice poskytují lepší predikci, ale jsou velmi složité (Ciampi, 2002). Modelování rozhodování prostřednictvím *soft tree* nám umožňuje konstruovat jednodušší rozhodovací stromy, ve kterých nemusíme uvádět všechny faktory, které by mohly výskyt dané události ovlivnit a jsou často jen velmi těžko (a draze) zjistitelné. Tyto proměnné lze nahradit jinou, lépe pozorovatelnou, proměnnou a výsledné rozhodnutí na základě hodnoty této proměnné je modelováno prostřednictvím uzlové rozhodovací funkce. Rozhodovací proces je však stále dobře interpretovatelný na rozdíl od „černých skříněk" ANN.

Na obrázcích č. 17 a 18 je jednoduchý strom se dvěma neterminálními uzly (kolečko) a třemi listy (čtverec). Ke každému listu je přiřazena pravděpodobnost jevu, p_k , k = 1, 2, 3. Potom pravděpodobnost nastání nějakého jevu $p(\mathbf{x})$, kterou modelujeme prostřednictvím hodnot vektoru faktorů **x**, je vážený průměr příslušných pravděpodobností p_k s váhami rovnajícími se pravděpodobnostmi, že osoba patří do listu k.

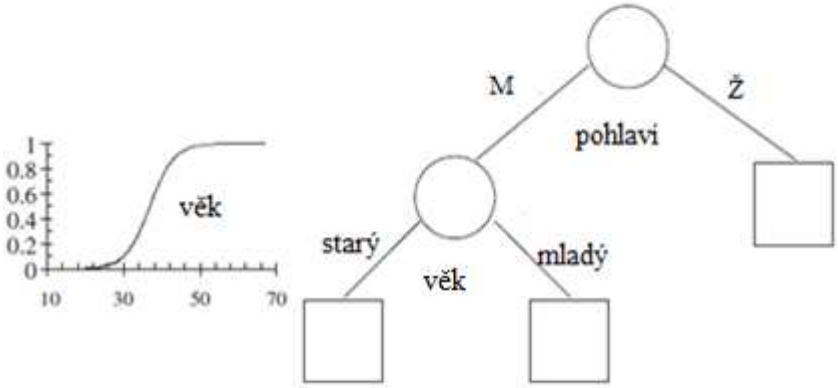


Obrázek č. 17. Hard tree.

Graf u uzlu spojeného s proměnnou věk reprezentuje rozhodovací funkci pro tento uzel. V případě klasického rozhodovacího stromu půjdeme doleva s pravděpodobností 1 a doprava s pravděpodobností 0. Uzlová rozhodovací funkce je skoková a vytváří tedy tzv. tvrdé rozhodnutí. Predikční rovnice (rozhodovací model) pro takový rozhodovací strom můžeme psát ve tvaru

$$p(x) = p_1 I[pohlaví = M] I[věk > 65] + p_2 I[pohlaví = M] I^c[věk > 65] + p_3 I^c[pohlaví = M],$$

kde skoková rozhodovací funkce je symbolicky nahrazena charakteristickou funkcí množiny definované na základě relace, značené $I[\ \ldots]$ a $I^c = 1$



Obrázek č. 18. Soft tree.

V případě *soft tree* rozhodovací struktury na obr. 18 uzlová rozhodovací funkce přiřazená k uzlu spojeného s věkem provádí tzv. měkká rozhodnutí. Velmi staří lidé mají pravděpodobnost rovnu 1 „jít doleva“ a velmi mladí lidé mají pravděpodobnost „jít

doleva“ rovnu 0. Lidé „uprostřed“ jsou rozdělováni do pravé nebo levé větve s pravděpodobností danou rozhodovací funkcí, v tomto případě logistickou funkcí. Predikční rovnice pro tento strom je potom tvaru

$$p(x) = p_1 I[pohlaví = M] g[65] + p_2 I[pohlaví = M] g^c[věk] + p_3 I^c[pohlaví = M],$$

kde *g* je logistická rozhodovací funkce. V grafu *soft tree* struktury z našeho příkladu není uzel s ordinální proměnnou věk popsán otázkou na rozhodnutí vztahující se ke konrétnímu věku (např. věk > 65?), ale pouze označen názvem faktoru (věk). Větve vycházející z daného uzlu jsou potom vždy popsány extrémními hodnotami tohoto prediktoru (Ciampi, 2002).

SOFT TREE VERSUS HARD TREE

Soft tree oproti hard tree je těžko interpretovatelný, ale přináší přesnější výsledky. V článku od autorů Ciampiho, Couturierho a Liho je srovnání hard tree se soft tree. Tyto stromy porovnávali dle čtyř kritérií, odchylka (deviance), plocha pod ROC křivkou, Brierův skór a nesprávná chyba zařazení (misclassification error). Autoři použili datový soubor Pima Indians Diabetes data, který zkoumá výskyt cukrovky u Pima Indiánů.

	soft tree	hard tree
deviance	448.23	453,67
plocha pod ROC křivkou	86,54	85,81
Brierův skór	0,14	0,14
nesprávná chyba zařazení	19,55%	21,05%

Tab. 4. Srovnání soft tree a hard tree.

Jak lze vidět až na jedno kritérium, *soft tree* má lepší výsledky než *hard tree*.

3. Logistická regrese

V 60. letech minulého století byla navržena logistická regrese jako alternativní postup k metodě nejmenších čtverců pro případ, že vysvětlovaná (závisle) proměnná *Y* je binární (0/1). Závisle proměnná *Y* odpovídá výskytu sledovaného jevu. Logistická regrese se v minulosti používala zejména v medicíně. Příkladem může být výpočet rizika vzniku srdeční choroby jako funkce tělesných charakteristik a charakteristik, týkajících se chování - věk, váha, hladina cholesterolu, kouření (Meloun, 2004).

Uvažují se nezávislé náhodné veličiny $Y_1, \ldots, Y_n$ s alternativním rozdělením s parametry μ_i, $i = 1,..,n$. Střední hodnoty těchto náhodných veličin μ_i jsou totožné s pravděpodobnostmi $P(Y_i = 1)$ a mohou záviset na nějakých nenáhodných doprovodných veličinách $\boldsymbol{x}_i = (x_{i1},..,x_{ip})'$. Platí, že $varY_i = \mu_i(1- \mu_i)$. Rozptyl závisí na střední hodnotě této veličiny. Toto je rozdíl v porovnání s lineární regresí, kde je rozptyl konstantní. Pravděpodobnost dvou možných hodnot $Y_i = 1$ a $Y_i = 0$ lze souhrnně zapsat jako

$$P(Y_i = j) = \mu_i^j (1 - \mu_i)^{1-j}, \qquad j = 0, 1.$$

Logaritmickou věrohodnostní funkci pro $\boldsymbol{\mu} = (\mu_1,.., \mu_n)$lze potom psát ve tvaru

$$l(\boldsymbol{\mu}) = log \prod_{i=1}^{n} \mu_i^{Y_i}(1 - \mu_i)^{1-Y_i} = \sum_{i=1}^{n} (Y_i \log \mu_i + (1 - Y_i) \log(1 - \mu_i))$$

$$= \sum_{i=1}^{n} Y_i \log \mu_i + \log(1 - \mu_i) - Y_i \log(1 - \mu_i) = \sum_{i=1}^{n} Y_i \log\left(\frac{\mu_i}{1 - \mu_i}\right) + \sum_{i=1}^{n} \log(1 - \mu_i).$$

Pozorované náhodné veličiny se v logaritmické věrohodnostní funkci projevují v součinech s výrazy $\log(\frac{\mu_i}{1-\mu_i})$. Podíl

$$\omega(x_i) = \frac{\mu_i}{1 - \mu_i} = \frac{P_{x_i}(Y_i = 1)}{P_{x_i}(Y_i = 0)},$$

který porovnává pravděpodobnost jedničky (výskyt sledovaného jevu) a nuly (pravděpodobnost, že se jev nevyskytne) se označuje jako *odds (šance).* Funkce

$$\gamma(\mu) = \log(\frac{\mu}{1-\mu}),$$

se nazývá *logitová*. Tato funkce v tzv. zobecněných lineárních modelech (GLM) hraje úlohu spojovací (linkové) funkce. Předpokládáme, že logit pravděpodobnosti μ_i, $i = 1,..,n$, je lineární funkcí neznámých parametrů $\boldsymbol{\beta} = (\beta_1,..,\beta_p)'$, tj.

$$\gamma_i(\boldsymbol{\beta}) = \beta' x_i.$$

Střední hodnotu EY_i, $i = 1,..,n$, pak můžeme v modelu logistické regrese vyjádřit ve tvaru

$$\mu_i(\beta) = \frac{\exp(\gamma_i(\boldsymbol{\beta}))}{1+\exp(\gamma_i(\boldsymbol{\beta}))} = \frac{\exp(\beta' x_i)}{1+\exp(\beta' x_i)} = \frac{1}{1+\exp(-(\beta' x_i))},$$

což zaručí, že platí $0 < \mu_i < 1$.

ODHAD PARAMETRŮ

Odhad parametrů $\boldsymbol{\beta}$ se provede metodou maximální věrohodnosti. Protože platí

$$\frac{\partial}{\partial \gamma_i}\log(1-\mu_i) = -\frac{\partial}{\partial \gamma_i}\log(1+e^{\gamma_i}) = -\frac{e^{\gamma_i}}{1+e^{\gamma_i}} = -\mu_i$$

a logaritmickou věrohodnostní funkci jsme upravili na tvar

$$\sum_{i=1}^{n} Y_i\,\gamma_i(\boldsymbol{\beta}) + \sum_{i=1}^{n}\log\big(1-\mu_i(\boldsymbol{\beta})\big),$$

můžeme normální rovnici psát ve tvaru

$$\frac{\partial l}{\partial \boldsymbol{\beta}} = \boldsymbol{X}'\big(\boldsymbol{Y}-\boldsymbol{\mu}(\boldsymbol{\beta})\big) = \mathbf{0}.$$

Řešení normální rovnice se hledá prostřednictvím iteračních metod a je implementováno v řadě statistických softwarů, včetně softwaru R, ve kterém je to funkce *glm()* určena pro výpočet zobecněného lineárního modelu, viz praktická část práce (Zvára, 2008).

4. Praktická část

Metody, které jsme zmínila v předcházejících kapitolách, použiji na data z datového souboru *Wisconsin breast cancer databases* (dále v textu označovaného `wisconsin`). Tento soubor je volně dostupný ke stažení na adrese ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin/. Data v souboru se týkají rakoviny prsu. Datový soubor **`wisconsin`** pochází z University of Wisconsin Hospitals, Madison ze státu Wisconis, USA od Dr. William H. Wolberg. Obsahuje údaje od celkem 699 pacientů. Bylo sledováno 10 proměnných, včetně třídy označení, zda je daný nádor benigní či maligní, viz tab. 4. Počet pozorování byl oproti původnímu souboru nepatrně změněn a ze souboru byla vyloučena pozorování, ve kterých se vyskytla chybějící hodnota některé z 10 sledovaných proměnných. Celkový počet pozorování po vyloučení neúplných dat je 683 (`wisconsin2`).

	název proměnné (upravený)	popis	rozmezí hodnot
1	id	identifikační kód pacienta	1-699
2	thickness	tloušťka shluku buněk	1-10
3	uShape	stejná velikost buněk shluku	1-10
4	uSize	stejný tvar buněk shluku	1-10
5	adhesion	marginální přilnavost	1-10
6	cellSize	velikost jedné epiteliální buňky	1-10
7	nuclei	„holé“ jádro	1-10
8	chromatin	nevýrazný chromatin	1-10
9	nucleoli	normální jádra	1-10
9	mitoses	mitóza	1-10
10	malignancy	třída (*klasifikace*)	2-benigní, 4-maligní

Tab. 5. Proměnné datového souboru `wisconsin`.

V následující kapitole stručně představím software R, který budu používat pro práci s datovým souborem.

4. 1 Software R

R je integrované prostředí pro analýzu dat, statistické či matematické výpočty a slouží i ke grafickému znázornění dat. „Rko“ je možno považovat za nástroj programovacího jazyka S, jež byl vyvinut Johnem Chambersem a kol. v Bell Laboratories. Instalační soubory je možné stáhnout zdarma na http://www.r-project.org. Zde jsou i veškeré informace o R. Na této stránce najdeme množství *knihoven*, tzv. *balíčků* (packages), které obsahují více či méně specializované funkce, popř. datové soubory. Knihovny lze instalovat pomocí funkce `install packages()` nebo přímo z menu *Packages* volbou *Install package(s).*

Instalační soubory, které lze nalézt v archivu CRAN na stránkách projektu (viz výše) se liší dle operačního systému. Práce se samotným programem po instalaci je ale v každém OS obdobná. Po spuštění programu je vhodné zvolit pracovní adresář, ve kterém se vyhledávají a ukládají automaticky soubory dat. Lze použít funkci `setwd`(*cesta*), která nastaví adresář na hodnotu danou parametrem *cesta*. Funkcí `getwd()`, která zobrazí pracovní adresář, můžeme toto nastavení ověřit. Další možností je využít menu *File* s volbu *Change dir*… Jednotlivé příkazy se zadávají pomocí klávesnice přímo do okna konzoly (*Console*), popř. se zapisují do textového editoru, který vyvoláme volbou *New script* z menu *File*. Skript, což je obyčejný textový soubor bez formátování, doporučuji uložit s příponou „.r“, aby byl v pracovním adresáři při příštím otevření volbou *Open script* z menu *File* ihned vidět (jinak je zapotřebí změnit filtr souborů v dialogovém okně *Open*). Software R rozlišuje malá a velká písmena, proto je zapotřebí dávat pozor nejen při psaní názvů proměnných, ale také názvů používaných funkcí. Textový výstup se objevuje v konsolovém okně. Klávesa ↑ vyvolá v konzolovém okně předchozí příkazy. Pro grafický výstup slouží grafické okno (lze jej vytisknout nebo jej uložit na disk ve formátu WMF, popř. EPS). Grafický výstup lze rovněž ukládat přímo do souboru odpovídajícího grafického formátu prostřednictvím speciálních funkcí balíčku *graphics* (viz nápověda programu).

Mezi výhody Rka oproti komerčnímu softwaru můžeme zařadit zejména jeho dostupnost (R je volně dostupný a šiřitelný), komptabilita (dostupný pro Linux, Macintosh i Windows), variabilita použití (množství balíčků s rozšiřujícími funkcemi). Mezi jeho

nevýhody patří snad jedině závislost na velikosti operační paměti a nutnost znalosti alespoň základů programování kódu, tj. jistá menší uživatelská přívětivost.

4. 2 Umělé neuronové sítě v softwaru R

V softwaru R je možné instalovat a využívat několik balíčků, které pracují s umělými neuronovými sítěmi. Mezi takové patří např. *neuralnet*, *nnet* a *AMORE*. Balíček *neuralnet* byl vytvořen proto, aby učil síť typu MLP (viz dále) ve smyslu regresní analýzy, tj. aproximace funkčního vztahu mezi vstupními (nezávisle) proměnnými a výstupními (závisle) proměnnými. Pro vlastní proces učení je možné zvolit klasický algoritmus backpropagation nebo tři modifikované verze tzv. *resilient backpropagation algoritmu*. Poněvadž budu právě tento balíček aplikovat na konkrétní data, popíšu princip fungování vybraných funkcí balíčku v následujícím textu spolu s nezbytnou teorií.

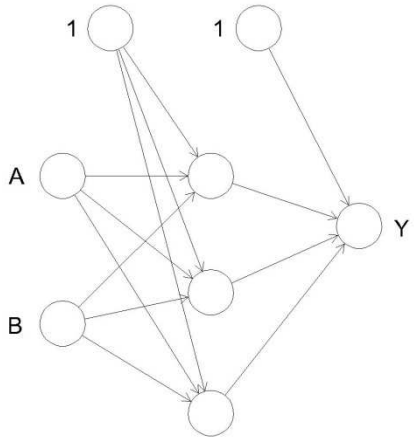
Neuralnet je velmi flexibilní balíček. Jak již bylo uvedeno, obsahuje funkce k trénování dopředné neuronové sítě. Současně poskytuje uživateli možnost výběru přenosové funkce, chybové funkce a také určení počtu skrytých vrstev. Teoreticky můžeme pracovat s libovolným počtem vstupních a výstupních proměnných, stejně jako s libovolným počtem skrytých vrstev a neuronů v nich. Balíček *neuralnet* rovněž poskytuje možnost výslednou síť (její topologii a nastavení všech vah) vizualizovat. Jediné, co v balíčku na první pohled schází, je možnost otestovat fungování vytvořené neuronové sítě na testovacích datech. Balíček *neuralnet* slouží tedy pouze pro vytvoření umělé neuronové sítě.

Dříve než se zaměřím na popis vlastních funkcí balíčku *neuralnet*, představím v následujícím textu princip fungování vícevrstvé sítě typu perceptron (Multi-Layer Perceptron – MLP), kterou prostřednictvím balíčku *neuralnet* konstruujeme a využíváme.

4.2.1 Multi – Layer Perceptron (MLP)

Vícevrstvá síť typu perceptron dobře modeluje funkční vztahy. Základem struktury tohoto typu umělé neuronové sítě je orientovaný graf. Každý neuron vykonává jednoduchou úlohu, zpracovává informace převáděním obdržených vstupů na zpracované výstupy. V MLP je tok informací jednosměrný, od vstupní přes skrytou až po výstupní vrstvu (dopředná síť, viz výše). Na obrázku č. 19 vidíme příklad umělé neuronové sítě s jednou skrytou vrstvou obsahující tři neurony. Podobný graf je možné pro konkrétní vytvořenou umělou neuronovou síť funkcí `plot()` zobrazit (verze funkce `plot()` v balíčku

neuralnet, upravená na základě principu polymorfismu pro objekty třídy *nn* definované v balíčku *neuralnet*).



Obrázek č. 19. Umělá neuronová síť s jednou skrytou vrstvou.

Je dokázáno (Hornik et al., 1989), že již jedna skrytá vrstva, tj. dvouvrstvá síť MLP je dostatečná pro modelování libovolné po částech spojité funkce. Mějme MLP s jednou skrytou vrstvou, která obsahuje *J* neuronů. Potom hodnota výstupu $o(\boldsymbol{x})$ - vypočtená výstupní hodnota jediného neuronu výstupní vrstvy Y pro daný vektor vstupů $\boldsymbol{x}$ - je dána vztahem

$$o(\boldsymbol{x}) = F\left(w_0 + \sum_{j=1}^{J} w_j\, F\left(w_{0j} + \sum_{i=1}^{n} w_{ij}\; x_i\right)\right) = F\left(w_0 + \sum_{j=1}^{J} w_j\, F\left(w_{0j} + \boldsymbol{w_j^T x}\right)\right),$$

kde F je přenosová funkce, w_0 je váha odpovídající konstantnímu vstupu $x_0 = 1$, w_j, $j = 1, .., n$ jsou váhy odpovídající synapsím (propojení) *j*-tého skrytého neuronu výstupního neuronu *Y*, $\boldsymbol{w_j} = (w_{1j}, \ldots, w_{nj})$ označuje vektor všech synaptických vah, odpovídajících propojením *j*-tého skrytého neuronu a *n* vstupů, $\boldsymbol{x} = (x_1, \ldots, x_n)$ je vektor všech vstupních proměnných (vstupů sítě).

Uvedený vztah naznačuje, že neuronové sítě jsou vlastně přímým rozšířením zobecněných lineárních modelů. Obvykle se požaduje, aby přenosová funkce *F* byla nerostoucí, nelineární a diferencovatelná. Příklady vhodných a nejčastěji používaných funkcí byly uvedeny výše (viz tab. 2).

4.2.2 Algoritmus backpropagation

Tento algoritmus, jak již bylo uvedeno, patří mezi algoritmy učení s učitelem, tj. vyžaduje rozdělení vstupních dat na trénovací množinu a testovací data. Parametry umělé neuronové sítě jsou vlastně jednotlivé váhy, které se ve fázi učení sítě nastavují na svou počáteční hodnotu pro fázi života ANN.

Algoritmus backpropagation popíšu tak, jak je implementován v balíčku *neuralnet*. Všechny váhy jsou v procesu učení nejprve inicializovány na náhodnou hodnotu vygenerovanou z normovaného (standardizovaného) normálního rozdělení. Iterační proces učení sítě pak probíhá v těchto základních krocích:

1. krok - výpočet výstupu o_{lh}, $l = 1,..,L$, $h = 1,..,H$ pro daný vstup **x** a aktuální nastavení vah.

2. krok - výpočet tzv. chybové funkce (error function) pro vypočtený výstup o_{lh}, $l = 1,..,L$, $h = 1,..,H$, a pozorovaný výstup y_{lh}, $l = 1,..,L$, $h = 1,..,H$, jakou je např. suma čtverců odchylek, tj.

$$E = \frac{1}{2}\sum_{l=1}^{L}\sum_{h=1}^{H}(o_{lh} - y_{lh})^2$$

nebo tzv. cross-entropy

$$E = -\sum_{l=1}^{L}\sum_{h=1}^{H}(y_{lh}\log(o_{lh}) + (1 - y_{lh})\log(1 - o_{lh})),$$

které jsou mírou diference mezi predikovaným a požadovaným výstupem, kde indexy $l = 1, \ldots, L$ označují pozorování (danou dvojici tréninkových dat) a $h = 1,..,H$ konkrétní výstupní neuron (jeho výstup).

3. krok – adaptace vah dle pravidla učícího algoritmu (bude popsáno dále pro vlastní algoritmus backpropagation a jeho varianty).

Iterační proces je ukončen v okamžiku, kdy je splněno předem definované kritérium, např. absolutní hodnoty parciálních derivací chybové funkce podle vah ($\partial E/\partial w$) jsou menší než daný práh.

Tradiční backpropagation algoritmus učení ANN modifikuje váhy sítě tak, aby bylo dosaženo lokálního minima chybové funkce. Za tímto účelem je vypočten gradient chybové funkce vzhledem k vektoru vah a nalezne se kořen rovnice $dE/d\boldsymbol{w} = 0$. Váhy jsou tedy modifikovány v opačném směru, než má parciální derivace, dokud není dosaženo

lokálního minima. Je-li parciální derivace záporná, váha se zvětší, je-li naopak kladná, váha se zmenšuje. Všechny varianty algoritmu backpropagation se snaží minimalizovat chybovou funkci přidáním koeficientu rychlosti učení vahám, které jdou proti směru gradientu. Na rozdíl od tradičního algoritmu backpropagation, jeho modifikace používají k přepočtu jednotlivých vah w_k ještě tzv. *separátní koeficient rychlosti učení* (separate learning rate) η_k, který může být rovněž v procesu učení modifikován. Navíc je zde pro přepočet vah využito pouze znaménko parciální derivace namísto její velikosti. Váhy jsou tedy v modifikovaném algoritmu backpropagation upravovány dle vztahu

$$w_k^{(t+1)} = w_k^{(t)} - \eta_k^{(t)}.sign\left(\frac{\partial E^{(t)}}{\partial w_k^{(t)}}\right),$$

zatímco v tradičním backpropagation algoritmu dle vztahu

$$w_k^{(t+1)} = w_k^{(t)} - \eta.\frac{\partial E^{(t)}}{\partial w_k^{(t)}},$$

kde *t* označuje *t*-tý iterační krok a *k* danou konkrétní váhu.

Separátní koeficient rychlosti učení se zvětšuje, pokud zůstává zachováno znaménko parciální derivace. Pokud se toto znaménko mění, koeficient η_k klesá. Změna znaménka parciální derivace naznačuje, že bylo minimum chybové funkce v iteračním kroku přeskočeno. Globálně konvergentní algoritmus byl představen Anastasiadisem (2005), který modifikoval koeficient rychlosti učení sítě s ohledem na velikosti všech separátních koeficientů rychlosti učení.

FUNKCE `neuralnet()`

Funkce `neuralnet()` balíčku *neuralnet* se používá pro trénování neuronové sítě. Funkce má následující syntaxi:

```
neuralnet(formula, data, hidden = 1, threshold = 0.01,
          stepmax = 1e+05, rep = 1, startweights = NULL,
          learningrate.limit = NULL,
          learningrate.factor = list(minus = 0.5, plus =1.2),
          learningrate=NULL, lifesign = "none",
          lifesign.step = 1000, algorithm = "rprop+",
          err.fct = "sse", act.fct = "logistic",
          linear.output = TRUE, exclude = NULL,
          constant.weights = NULL, likelihood = FALSE)
```

Vybrané argumenty a parametry funkce jsou popsány v tab.5.

parametr	popis
formula	symbolický popis modelu, který má být fitován, ve tvaru formule
data	obsahuje proměnné určené v *formula*
hidden	vektor určující počet skrytých vrstev a skrytých neuronů v každé z nich, např. (3,2,1) označuje 3 skryté vrstvy, v první vrstvě jsou tři skryté neurony, ve druhé vrstvě dva skryté neurony a v poslední vrstvě je jeden skrytý neuron. Výchozí hodnota 1.
treshold	číslo určující práh pro parciální derivace chybové funkce, jako kritérium zastavení procesu. Výchozí hodnota 0,01.
rep	počet opakování trénovacího procesu. Výchozí hodnota 1.
startweights	vektor obsahující předem určené výchozí hodnoty pro váhy. Výchozí hodnota: váhy jsou náhodná čísla ze standardního normálního rozdělení.
algorithm	říká, který typ algoritmu bude použit
err.fct	diferencovatelná chybová funkce používá se “sse” a “ce”. Výchozí hodnota je “sse”.
act. fct	diferencovatelná přenosová funkce, možné jsou „logistic“ nebo „tanh“. Výchozí hodnota je funkce „logistic“.

Tab. 6. Argumenty a parametry funkce `neuralnet`.

Funkce `neuralnet()` vrací jako svou návratovou hodnotu objekt třídy *nn*, což je hodnota proměnné typu `seznam(list)`. Nejdůležitější složky tohoto seznamu jsou popsané v tab. 6 a lze se k nim dostat prostřednictvím $ (viz praktický příklad uveden dále).

net.result	výsledky (seznam) aplikace sítě na každé pozorování (o_{lh})
weights	váhy ANN, dle jednotlivých vrstev a propojení (seznam)
result.matrix	souhrnná matice výsledků obsahující informace o počtu iterací, skutečně dosažené prahové hodnotě, chybě, kritériích AIC a BIC (jsou-li požadována) a váhy
startweights	startovací váhy

Tab. 7. Nejdůležitější složky návratové hodnoty objektu třídy *nn*.

Funkci `neuralnet()` použiji pro konstrukci ANN z dat `wisconsin2` (viz výše).

Výstupní proměnnou bude proměnná malignancy *class* určující zařazení do klasifikační třídy maligních nádorů. Původní proměnnou (class) s hodnotami 2 (benigní) a 4 (maligní) jsem tedy přejmenovala a změnila na binární (0 – benigní, 1 - maligní). Formule, která je prvním argumentem funkce *neuralnet ()* je ve formátu závislosti výstupní proměnné na proměnných vstupních, přičemž výstupní proměnná je od vstupů oddělena znakem „~" a vstupy jsou odděleny znakem "+" (viz zápis kódu dále). Přenosová funkce je nastavena defaultně jako logistická funkce prostřednictvím parametru *act.fct* a chybovou funkci jsem prostřednictvím parametru *err.fct* nastavila jako funkci *cross-entropy* (hodnota „*ce*"). Jinou možností je hodnota „*sse*" odpovídající střední kvadratické chybě. Hodnota parametru *linear.output* je logická. Logická hodnota určuje, zda má (hodnota FALSE) či nemá (hodnota TRUE) být přenosová funkce aplikována také na výstupní neurony. Dále nastavuji počet opakování algoritmu (1x), počet skrytých vrstev a počet neuronů v těchto vrstvách parametrem *hidden* (hodnota 2 odpovídá jedné skryté vrstvě se dvěma neurony). Parametr *data* určuje zpracovávanou datovou množinu a umožňuje přímý přístup k proměnným bez operátoru *$*. Výstup funkce je uložen do proměnné *nn*.

```
nn <-
neuralnet(malignancy~thickness+uSize+uShape+adhesion+cellSize+nuclei+chromatin+
          nucleoli+mitoses, data=wisconsin2, rep=1, hidden=2, err.fct="ce",
          linear.output=FALSE)
```

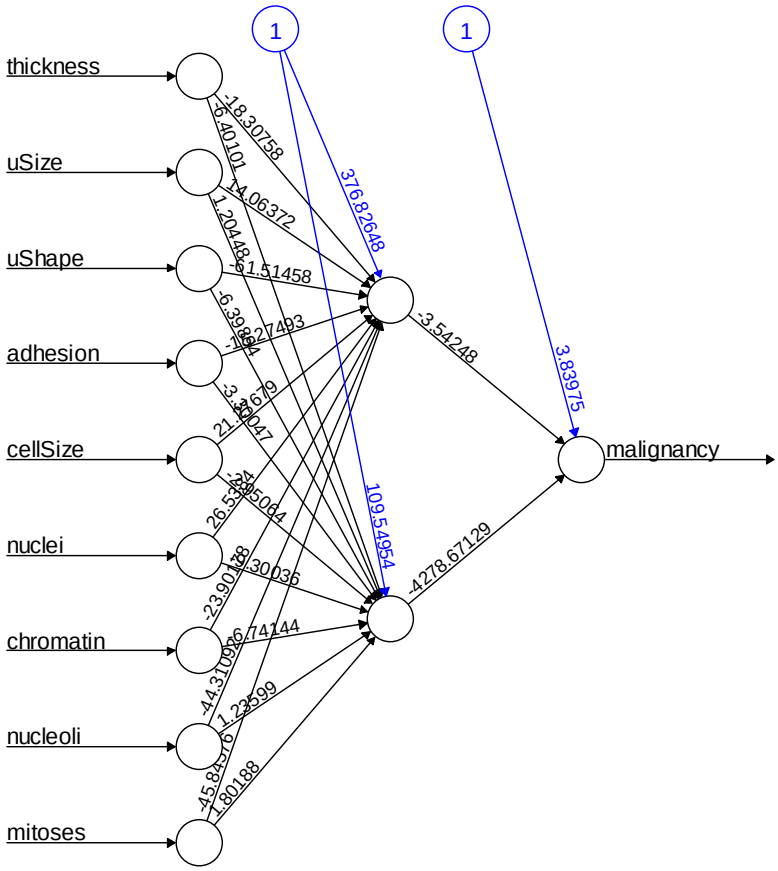
Nejzajímavější částí výstupu je matice *result.matrix*, z níž je vidět, že v jednom opakování algoritmu (matice má jen jeden sloupec) bylo dosaženo odpovídajícího kritéria pro zastavení až v 72636 iteraci. Kritériem byla velikost parciální derivace chybové funkce menší než implicitně stanovený práh 0.01 (lze měnit prostřednictvím parametru *threshold*). Skutečná hodnota této derivace je přibližně 0.00935 (*reached.threshold*). Hodnota samotné chybové funkce v okamžiku zastavení algoritmu je přibližně 38.85352.

```
error                       38.853516566416
reached.threshold            0.009352299928
steps                    72636.000000000000
Intercept.to.1layhid1      376.826480646124
thickness.to.1layhid1      -18.307579473158
uSize.to.1layhid1           14.063719646147
uShape.to.1layhid1         -61.514577406145
adhesion.to.1layhid1       -15.274927318517
cellSize.to.1layhid1        21.276791304547
nuclei.to.1layhid1          26.538400491591
chromatin.to.1layhid1      -23.901783327675
nucleoli.to.1layhid1       -44.310918352607
mitoses.to.1layhid1        -45.845759560339
```

```
Intercept.to.1layhid2     109.549539153463
thickness.to.1layhid2      -6.401008983488
uSize.to.1layhid2           1.204475151458
uShape.to.1layhid2         -6.398936795847
adhesion.to.1layhid2       -3.300467093820
cellSize.to.1layhid2       -2.950643154642
nuclei.to.1layhid2         -9.300364900030
chromatin.to.1layhid2      -6.741438151645
nucleoli.to.1layhid2        1.235986460914
mitoses.to.1layhid2         1.801879560056
Intercept.to.malignancy     3.839754671914
1layhid.1.to.malignancy    -3.542482687423
1layhid.2.to.malignancy -4278.671288127179
```

Výslednou umělou neuronovou síť můžeme graficky znázornit příkazem `plot(nn)`. Výsledek je vidět na obr. 20. Modře jsou naznačeny váhy w_0, které odpovídají *biasu* a v tabulce výsledků *result.matrix* jsou v názvech řádků identifikovatelné slovem *Intercept* ve spojení s názvem příslušné skryté vrstvy a jejího neuronu, tj. např. *Intercept.to.1layhid1* je *bias* pro první neuron první skryté vrstvy. Ostatní váhy jsou u příslušných spojení vyznačeny standardní černou barvou, vstupy a výstup odpovídajícími názvy proměnných z modelu (první parametr funkce `neuralnet` - *formula*).

```
> plot(nn)
```



Obrázek č. 20. Grafický výstup neuronové sítě v R.

Poněvadž balíček *neuralnet* nedisponuje funkcí pro výpočet výstupu vytvořené umělé neuronové sítě na testovacích datech a tuto funkci budu dále potřebovat, nezbylo mi nic jiného, než vytvořit funkci vlastní. Funkce `predikceANN()` má jako argument výstup funkce *neuralnet* a testovací datovou množinu. Princip spočívá v průchodu sítí tak, jak je zaznamenána v matici *result.matrix*. Názvy řádků této matice určují propojení neuronů a vlastní hodnoty jsou váhy jednotlivých vstupů příslušných neuronů. Funkcí `striplit` lze z názvu oddělit počáteční neuron od koncového neuronu z každého spojení. Neuron označený jako *Intercept* je nastaven na hodnotu 1. Posledním zpracovávaným neuronem je výstup sítě, který je současně návratovou hodnotou funkce.

```
predikceANN<-function(nn,testovaci){
 vahy=as.numeric(nn$result.matrix[-(1:3),])
 seznam=strsplit(rownames(nn$result.matrix)[-(1:3)],".to.",fixed=TRUE)
 levy=sapply(seznam, "[", 1);levy=sub("layhid.","layhid",levy,fixed=TRUE)
 pravy=sapply(seznam, "[", 2)
 nazvy=unique(pravy)

pom=numeric(length(nazvy)+1);names(pom)=c("Intercept",nazvy);
pom["Intercept"]=1
 pocet=dim(testovaci)[1]
 vysledek=numeric(pocet)
 nazvy=unique(levy[!grepl("layhid|Intercept",levy)])
 for (i in 1:pocet){
  # vypocet hodnoty - pruchod siti pro jedna data
  pomData=as.numeric(testovaci[i,nazvy]);names(pomData)=nazvy
  pomData=c(pom,pomData)
  for (j in 1:length(pravy)){

pomData[pravy[j]]=ifelse(grepl("layhid",levy[j]),pomData[pravy[j]]+1/
(1+exp(pomData[levy[j]]))*vahy[j],pomData[pravy[j]]+pomData[levy[j]]*vahy[j])
  }
  pomData[pravy[length(pravy)]]=1/(1+exp(-pomData[pravy[length(pravy)]]))
  vysledek[i]=pomData[pravy[length(pravy)]]
 }
 return(vysledek)
}
```

4. 3 Stromové struktury v softwaru R

Pro vytváření klasifikačních a regresních stromů v softwaru R slouží balíček *tree.* Strom je pěstován promocí binárního rekurzivního dělení v závislosti na zvolené odpovědi ve *formuli* a zároveň je vybíráno rozhraní, které rozdělí data na levou a pravou větev. Syntaxe funkce `tree()` se základními argumenty a parametry vypadá následovně

```
tree(formula, data).
```

Popis těchto argumentů a parametrů je uveden v následující tabulce.

parametr	popis
formula	symbolický popis modelu, který má být fitován, ve tvaru formule
data	obsahuje proměnné určené v *formula.*

Tab. 8. Parametry funkce `tree()`.

Výstupní hodnota funkce `tree()` je zase seznam s položkami popsanými v nápovědě k dané funkce. Pro datový soubor `wisconsin2` výtup funkce `tree()` je ve tvaru

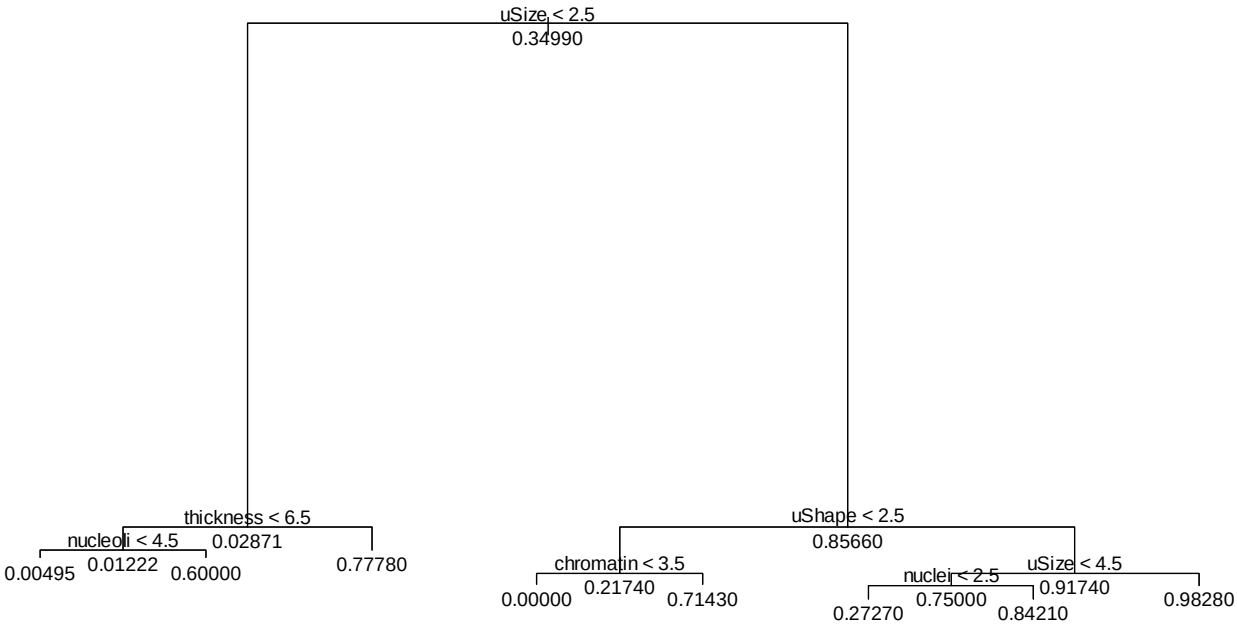
```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 683 155.400 0.34990
   2) uSize < 2.5 418  11.660 0.02871
     4) thickness < 6.5 409   4.939 0.01222
       8) nucleoli < 4.5 404   1.990 0.00495 *
       9) nucleoli > 4.5 5   1.200 0.60000 *
     5) thickness > 6.5 9   1.556 0.77780 *
   3) uSize > 2.5 265  32.550 0.85660
     6) uShape < 2.5 23   3.913 0.21740
      12) chromatin < 3.5 16   0.000 0.00000 *
      13) chromatin > 3.5 7   1.429 0.71430 *
     7) uShape > 2.5 242  18.350 0.91740
      14) uSize < 4.5 68  12.750 0.75000
        28) nuclei < 2.5 11   2.182 0.27270 *
        29) nuclei > 2.5 57   7.579 0.84210 *
      15) uSize > 4.5 174   2.948 0.98280
```

Výstup zobrazuje jednotlivé uzly spolu se svými rozhraními. Symbol * označuje terminální uzly (viz kapitola 2). Je zde uveden počet pozorování, která splňují dané kritérium, odchylka (deviance) a hodnota modelované výstupní proměnné. Výsledek můžeme také zobrazit pomocí funkce `summary()` a vypadá takto

```
Regression tree:
tree(formula = malignancy ~ thickness + uSize + uShape + adhesion +
    cellSize + nuclei + chromatin + nucleoli + mitoses, data = wisconsin2)
Variables actually used in tree construction:
[1] "uSize"     "thickness" "nucleoli"  "uShape"    "chromatin" "nuclei"
Number of terminal nodes:  8
Residual mean deviance:  0.02798 = 18.88 / 675
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.98280 -0.00495 -0.00495  0.00000  0.01724  0.99500
```

Z výstupu lze vidět, které proměnné z datového souboru `wisconsin2` jsou vybrány do stromové struktury. Je zde uveden i počet terminálních uzlů a analýza reziduí.

Výsledkem grafického výstupu (volání funkce `plot()` na objekt vrácený funkcí `tree())` je na obr. č. 21.



Obrázek č. 21. Grafický výstup z funkce `tree()`.

Z grafu vidíme, že nejprve byla vybrána proměnná *uSize*, která rozdělila všechna data do dvou větví (levá větev = odpověď ANO).

Pozn.: Popisky k jednotlivým větvím se musí po volání funkce `plot()` přidat pomocí funkce `text()`aplikované na objekt vrácený funkcí `tree()`.

4.4 Logistická regrese v softwaru R

Model logistické regrese patří do skupiny zobecněných lineárních modelů. Pro výpočet zobecněného lineárního modelu specifikovaného prostřednictvím symbolického zápisu (*formule*) a popisu distribuce chyb (rodina distribučních funkcí) se v softwaru R používá funkce `glm()` se základní syntaxí:

```
glm(formula, family, data, intercept).
```

Popis vybraných parametrů funkce `glm()` je v tab. 8.

parametr	popis
formula	symbolický popis modelu, který má být fitován, ve tvaru formule
data	datová tabulka obsahující proměnné určené v *formula.*
family	popis distribuce chyb a linkové funkce v zobecněném lineárním modelu ve tvaru řetězce nebo volání funkce reprezentující třídu distribucí (viz funkce family()), např.: „binomial“ pro model logistické regrese s linkovou funkcí "logit", „poisson“ pro model Poissonovské regrese s linkovou funkcí "log" apod.
intercept	logická hodnota specifikující zahrnutí absolutního členu do modelu

Tab. 9. Parametry funkce `glm()`.

Výstupní hodnotou funkce `glm()` je opět seznam s položkami popsanými v nápovědě k této funkci. Mezi nejzajímavější patří položka *coefficients*, která obsahuje odhady parametrů modelu jako pojmenovaný numerický vektor. Přehledný výstup procedury lze však získat také prostřednictvím funkce `summary.glm()` aplikované na výsledek funkce `glm()`. Pokud výstup této funkce opět uložíme, můžeme k jednotlivým prvkům výstupu přistupovat jako k položkám seznamu. Položka *coefficients* není jen vektorem odhadů parametrů modelu, ale vrací celou souhrnnou tabulku, ve které je kromě samotných parametrů také směrodatná odchylka odhadů a výsledek testu významnosti každého parametru modelu ve formě P-hodnoty. Pro data z datové tabulky `wisconsin2` a logistický regresní model je volání funkce `glm()` ve tvaru

```
Call:  glm(formula = malignancy ~ thickness + uSize + uShape + adhesion +
    cellSize + nuclei + chromatin + nucleoli + mitoses, family = binomial,
    data = wisconsin2)


Coefficients:
(Intercept)    thickness        uSize       uShape     adhesion     cellSize
  -11.05697      0.58181     -0.05776      0.52133      0.38661      0.18570
     nuclei    chromatin     nucleoli      mitoses
    0.21915      0.61953      0.14997      0.58935

Degrees of Freedom: 682 Total (i.e. Null);  673 Residual
Null Deviance:      884.4
Residual Deviance: 117.9        AIC: 137.9
```

Výstup funkce zobrazený přímo jako hodnota proměnné, ve které je uložen obsahuje informace o odhadech parametrů modelu a celkové zhodnocení modelu. Je zde například uvedena informace o hodnotě Akaikeho informačního kritéria (AIC), pomocí něhož lze

porovnat kvalitu různých modelů mezi sebou navzájem. Čím menší je hodnota tohoto kritéria, tím je model pro daná data vhodnější (Burnham, Anderson, 2002).

Výsledek výpočtu modelu zobrazený funkcí `summary.glm()`, resp. funkcí `summary()` aplikovanou na výstup funkce `glm()` vypadá takto

```
Call:
glm(formula = malignancy ~ thickness + uSize + uShape + adhesion +
    cellSize + nuclei + chromatin + nucleoli + mitoses, family = binomial,
    data = wisconsin2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.6922  -0.1128  -0.0588   0.0206   2.6233

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.05697    1.24336  -8.893  < 2e-16 ***
thickness     0.58181    0.12937   4.497 6.88e-06 ***
uSize        -0.05776    0.20026  -0.288 0.773024
uShape        0.52133    0.21391   2.437 0.014802 *
adhesion      0.38661    0.10949   3.531 0.000414 ***
cellSize      0.18570    0.14787   1.256 0.209179
nuclei        0.21915    0.11224   1.952 0.050887 .
chromatin     0.61953    0.16219   3.820 0.000134 ***
nucleoli      0.14997    0.10936   1.371 0.170251
mitoses       0.58935    0.35733   1.649 0.099087 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 884.35  on 682  degrees of freedom
Residual deviance: 117.91  on 673  degrees of freedom
AIC: 137.91

Number of Fisher Scoring iterations: 8
```

Z tabulky odhadů parametrů je vidět, že mezi významné faktory (na hladině významnosti 5%) ovlivňující hodnotu výstupní proměnné *malignancy* patří (P-hodnota, tj. sloupec Pr(>|z|)): absolutní člen `Intercept`, `thickness`, `uShape`, `adhesion`, `chromatin`. Mezi hraničně významné faktory (statisticky významné na hladině 10%) můžeme ještě zařadit proměnné `nuclei` a `mitoses`. V rychlé orientaci nám pomůžou hvězdičky zobrazované u významných parametrů (viz vysvětlivky pod tabulkou odhadů). Pokud si zobrazíme výstup ve formě `summary(fit.lr)$coefficients` dostaneme tabulku

```
               Estimate   Std. Error      z value
(Intercept) -11.05697396976 1.2433613434 -8.892808216
thickness     0.58181209515 0.1293689215  4.497309620
uSize        -0.05775971961 0.2002612563 -0.288421838
uShape        0.52132968647 0.2139053997  2.437197412
adhesion      0.38661119823 0.1094899993  3.531018363
cellSize      0.18570381113 0.1478739297  1.255825225
nuclei        0.21914675462 0.1122431907  1.952428057
```

```
chromatin     0.61952933886 0.1621904766  3.819763971
nucleoli      0.14997392766 0.1093584551  1.371397643
mitoses       0.58934818946 0.3573326857  1.649298295
                                  Pr(>|z|)
(Intercept) 0.00000000000000000005958360076
thickness   0.00000688187354432073652250 53
uSize       0.77302385336105639002113321 11
uShape      0.01480159625326640457365634 52
adhesion    0.00041396300722126470406428 41
cellSize    0.20917934375496355370316337 03
nuclei      0.05088740472435981238019664 37
chromatin   0.00013357943625889439611884 04
nucleoli    0.17025103171211466990264682 41
mitoses     0.09908653869100149391702103 69
```

ve které *z - hodnota* vyjadřuje hodnotu testové statistiky Waldova testu. Waldův test se používá, pokud chceme testovat významnost jen jednoho parametru β daného regresního modelu. Nulová hypotéza pro tento test má tvar

$$H_0: \beta = 0.$$

Testovací statistika má přibližně normované normální rozdělení N (0,1) a má tvar

$$Z = \frac{\hat{\beta}}{s_{\hat{\beta}}},$$

kde $\hat{\beta}$ je maximálně věrohodný odhad testovacího parametru a $s_{\hat{\beta}}$ je odhad standardní chyby testovacího parametru. Pro tento test je důležité mít dostatečně velký výběr (Hosmer, 2004).

4.5 Srovnání metod

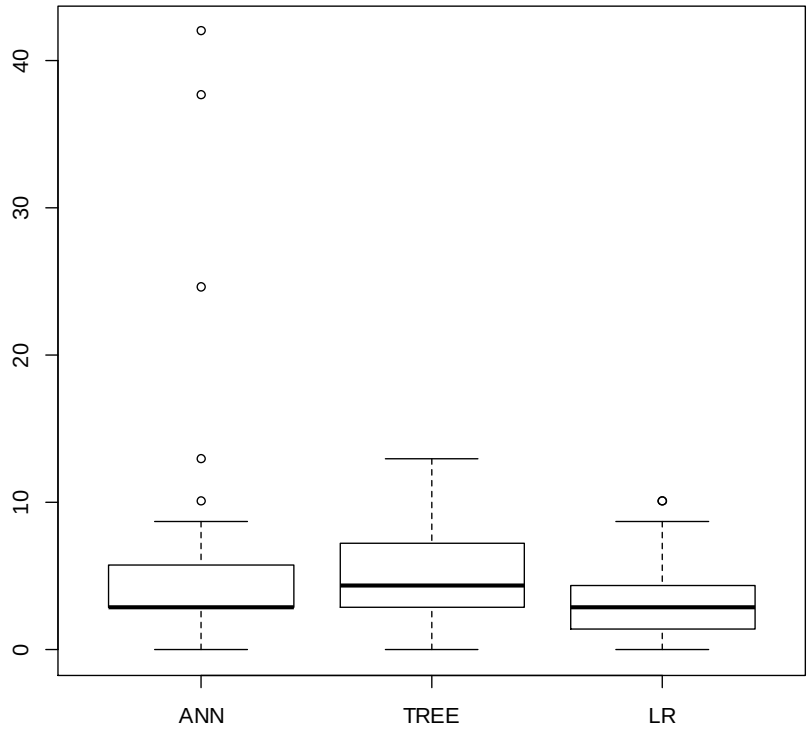
Pro srovnání použitých metod, tj. neuronové sítě, rozhodovacího stromu a modelu logistické regrese použiji stejná data jako pro vzorové příklady, pouze s tím rozdílem, že je postupně 100x rozdělím na data trénovací a data testovací. Trénovací data budou vybrána náhodně (viz vlastní funkce `testKomplet()`) jako 90% původních pozorování. Výstupem každé metody bude pro každé rozdělení dat procento špatně zařazených pozorování. Pro každé rozdělení na testovací a trénovací množinu dat tak vlastně získám trojici hodnot, pro každou metodu výpočtu jednu hodnotu. Výstupem funkce `testKomplet()` je tedy tabulka se 100 řádky a 3 sloupci (proměnnými ANN – neuronové sítě, TREE – rozhodovací stromy, LR – logistická regrese). Současně jsou do adresáře *obrazky* a podsložek ANN a TREE ukládány grafické výstupy (tyto složky musí existovat před spuštěním funkce). Vlastní

tabulku je vhodné uložit do souboru (soubor srovnani.txt na přiloženém CD) pro pozdější analýzy. Výsledek funkce `testKomplet()` je uveden rovněž v příloze 1.

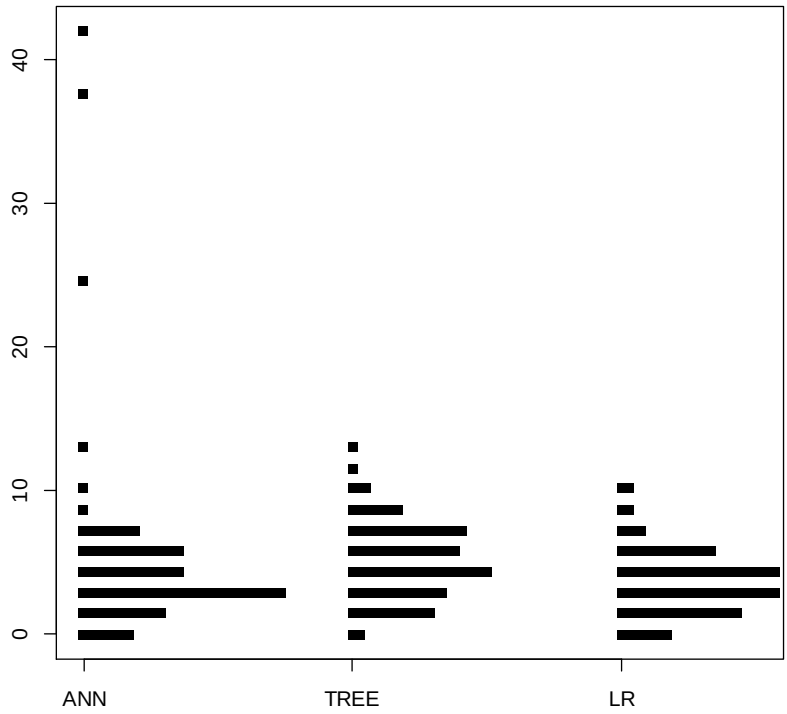
Pro prvotní náhled na data výstupů jednotlivých metod můžeme použít sumarizační charakteristiky (viz tab.č. 8) a jednoduchá grafická zobrazení, např. boxplot (viz. obr. 22) a stripchart (viz. obr. 23)

	ANN	TREE	LR
minimum	0	0	0
1. kvartil	2,899	2,899	1,449
medián	2,899	4,348	2,899
střední hodnota	4,754	5,116	3,623
3. kvartil	5,797	7,246	4,348
maximum	42,03	13,04	10,140
rozptyl	5,938404	2,633585	2,2042

Tab. 10. Sumarizační charakteristiky srovnání metod pro kompletní model.



Obrázek č. 22. Srovnávací krabicový graf pro srovnání metod a kompletní model.



Obrázek č. 23. Graf typu stripchart pro srovnání metod a kompletní model.

V grafu boxplot je okamžitě vidět medián, horní kvantil, dolní kvantil, maximum, minimum a odlehlé hodnoty. Medián je graficky znázorněn jako silná černá čára uvnitř obdélníku. Uvnitř obdélníku se nachází 50% všech dat. Dolní hranici obdélníku tvoří 25. percentil, tzn., že 25% všech dat je nižší než hodnota tohoto dolního kvartilu. Horní hranice představuje 75. percentil, tzn., že 75% všech dat je nižších než hodnota horního kvartilu. Boxplot obsahuje tzv. „vousky“ na jejichž koncích se nachází minimum a maximum. Odlehlé hodnoty jsou vyznačeny kolečky ležící za „vousky“. Čísla vyjadřují procento špatně zařazených hodnot v datovém souboru `wisconsin2`.

Z boxplotu je patrné, že v případě ANN je více odlehlých (extrémních) hodnot než u rozhodovacího stromu a logistické regrese. Též si můžeme všimnout, že v případě ANN medián splývá s dolním kvantilem.

Stripchart je bodový graf, který umožňuje vykreslit hodnoty vedle sebe, tzn., že body jsou na sebe navrstveny jeden na druhý. Výška sloupce bodů vyjadřuje četnost hodnot. Například u ANN bylo nejčastěji špatně zařazeno 4% ze všech dat.

Hodnoty výstupů jednotlivých metod porovnáme prostřednictvím statistických testů. Při testování statistických hypotéz se používají parametrické nebo neparametrické metody. V parametrických metodách musí být splněny určité předpoklady. Jedním z těchto

předpokladů je, že výběr pochází z určitého rozdělení, které je buď určeno úplně, nebo je známé až na nějaké parametry. Například u dvouvýběrového t-testu je předpoklad dvou nezávislých výběrů ze dvou normálních rozdělení, jejichž parametry nemusí být známy (jen rozptyly musí být stejné u obou rozdělení). U neparametrických metod není potřeba znát předpoklad o konkrétním typu rozdělení. Stačí, aby byla splněna obecná podmínka, např. že distribuční funkce rozdělení základního souboru je spojitá. Z parametrických metod jsem použila ANOVU, která je zobecněním Studentova t-testu (testování rovnosti středních hodnot) pro srovnání více jak dvou středních hodnot. Metodu analýzy rozptylu, zkráceně ANOVA, stručně připomenu.

ANOVA

K dispozici je $k > 2$ nezávislých výběrů z normálního rozdělení se stejným rozptylem, který je neznámý, tj.

$$Y_{11}, \dots, Y_{1n_1} \sim N(\mu_1, \sigma^2),$$

$$\dots.$$

$$Y_{k1}, \dots, Y_{kn_k} \sim N(\mu_k, \sigma^2),$$

kde n_i, $i = 1, \dots, k$ je počet pozorování ve výběru. A testuje se hypotéza

$$H_o: \mu_1 = \cdots = \mu_k$$

tzn. rovnost středních hodnot, proti alternativě, že alespoň jedna rovnost neplatí. Jako testovací statistika se používá výraz

$$F_A = \frac{\frac{S_T - S_e}{k-1}}{\frac{S_e}{n-k}} = \frac{S_A}{S_e}\frac{n-k}{k-1},$$

kde $S_e$je součet součtů čtverců odchylek od průměrů v jednotlivých výběrech

$$S_e = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(Y_{ij} - \bar{Y}_{i.}\right)^2,$$

S_T je reziduální součet čtverců

$$S_T = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(Y_{ij} - \bar{Y}_{..}\right)^2,$$

a $S_A$je skupinový součet čtverců (vyjadřuje variabilitu mezi jednotlivými náhodnými výběry)

$$S_A = S_T - S_e = \sum_{j=1}^{n_i} (\bar{Y}_{..} - \bar{Y}_{i.})^2.$$

Testovací statistika F_A za platnosti H_0 má Fisherovo-Snedecorovo rozdělení $F_{k-1,\, n-k}$.
Hypotézu H_0 zamítáme na hladině α, jestliže $F_A \geq F_{k-1,\, n-k}\,(1-\alpha)$.
Pokud zamítneme hypotézu o rovnosti středních hodnot na hladině α, je nutno na této hladině rozhodnout, které výběry se od sebe navzájem liší svými středními hodnotami.
Testuje se hypotéza pro *i*-tý a *j*-tý výběr ($i, j = 1, \ldots, k,\ i \neq j$)

$$H_0^*: \mu_i = \mu_j \quad proti\ H_A: \mu_i \neq \mu_j\,.$$

Požívají se dvě metody mnohonásobného porovnávání *Scheffého metoda* nebo *Tukeyho metoda*. Jelikož mám výběry o stejném rozsahu (100 opakování rozdělení datové množiny), tj. $p = n_1 = n_2 = \cdots = n_k$, používám ve srovnání Tukeyho metodu. Rovnost středních hodnot v tomto případě zamítneme na hladině významnosti α, jestliže

$$|y_{i.} - y_{j.}| > q_{k,n-k}(1-\alpha)\frac{S}{\sqrt{p}}$$

Hodnoty $q_{k,n-k}(1-\alpha)$jsou kvantily tzv. *studentizovaného rozpětí*, které najdeme ve statistických tabulkách nebo je zjistíme prostřednictvím statistického softwaru.

Předpokladem použití analýzy rozptylu je mimo jiné tzv. *homoskedasticita* (shoda rozptylů). Pro ověření předpokladu, že *k* výběrů pochází z normálního rozdělení se stejným rozptylem σ^2tj.

$$H_0: \sigma_1^2 = \cdots = \sigma_k^2,$$

proti alternativě H_A, že alespoň jedna rovnost neplatí, použiji *Bartlettův test*. Součet pozorování ve všech výběrech označme n a platí $n = n_1 + \cdots + n_k$. Zavedeme veličinu s_i^2

$$s_i^2 = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}\left(Y_{ij} - n_i y_{i.}^2\right)^2 \quad pro\ i = 1, \ldots, k$$

tj. výběrový rozptyl pozorování v *i*-tém řádku a veličinu

$$s^2 = \frac{1}{n-k}\sum_{i=1}^{k}(n_i - 1)s_i^2 , C = 1 + \frac{1}{3(k-1)}\left(\sum_{i=1}^{k}\frac{1}{n_i - 1} - \frac{1}{n-k}\right).$$

Pokud je $n_i > 6$ pro všechny *i*, pak testovací statistika má

$$B = \frac{1}{C}\left[(n-k)lns^2 - \sum_{i=1}^{k}(n_i - 1)ln\, s_i^2\right]$$

za platnosti hypotézy o shodnosti rozptylů má přibližně $\chi^2_{k-1}(1-\alpha)$. Pokud $B \geq \chi^2_{k-1}(1-\alpha)$ zamítá se hypotéza o shodnosti rozptylů na hladině přibližně α.

K vlastnímu testování využiji opět software R. Pro výpočet analýzy rozptylu lze využít funkce `aov()`, jejímž nejdůležitějším parametrem je vlastní model zadaný pomocí předpisu (formule), v mém případě `fit<-aov(pom$value~pom$ind)`, kde proměnná `pom` obsahuje výsledek funkce *testKomplet ()*, ve kterém jsou sloupce ANN, TREE a LR shluknuty do jednoho sloupce – proměnné *value*. Proměnná *ind* identifikuje původní název proměnné, ze které vlastní hodnota pochází (ANN, TREE, LR) a slouží tak jako klasifikační proměnná (faktor). Operaci shluknutí sloupců (transformaci původních dat, tj. tabulky srovnani) zajistí funkce `stack()` ve tvaru `pom=stack(srovnani[,2:4])`.

Přehledný výstup analýzy rozptylu zobrazí funkce `summary.aov(fit)`.

```
Df Sum Sq Mean Sq F value  Pr(>F)
pom$ind       2  121.2  60.625  3.8648 0.02203 *
Residuals   297 4658.8  15.686
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Z výstupu vidíme, že na hladině významnosti 10% hypotézu zamítáme.

Analýza rozptylu předpokládá, že výběry (data) pocházejí z normálního rozdělení s ekvivalentními rozptyly. Pro testování normality dat lze využít mnoho statistických testů. Mezi nejpoužívanější patří *Shapirův-Wilkův test*, který je implementován v základním balíčku *stats* softwaru R jako funkce `shapiro.test()`. Výsledné P-hodnoty jsou uvedeny v tabulce 10.

metoda	p-hodnota
ANN	`p-value < 2.2e-16`
TREE	`p-value = 0.004466`
LR	`p-value = 0.0001459`

Tab.11. Výsledek Shapirova-Wilkova testu (P-hodnota).

Jak je vidět výsledné P-hodnoty u všech metod jsou menší než 0,05. Tím pádem můžeme říct, že rozložení dat není ani v jedné metodě normální, tzn., že nepochází z normálního rozdělení. Homoskedasticitu otestujeme pomocí Bartlettova testu, který je rovněž implementován v balíčku *stats* jako funkce `bartlett.test`. V našem případě je výsledná P-hodnota tohoto testu je menší než 0.05 tj. hypotézu o shodě rozptylů na hladině významnosti 5% zamítáme.

Na základě testů předpokladů použití analýzy rozptylu (viz výše) nelze pokládat výsledky testu ANOVA pro srovnání metod rozhodování za relevantní. Místo testu ANOVA proto raději použiji Kruskalův – Wallisův test. Tento test je neparametrickou obdobou jednoduchého třídění analýzou rozptylu. Používá se právě tehdy, nejsou-li splněny předpoklady parametrické analýzy rozptylu. Nechť $Y_{i1}, \dots, Y_{in_i}$ je výběr z nějakého rozdělení se spojitou distribuční funkcí F_i, $i = 1, \dots, k$ a nechť tyto výběry jsou na sobě nezávislé. Testujeme hypotézu

$$H_0: F_1(x) = F_2(x) = \cdots = F_k(x) \qquad \text{pro všechna x,}$$

proti alternativě H_1, že H_0 neplatí. Všechny veličiny Y_{ij} dohromady tvoří *sdružený náhodný výběr* o rozsahu $n = n_1 + \cdots + n_k$.Jedná se o zobecnění Wilcoxonova dvouvýběrového testu pro případ k výběrů ($k \geq 3$). V příkladě s datovým souborem `wisconsin2` mám 3 nezávislé výběry. Všechny tři výběry jsou stejného rozsahu, tzn. $n_1 = n_2 = n_3 = 100$. Všechny veličiny Y_{ij} se musí seřadit do rostoucí posloupnosti a určí se pořadí R_{ij} každé veličiny Y_{ij} ve sdruženém výběru. Musí platit

$$T_1 + T_2 + \cdots + T_k = \frac{n(n+1)}{2},$$

kde T_i, $i = 1, \dots, k$ je součet pořadí prvků patřící do i-tého výběru. Za platnosti hypotézy má veličiny

$$Q = \frac{12}{n(n-1)} \sum_{i=1}^{k} \frac{T_i^2}{n_i} - 3(n+1),$$

při $n_i \to \infty$asymptoticky χ^2 rozdělení o *k-1* stupních volnosti. V případě, že $Q \geq \chi^2_{k-1}(1-\alpha)$ se hypotéza zamítá na hladině významnosti, která je asymptoticky rovna α. Pokud je hypotéza zamítnuta, je třeba určit, které dvojice výběrů se od sebe vzájemně liší. Nejprve si označíme$t_i = T_i/n_i$, pro i= 1, …., k. Nechť $h_{k-1}(\alpha)$ je kritická hodnota Kruskalova-Wallisova testu na hladině α. Pokud jsou výběry malého rozsahu, hodnota $h_{k-1}(\alpha)$ se najde ve speciálních tabulkách, při velkém rozsahu výběru se použije aproximace $h_{k-1}(\alpha) \doteq \chi^2_{k-1}(\alpha)$. Prohlásíme, že distribuční funkce i-tého a j-tého výběru se významně liší pokud, jestliže platí

$$|t_i - t_j| > \sqrt{\frac{1}{12}\left(\frac{1}{n_i} - \frac{1}{n_j}\right) n(n+1) h_{k-1}(\alpha)}.$$

Pravděpodobnost, že alespoň u jedné z $k(k-1)/2$ dvojic distribučních funkcí F_i, F_j bude vypočteno, že se F_i, F_j významně liší, ačkoliv ve skutečnosti platí hypotéza H_0, přitom nepřekročí α.

Zatímco vlastní Kruskalův-Wallisův test je v softwaru R implementován jako funkce `kruskal.test()`, následné násobné porovnání ve standardních balíčcích softwaru R implementováno není, proto jsem si vytvořila vlastní algoritmus odpovídající výpočtu popsanému výše, přičemž používám kritickou hodnotu, resp. kvantil χ^2 rozdělení.

```
# hledani odlisnych dvojic
poradi=rank(pom$values)
tab=table(pom$ind)
T=tapply(poradi,pom$ind,sum);t=T/tab;N=sum(tab)
k=length(tab)
konst=sqrt(1/12*N*(N+1)*qchisq(.95,k-1))
for (i in 1:(k-1)){
 for (j in (i+1):k){
  if (abs(t[i]-t[j])>konst*sqrt(1/tab[i]+1/tab[j]))
print(paste(names(tab)[i],names(tab)[j]))
 }
}
```

Výsledek Kruskalova-Wallisova testu (P-hodnota = <0.001) říká, že distribuce procent špatně zařazených pozorování se pro jednotlivé metody rozhodování statisticky významně liší na stanované hladině významnosti 5%. Následné párové porovnání odhalilo rozdíly mezi dvojicemi: ANN a TREE, LR a TREE.

Zatím byly výsledky porovnány po sloupcích, tj. bez ohledu na konkrétní způsob rozdělení původních dat (`wisconsin2`) na testovací a trénovací datovou množinu. Jednou z metod, kterou bychom mohli využít pro srovnání metod a současně neztratit vazbu na konkrétní testovací data, je použití Hotellingovy statistiky. V dalším textu se pokusím nastínit postup.

HOTELLINGOVA STATISTIKA

Máme k dispozici *n* objektů a na každém z těchto objektů změříme nebo napozorujeme *p* veličin (proměnných, vlastností, znaků). Jednotlivá *p* – rozměrná pozorování jsou nezávislá. Zde budeme označovat náhodný vektor písmenem $x = (X_1, \ldots, X_p)^T$, vektor nenáhodných parametrů μ. matici písmenem $X = (X_{ij})$, $M = (\mu_{ik})$ nebo *Z*. Nejprve musíme definovat *Wishartovo rozdělení*. Nechť jsou dány $x_1 \sim N_p(\mu_1, \Sigma), \ldots, x_n \sim N_p(\mu_n, \Sigma)$ nezávislé náhodné vektory, kde μ_i, $i = 1, \ldots, n$ jsou střední hodnoty a $\Sigma_{(pxp)}$ je varianční matice. Nechť je dána matice $M_{(nxp)}$ tvořena středními hodnotami μ_i, $i = 1, \ldots, n$, náhodných vektorů x_i, $i = 1, \ldots, n$,

$$M = \begin{pmatrix} \mu_1^T \\ \mu_2^T \\ \vdots \\ \mu_n^T \end{pmatrix},$$

kde *n* je počet pozorování a *p* je počet parametrů v modelu. Předpokládáme, že je dána matice $X_{(nxp)} = (X_{ij})_{i=1}^{n}{}_{j=1}^{p}$, kde prvek X_{ij}, představuje *j*-tou proměnnou pozorovanou u *i*-tého objektu, tj.

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix}.$$

Sdružené rozdělení prvků matice $Y_{(pxp)} = X^T X = N \sum_{i=1}^{n} x_i x_i^T$ se nazývá *p – rozměrné Wishartovo rozdělení* o *n* stupních volnosti s parametry Σ a *M*. Značíme $Y \sim W_p(n, \Sigma)$. *Wishartova matice* má tvar $\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T \sim W_p(n-1, \Sigma)$, kde $x = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Definice 4. 1 *Uvažujeme náhodnou matici* $Y \sim W_p(k, \Sigma)$ *a p-složkový vektor* $y \sim N_p(\upsilon, \Sigma)$ *nezávislý na matici Y, kde* $c^{(px1)}$. *Potom*

$$T^2 = kcy^T Y^{-1} y = \frac{ky^T Y^{-1} y}{y^T \Sigma^{-1} y} cy^T \Sigma^{-1} y$$

se nazývá Hotellingova T^2 statistika.

Věta 4.1 *Nechť je n >p -1. Potom*

$$F = \frac{k-p+1}{p}\frac{T^2}{n} \sim F_{p,k-p+1}(\delta)$$

kde je δ parametr necentrality a platí $\delta = cv^T\Sigma^{-1}v$.

TEST HYPOTÉZY O VEKTORU μ PŘI NEZNÁMÉ MATICI Σ

Nechť $x_i = (X_{i1}, X_{i2}, \ldots, X_{ip})^T$, *i* = 1, …, *n* je náhodný výběr, tzn. $x_i \sim N_p(\mu, \Sigma)$.

Dále nechť je dán $\bar{x} \sim N_p(\mu, \frac{1}{n}\Sigma)$ a Wishartova matice $W \sim W_p(n-1, \Sigma)$, je dokázáno, že $\bar{x}$ a W jsou nezávislé. Ověřujeme nulovou hypotézu $H_0: \mu = \mu_0$, proti alternativě $H_0: \mu \neq \mu_0$. Hotellingův test vypadá následovně

$$T_0^2 = n(n-1)(\bar{x}-\mu_0)^T W^{-1}(\bar{x}-\mu_0)$$

$$F = \frac{T_0^2}{n-1}\frac{n-p}{p} = \frac{n-p}{p} n(\bar{x}-\mu_0)^T W^{-1}(\bar{x}-\mu_0) \sim F_{p,n-p}.$$

Pokud platí $F > F_{p,n-p}(\alpha)$ nulovou hypotézu zamítáme. Pro účely srovnání výsledků (procento špatně klasifikovaných pozorování, matice $Z_{(100x3)}$), které poskytují jednotlivé rozhodovací procedury (ANN, TREE, LR) využijeme Hotellingův test. Matice X z definice Hotellingova testu bude představována rozdílem výsledků testování ANN-LR a TREE-LR pro všech 100 opakování rozdělení dat na trénovací a testovací množinu. Předpokládáme, že procento špatně zařazených výsledků bude mít pro všechny metody a všechna opakování stejnou hodnotu (nulová hypotéza), tj. jejich rozdíl bude nula.

Vektor $x_i \sim N(\mu, \Sigma)$, *i* = 1, .., 100.

$x_i = \begin{pmatrix} Z_{i1} - Z_{i2} \\ Z_{i2} - Z_{i3} \end{pmatrix}$ a za platnosti hypotézy $\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, *i* = 1, …, 100 pro Wishartovu matici W platí $\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T \sim W_2(99, \Sigma)$

V softwaru R k vlastnímu výpočtu postačí tři řádky (uvedené dále). Výstupem posledního řádku bude informace o tom, zda nulovou hypotézu o shodě procentuálního podílu špatně zařazených dat jednotlivými rozhodovacími metodami na hladině významnosti 5% zamítáme (TRUE) nebo ne (FALSE).

```
ksi=data.frame(ksi1=srovnani[,2]-srovnani[,4],ksi2=srovnani[,3]-srovnani[,4])
(F=sqrt(99)*(96/2)*t(sapply(ksi,mean))%*%solve(var(ksi)*99)%*%sapply(ksi,mean))
F>=qf(0.95,2,96)
```

V našem případě vyšlo, že hypotézu nelze zamítnout, tj metody při tomto postupu testování poskytují nesignifikantně odlišné výsledky. Tento výsledek se liší od předchozích, je však nutno upozornit na to, že předchozí metody nezohledňují konkrétní rozdělení dat na trénovací a testovací množinu, ale výsledky smíchají pro každou metodu společně jako jednu skupinu.

4.5.1 Model s vybranými faktory

Spolu s výsledky jednotlivých metod rozhodování se pro každé dělení na testovací a trénovací data zaznamenával výsledek logistické regrese s ohledem na statisticky významné faktory modelu. Výstupem je pojmenovaný numerický vektor, který informuje o tom, kolikrát se která proměnná v celkovém počtu 100 modelů projevila v rámci logistického regresního modelu jako statisticky významná na hladině 5%. V mém případě vypadá takto

```
$faktory
(Intercept)   thickness       uSize      uShape    adhesion    cellSize
        100         100           0          87         100           1
     nuclei   chromatin    nucleoli     mitoses
         37         100           0           7
```

Pro další srovnání vyberu proměnné `thickness,uShape,adhesion,chromatin`, které jsou označeny jako významné faktory modelu LR ve více jak 50% případů. Na základě tohoto výsledku provedu 100x nové rozdělení dat na testovací a trénovací datovou množinu a aplikuji na redukovaný model jednotlivé rozhodovací metody (na trénovací množině), které otestuji na odpovídajících testovacích datech. Výsledek opět zaznamenám do souboru *srovnani2.txt* a otestuji podobně jako v případě kompletního modelu. Výsledky uvádím již jen ve stručné podobě spolu s krátkým komentářem.

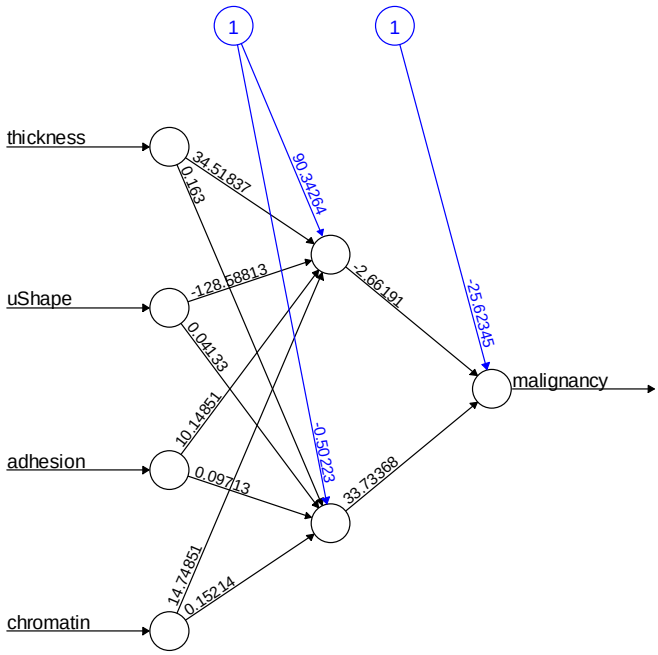
ANN

Zde je výstup z matice *result.matrix.*

```
                                    1
error                   57.05501092061
reached.threshold        0.00810043993
steps                61325.00000000000
Intercept.to.1layhid1   90.34264292470
thickness.to.1layhid1   34.51837259473
```

```
uShape.to.1layhid1       -128.58813175996
adhesion.to.1layhid1       10.14851299267
chromatin.to.1layhid1      14.74851318373
Intercept.to.1layhid2      -0.50222841665
thickness.to.1layhid2       0.16300244169
uShape.to.1layhid2          0.04132561356
adhesion.to.1layhid2        0.09712612718
chromatin.to.1layhid2       0.15214373791
Intercept.to.malignancy   -25.62345213035
1layhid.1.to.malignancy    -2.66190575597
1layhid.2.to.malignancy    33.73368367195
```

Můžeme si všimnout, že počet kroků v jednom opakování algoritmu se zmenšil na 61325. Zmenšila se i hodnota prahu na hodnotu 0,0081. Graf sítě pro redukovaný model vidíme na obrázku č. 24



Obrázek č. 24. Grafický výstup pro redukovaný model ANN z R.

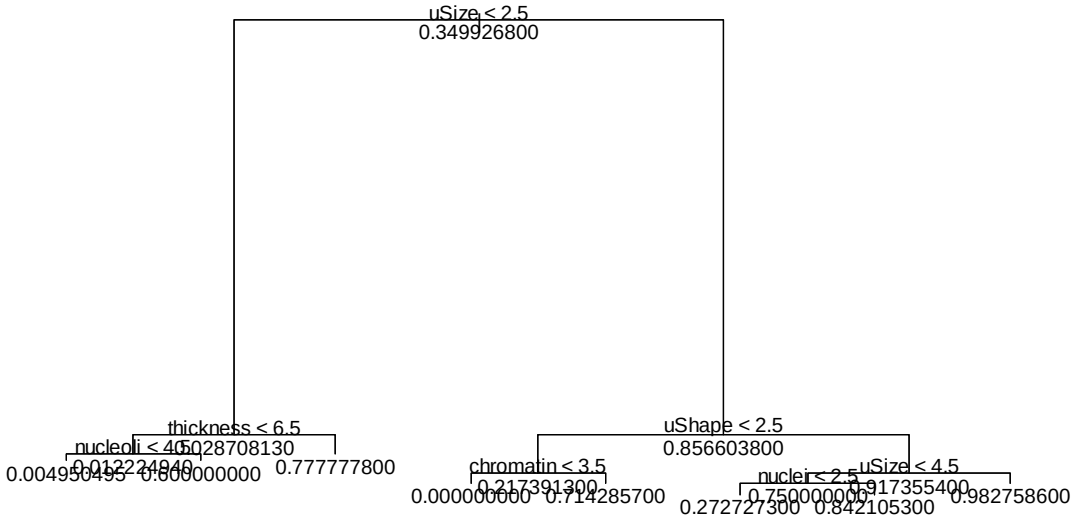
TREE

Výstup z funkce `tree ()` pro redukovaný model je opět zobrazen formou následující tabulky

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 683 155.367500 0.349926800
   2) uSize < 2.5 418  11.655500 0.028708130
     4) thickness < 6.5 409   4.938875 0.012224940
       8) nucleoli < 4.5 404   1.990099 0.004950495 *
       9) nucleoli > 4.5 5   1.200000 0.600000000 *
     5) thickness > 6.5 9   1.555556 0.777777800 *
   3) uSize > 2.5 265  32.550940 0.856603800
     6) uShape < 2.5 23   3.913043 0.217391300
```

```
 12) chromatin < 3.5 16   0.000000 0.000000000 *
 13) chromatin > 3.5 7   1.428571 0.714285700 *
7) uShape > 2.5 242  18.347110 0.917355400
 14) uSize < 4.5 68  12.750000 0.750000000
   28) nuclei < 2.5 11   2.181818 0.272727300 *
   29) nuclei > 2.5 57   7.578947 0.842105300 *
 15) uSize > 4.5 174   2.948276 0.982758600 *
```

Vidíme, že jako první prediktor byl opět vybrán `uSize` a je stejné i rozhraní, podle kterého se data rozdělují do dvou větví.



Obrázek č. 25. Grafický výstup stromu pro redukovaný model.

Opět můžeme použit funkci `summary ()`

```
Regression tree:

tree(formula = malignancy ~ thickness + uShape + adhesion + chromatin,
    data = wisconsin2)
Variables actually used in tree construction:
[1] "uShape"    "chromatin" "thickness"
Number of terminal nodes:  6
Residual mean deviance:  0.03588083 = 24.29132 / 677
Distribution of residuals:
       Min.     1st Qu.      Median        Mean     3rd Qu.        Max.
-0.93212670 -0.01666667 -0.01666667  0.00000000  0.06787330  0.98333330
```

LR

Volání funkce `glm ()` pro redukovaný model je tvaru

```
Call:  glm(formula = malignancy ~ thickness + uShape + adhesion + chromatin,
    family = binomial, data = wisconsin2)

Coefficients:
(Intercept)    thickness       uShape     adhesion    chromatin
-10.0958717    0.6834980    0.7368615    0.4235931    0.7107617

Degrees of Freedom: 682 Total (i.e. Null);  678 Residual
Null Deviance:      884.3502
Residual Deviance: 131.6494     AIC: 141.6494
```

Výsledek modelu zobrazíme opět funkcí `summary.glm ()`

```
Call:
glm(formula = malignancy ~ thickness + uShape + adhesion + chromatin,
    family = binomial, data = wisconsin2)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-3.2915968  -0.1310869  -0.0645040   0.0213954   2.8114468

Coefficients:
              Estimate  Std. Error  z value               Pr(>|z|)
(Intercept) -10.0958717   1.0318332 -9.78440 < 0.000000000000000222 ***
thickness     0.6834980   0.1255980  5.44195         0.000000052701 ***
uShape        0.7368615   0.1448590  5.08675         0.000000364253 ***
adhesion      0.4235931   0.1041845  4.06580         0.000047868039 ***
chromatin     0.7107617   0.1489511  4.77178         0.000001826058 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 884.35019  on 682  degrees of freedom
Residual deviance: 131.64940  on 678  degrees of freedom
AIC: 141.6494

Number of Fisher Scoring iterations: 8
```

Tady vidíme, že vybrané factory jsou všechny významné. Minimum se nám snížilo, ale ostatní charakteristiky (1. a 3. kvantil, median a maximum) se zvýšily. Pro potvrzení významnosti faktorů si zobrazím výstup ve formě `summary(fit.lr)$coefficients`

```
                    Estimate            Std. Error      z value
(Intercept)         -10.0958717305      1.0318332206 -9.784402682
Pr(>|z|)0.000000000000000000000001313686548
thickness           0.6834979525        0.1255980448  5.441947393
Pr(>|z|)0.000000052701240673820175844897 8
uShape              0.7368614925        0.1448590133  5.086749356
Pr(>|z|)0.000000364252708431283261751587 6
adhesion            0.4235931315        0.1041844568  4.065799682
Pr(>|z|)0.000047868039240681846931482756 9
chromatin           0.7107617402        0.1489511025  4.771778983
Pr(>|z|)0.000001826058231068855489814817 3
```

Hodnoty Waldova testu pro všechny faktorory jsou menší než 0,05, tzn. že jsou statisticky významné.

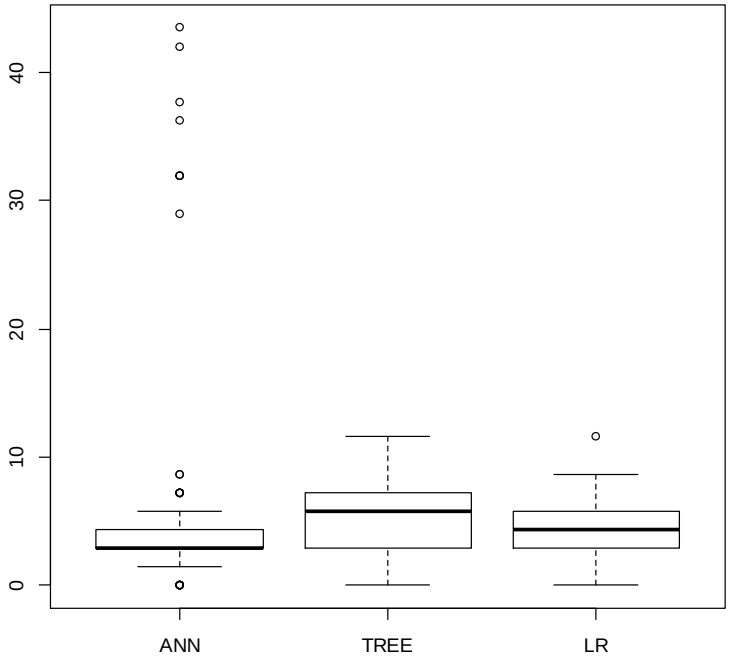
SROVNÁNÍ METOD U REDUKOVANÉHO MODELU

V tabulka 11 jsou uvedeny sumarizační charakteristiky pro redukovaný model.

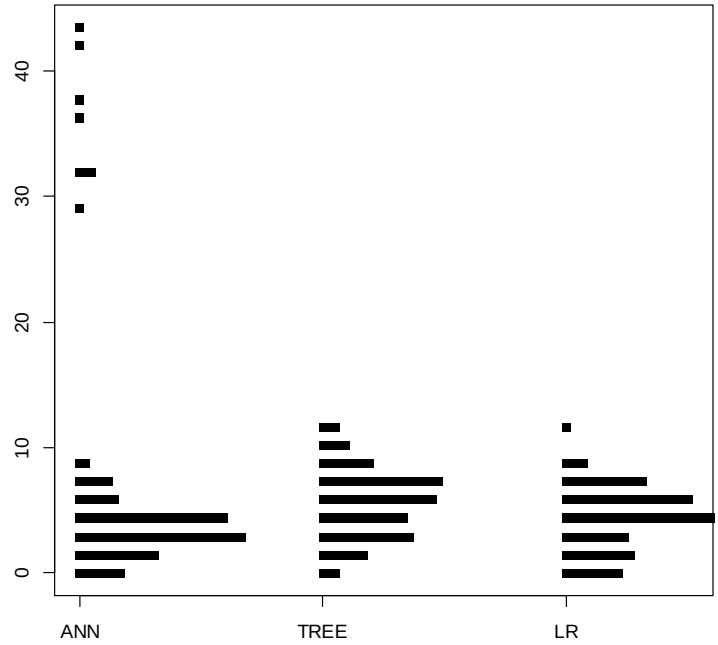
	ANN	TREE	LR
minimum	0	0	0
1. kvartil	2,899	2,899	2,899
medián	2,899	5,797	4,348
střední hodnota	6,028	5,551	4,377
3. kvartil	4,347	7,246	5,797
maximum	43,48	11,59	11,59
rozptyl	9,051	2,71	2,471

Tab. 12. Sumarizační charakteristiky pro redukovaný model.

Ve srovnání s kompletním modelem se všechny charakteristiky změnily až na minimum. U stromové struktury všechny charakteristiky až na minimum a 3. kvantil se zvýšily. U logistické regrese se hodnoty všech charakteristik zvýšily. U umělé neuronové sítě minimum, 1. kvantil a medián se nezměnily, 3. kvantil se snížil a zbývající charakteristiky se zvýšily. Opět jsem si vykreslila boxplot a stripchart.



Obrázek č. 26. Boxplot pro rekukovaný model.



Obrázek č. 27. Stripchart pro redukovaný model.

ANOVA

```
             Df Sum Sq Mean Sq F value Pr(>F)
pom$ind       2    145   72.27   2.273  0.105
Residuals   297   9442   31.79
```

Pokud použijeme parametrický test, dojdeme podobně jako v případě kompletního modelu k zamítnutí hypotézy.

SHAPIRŮV TEST

V tabulce č. 12 jsou výsledky shapirova testu pro redukovaný model.

metoda	p-hodnota
ANN	p-value < 0.00000000000000022
TREE	p-value = 0.0005512
LR	p-value = 0.01185

Tab. 13. Hodnoty Shapirova – Wilkova testu pro redukovaný model.

Opět můžeme říct, že P-hodnota pro všechny modely jsou menší než 0,05, tzn. že i v tomto případě se nejedná ani v jedné metodě o normální rozdělení.

BARTLETTŮV TEST

Hypotézu o rovnosti rozptylů je zamítnuta na hladině významnosti 5%.

```
Bartlett test of homogeneity of variances

data:  pom$value by pom$ind
Bartlett's K-squared = 213.7016, df = 2, p-value < 0.00000000000000022
```

KRUSKAL-WALLISŮV TEST

Tímto testem zjistíme, že procento špatně zařazených pozorování pro jednotlivé metody se statisticky významě liší na hladině významnosti 5%.

```
Kruskal-Wallis rank sum test

data:  srovnani2
Kruskal-Wallis chi-squared = 196.115, df = 3, p-value < 0.00000000000000022
```

Provedla jsem párové porovnání, které odhalilo rozdíl mezi ANN a TREE, LR a TREE. Ke stejnému výsledku jsme došli i v kompletním modelu.

HOTELLINGŮV TEST

Výsledek Hotellingova testu je stejný jako v případě kompletního modelu, tzn. hypotézu nelze zamítnout.

ZÁVĚR

První tři kapitoly byly věnovány třem různým modelům (ANN, stromové struktuře a logistické regresi), které byly následně použity v praktické části. Nejvíce jsem se věnovala 1. kapitole, ve které jsou popsány umělé neuronové sítě.

Poslední čtvrtá kapitola se zabývá zpracováním datového souboru `wisnosin` pomocí zmíněných modelů. Data byla nasbírána za účelem výzkumu rakoviny prsu. Tento soubor obsahuje záznamy od 699 žen, u nichž byly naměřeny hodnoty 10 proměnných. Nejprve jsem datový soubor pozměnila, protože obsahoval chybějící hodnoty. `Wisconsin2` označuje soubor s vynechanými hodnotami. Pro práci jsem využívala software R. Pro práci s umělými neuronovými sítěmi jsem využívala balíček *neuralnet*. Balíček *tree* jsem využila u stromových struktur a funkci `glm` se obecně používá pro práci s logistickou regresí.

Nejprve jsem datový soubor aplikovala na každou metodu zvlášť a podívala se na jednotlivé výstupy. Poté jsme se rozhodla porovnat tyto tři metody a zkoumala jsem, zda se liší.A pokud ano, tak které z nich. Data jsem 100x rozdělila na trénovací a testovací. Výstupem je procento špatně zařazených pozorování. Následně jsem provedla parametrické i neparametrické testy. Z parametrických metod bylo využito ANOVY, která nulovou hypotézu o rovnosti středních hodnot na hladině významnosti 10% zamítla. Prostřednictvím Shapirova testu jsem otestovala normalitu rozdělení dat pro zadané metody. Z toho testu vyplynulo, že ani v jedné metodě data nepocházela z normálního rozdělení. Díky těmto výsledkům jsem použila neparametrický Kruskalův – Wallisův test. Výsledkem testu je, že proceno špatně zařazených pro jednotlivé metody se statisticky významně liší na hladině významnosti 5%. Párové porovnání určilo rozdíly mezi dvojicemi, ANN (umělé neuronové sítě) a TREE (stromové struktury), LR (logistická regrese) a TREE.

Všechny tyto testy porovnávaly výsledky bez ohledu na konkrétní způsob rozdělení původních dat na testovací a trénovací množinu. Použila jsme Hotellingovu statistiku na porovnání všech tří metod. Výsledek této statistiky byl, že daná hypotéza nelze zamítnout, tzn. že stanovené metody poskytují odlišné výsledky.

Prostřednictvím logistické regrese jsem si určila významné parametry datového souboru. Opět jsem porovnávala tři statistické metody, ale z datového souboru jsem vybrala pouze významné faktory. Zopakovala jsem všechny testy, stejně jako u

kompletního modelu. Všechny testy vyšly stejně jako u předchozího modelu, tzv. kompletního testu, v němž byly použity všechny proměnné.

PŘÍLOHA 1

Kompletní test

```
library(neuralnet)
library(tree)
library(stringr)
testKomplet<-function(data,predpis,opakovani){
 n=dim(data)[1]
 vysledek=NULL
 k=1;chyba=0
nazvy=str_trim(strsplit(strsplit(predpis,"~")[[1]][2],"+",fixed=TRUE)[[1]],
side="both");nazvy=c("(Intercept)",nazvy)
 faktory=numeric(length(nazvy));names(faktory)=nazvy
 while (k<=opakovani){
  test=sample.int(n,size=ceiling(.1*n))
  testovaci=data[test,]
  trenovaci=data[-test,]
  radek=NULL
# ANN
  fit.nn<-neuralnet(formula = as.formula(predpis),data=trenovaci,rep=1, hidden=2,
err.fct="ce", linear.output=FALSE)
  result <- try(plot(fit.nn))
  if(class(result) == "try-error") {chyba=chyba+1;next;}
  dev.copy(win.metafile,file=paste("./obrazky/ANN",k,".wmf",sep=""))
  dev.off()
  radek=100*sum(!round(predikceANN(fit.nn,
testovaci))==testovaci[,strsplit(predpis,"~")[[1]][1]])/dim(testovaci)[1]
# TREE
  fit.tree<-tree(formula = as.formula(predpis),data=trenovaci)
  win.metafile(file=paste("./obrazky/TREE",k,".wmf",sep=""))
   plot(fit.tree);text(fit.tree, all = T)
  graphics.off()
  radek=c(radek,100*sum(!round(predict(fit.tree,
testovaci))==testovaci[,strsplit(predpis,"~")[[1]][1]])/dim(testovaci)[1])
# LR
  fit.lr<-glm(formula = as.formula(predpis),data=trenovaci,family=binomial)
  bety=fit.lr$coefficients
  tab=summary(fit.lr)$coefficients
  faktory[rownames(tab)[tab[,4]<0.05]]=faktory[rownames(tab)[tab[,4]<0.05]]+1
#  print(faktory)
  pocet=dim(testovaci)[1]
  vysledekLR=numeric(pocet)
  for (i in 1:pocet){
   pomData=as.numeric(testovaci[i,names(bety)[-1]])
   vysledekLR[i]=1/(1+exp(-bety[1]-t(bety[-1])%*%pomData))
  }

radek=c(radek,100*sum(!round(vysledekLR)==testovaci[,strsplit(predpis,"~")[[1]]
[1]])/dim(testovaci)[1])
# kompletace do tabulky
  vysledek=rbind(vysledek,radek)
  k=k+1
#  print(vysledek)
 }
 colnames(vysledek)=c("ANN","TREE","LR")
 rownames(vysledek)=1:dim(vysledek)[1]
 vratit=list(vysledek=vysledek,faktory=faktory,chyba=chyba)
 return(vratit)
}

qqq=testKomplet(wisconsin2,"malignancy~thickness+uSize+uShape+adhesion+cellSize+
                 nuclei+chromatin+nucleoli+mitoses",100)

qqq
```

výsledek

n	ANN	TREE	LR
1	1.449275362	1.449275362	1.449275362
2	4.347826087	5.797101449	2.898550725
3	5.797101449	7.246376812	5.797101449
4	5.797101449	4.347826087	2.898550725
5	5.797101449	5.797101449	4.347826087
6	5.797101449	7.246376812	4.347826087
7	5.797101449	8.695652174	7.246376812
8	0.000000000	2.898550725	1.449275362
9	13.043478261	7.246376812	5.797101449
10	2.898550725	4.347826087	1.449275362
11	5.797101449	7.246376812	4.347826087
12	1.449275362	4.347826087	0.000000000
13	7.246376812	8.695652174	4.347826087
14	2.898550725	7.246376812	1.449275362
15	4.347826087	5.797101449	4.347826087
16	2.898550725	5.797101449	5.797101449
17	5.797101449	7.246376812	8.695652174
18	0.000000000	4.347826087	0.000000000
19	2.898550725	5.797101449	2.898550725
20	2.898550725	8.695652174	4.347826087
21	1.449275362	8.695652174	2.898550725
22	1.449275362	4.347826087	4.347826087
23	1.449275362	2.898550725	2.898550725
24	2.898550725	1.449275362	1.449275362
25	7.246376812	5.797101449	4.347826087
26	7.246376812	2.898550725	4.347826087
27	4.347826087	7.246376812	7.246376812
28	2.898550725	7.246376812	4.347826087
29	2.898550725	2.898550725	2.898550725
30	1.449275362	7.246376812	0.000000000
31	2.898550725	8.695652174	1.449275362
32	0.000000000	1.449275362	0.000000000
33	2.898550725	2.898550725	1.449275362
34	4.347826087	0.000000000	2.898550725
35	5.797101449	4.347826087	5.797101449
36	2.898550725	5.797101449	2.898550725
37	2.898550725	4.347826087	2.898550725
38	1.449275362	1.449275362	5.797101449
39	0.000000000	4.347826087	2.898550725
40	2.898550725	7.246376812	2.898550725
41	2.898550725	4.347826087	2.898550725
42	7.246376812	5.797101449	7.246376812
43	5.797101449	7.246376812	5.797101449
44	1.449275362	7.246376812	4.347826087
45	4.347826087	2.898550725	4.347826087
46	1.449275362	2.898550725	1.449275362
47	4.347826087	10.144927536	5.797101449
48	2.898550725	4.347826087	1.449275362
49	10.144927536	8.695652174	5.797101449
50	2.898550725	2.898550725	1.449275362
51	0.000000000	1.449275362	0.000000000
52	42.028985507	4.347826087	2.898550725
53	4.347826087	7.246376812	5.797101449
54	4.347826087	2.898550725	2.898550725
55	2.898550725	7.246376812	7.246376812
56	4.347826087	5.797101449	4.347826087

57 4.347826087 7.246376812 2.898550725
58 2.898550725 4.347826087 1.449275362
59 0.000000000 1.449275362 1.449275362
60 0.000000000 2.898550725 0.000000000
61 4.347826087 4.347826087 4.347826087
62 5.797101449 5.797101449 2.898550725
63 5.797101449 5.797101449 2.898550725
64 24.637681159 4.347826087 2.898550725
65 2.898550725 1.449275362 4.347826087
66 2.898550725 0.000000000 1.449275362
67 1.449275362 2.898550725 1.449275362
68 4.347826087 2.898550725 2.898550725
69 2.898550725 5.797101449 5.797101449
70 37.681159420 1.449275362 1.449275362
71 2.898550725 2.898550725 2.898550725
72 5.797101449 7.246376812 4.347826087
73 2.898550725 5.797101449 2.898550725
74 1.449275362 5.797101449 2.898550725
75 1.449275362 1.449275362 2.898550725
76 4.347826087 8.695652174 4.347826087
77 2.898550725 4.347826087 4.347826087
78 2.898550725 4.347826087 2.898550725
79 4.347826087 4.347826087 2.898550725
80 2.898550725 7.246376812 4.347826087
81 7.246376812 10.144927536 10.144927536
82 5.797101449 11.594202899 4.347826087
83 1.449275362 1.449275362 1.449275362
84 2.898550725 2.898550725 1.449275362
85 7.246376812 8.695652174 4.347826087
86 2.898550725 4.347826087 4.347826087
87 0.000000000 1.449275362 0.000000000
88 4.347826087 4.347826087 5.797101449
89 2.898550725 5.797101449 5.797101449
90 8.695652174 7.246376812 10.144927536
91 4.347826087 4.347826087 4.347826087
92 5.797101449 1.449275362 1.449275362
93 2.898550725 1.449275362 0.000000000
94 7.246376812 10.144927536 8.695652174
95 2.898550725 4.347826087 1.449275362
96 7.246376812 5.797101449 5.797101449
97 5.797101449 13.043478261 5.797101449
98 7.246376812 4.347826087 4.347826087
99 2.898550725 2.898550725 4.347826087
100 5.797101449 5.797101449 5.797101449

PŘÍLOHA 2

Výsledek redukovaného modelu.

	ANN	TREE	LR
1	4.347826087	4.347826087	4.347826087
2	5.797101449	10.144927536	5.797101449
3	4.347826087	4.347826087	5.797101449
4	0.000000000	4.347826087	1.449275362
5	1.449275362	1.449275362	0.000000000
6	7.246376812	7.246376812	7.246376812
7	5.797101449	7.246376812	8.695652174
8	31.884057971	2.898550725	1.449275362
9	2.898550725	5.797101449	1.449275362
10	2.898550725	7.246376812	4.347826087
11	2.898550725	5.797101449	5.797101449
12	4.347826087	5.797101449	4.347826087
13	42.028985507	2.898550725	7.246376812
14	1.449275362	10.144927536	0.000000000
15	2.898550725	8.695652174	4.347826087
16	1.449275362	2.898550725	1.449275362
17	4.347826087	7.246376812	5.797101449
18	0.000000000	2.898550725	0.000000000
19	5.797101449	5.797101449	7.246376812
20	4.347826087	7.246376812	7.246376812
21	37.681159420	11.594202899	7.246376812
22	0.000000000	0.000000000	0.000000000
23	7.246376812	8.695652174	7.246376812
24	4.347826087	5.797101449	5.797101449
25	1.449275362	2.898550725	1.449275362
26	4.347826087	7.246376812	5.797101449
27	43.478260870	10.144927536	4.347826087
28	4.347826087	8.695652174	5.797101449
29	4.347826087	4.347826087	4.347826087
30	2.898550725	5.797101449	2.898550725
31	2.898550725	8.695652174	5.797101449
32	4.347826087	4.347826087	5.797101449
33	4.347826087	5.797101449	4.347826087
34	1.449275362	7.246376812	2.898550725
35	2.898550725	2.898550725	0.000000000
36	8.695652174	10.144927536	8.695652174
37	1.449275362	1.449275362	1.449275362
38	2.898550725	4.347826087	4.347826087
39	5.797101449	11.594202899	5.797101449
40	8.695652174	7.246376812	8.695652174
41	0.000000000	8.695652174	5.797101449
42	2.898550725	4.347826087	2.898550725
43	1.449275362	0.000000000	0.000000000
44	4.347826087	4.347826087	4.347826087
45	2.898550725	4.347826087	4.347826087
46	2.898550725	5.797101449	5.797101449
47	4.347826087	10.144927536	7.246376812
48	1.449275362	1.449275362	1.449275362
49	2.898550725	1.449275362	5.797101449
50	4.347826087	2.898550725	2.898550725
51	2.898550725	8.695652174	5.797101449
52	0.000000000	7.246376812	0.000000000
53	4.347826087	7.246376812	7.246376812
54	2.898550725	4.347826087	2.898550725

55	36.231884058	2.898550725	5.797101449
56	4.347826087	7.246376812	5.797101449
57	5.797101449	5.797101449	5.797101449
58	2.898550725	5.797101449	1.449275362
59	2.898550725	2.898550725	4.347826087
60	2.898550725	5.797101449	4.347826087
61	31.884057971	5.797101449	4.347826087
62	0.000000000	0.000000000	0.000000000
63	1.449275362	1.449275362	1.449275362
64	4.347826087	7.246376812	4.347826087
65	4.347826087	2.898550725	4.347826087
66	2.898550725	5.797101449	5.797101449
67	0.000000000	1.449275362	1.449275362
68	2.898550725	7.246376812	4.347826087
69	7.246376812	5.797101449	7.246376812
70	4.347826087	7.246376812	4.347826087
71	1.449275362	2.898550725	2.898550725
72	4.347826087	5.797101449	4.347826087
73	4.347826087	2.898550725	7.246376812
74	31.884057971	4.347826087	4.347826087
75	0.000000000	1.449275362	0.000000000
76	2.898550725	2.898550725	2.898550725
77	2.898550725	2.898550725	5.797101449
78	28.985507246	7.246376812	8.695652174
79	4.347826087	8.695652174	4.347826087
80	2.898550725	7.246376812	2.898550725
81	2.898550725	1.449275362	7.246376812
82	2.898550725	4.347826087	4.347826087
83	4.347826087	7.246376812	4.347826087
84	2.898550725	7.246376812	5.797101449
85	2.898550725	8.695652174	2.898550725
86	5.797101449	8.695652174	4.347826087
87	2.898550725	5.797101449	4.347826087
88	2.898550725	4.347826087	2.898550725
89	1.449275362	4.347826087	0.000000000
90	7.246376812	7.246376812	7.246376812
91	1.449275362	7.246376812	2.898550725
92	1.449275362	2.898550725	1.449275362
93	4.347826087	4.347826087	4.347826087
94	1.449275362	2.898550725	1.449275362
95	2.898550725	5.797101449	4.347826087
96	4.347826087	7.246376812	7.246376812
97	7.246376812	11.594202899	11.594202899
98	7.246376812	5.797101449	7.246376812
99	5.797101449	5.797101449	5.797101449
100	4.347826087	5.797101449	5.797101449

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Literatura

[1] NOVÁK, Mirko, 1998. Umělé neuronové sítě: teorie a aplikace. Praha: C. H. Beck. ISBN 80-7179-132-6.

[2] ŠÍMA, Jiří a NERUDA, Roman, 1996. Teoretické otázky neuronových sítí. Praha: Matfyzpress. Dostupné z: <http://www2.cs.cas.cz/~sima/kniha.pdf>.

[3] BÍLA, Jiří, 1998. Umělá inteligence a neuronové sítě v aplikacích. Praha: Vydavatelství ČVUT. ISBN 80-01-01769-9.

[4] ZELINKA, Ivan, 2005. Umělá intelligence. Univerzita Tomáše Bati ve Zlíně. ISBN 80-7318-277-7.

[5] ZVÁROVÁ, J., SVAČINA Š. a VALENTA Z., 2009. Systém pro podporu lékařského rozhodování. Praha. ISBN 978-80-246-1732-9.

[6] TUČKOVÁ, Jana, 2003. Úvod do teorie a aplikace umělých neuronových sítí. Praha: Vydavatelství ČVUT. ISBN 80-01-02800-3.

[7] OLEJ, Vladimír a HÁJEK, Petr, 2010. Úvod do umělé inteligence. Pardubice. ISBN 978-80-7395-307-2.

[8] WESTON, Trevor, 2003. Atlas lidského těla. ISBN 80-7321-092-4.

[9] PFEIFFER, Jan, 2007. Neurologie v rehabilitaci: pro studium a praxi. Praha: Grada Publishing, a.s. ISBN 978-80-247-1135-5.

[10] DYLEVSKÝ, Ivan, 2000. Somatologie. Olomouc. ISBN 80-86297-05-5.

[11] KOPÁČOVÁ, L., V. SEMECKÝ a M. HRONEK, 1996. Nervový systém. Praha. ISBN 80-7184-292-3.

[12] VONDRÁK, Ivo, 2001. Neuronové sítě. VŠB-TU Ostrava. ISBN 80-7078-259-5.

[13] JELÍNEK, Jan a ZICHÁČEK, Vladimír, 1996. Biologie pro střední školy gymnazijního typu. Olomouc. ISBN 80-86002-01-2.

[14] GODAUX, Émile, 2007. Mozek. ISBN 978-80-7309-389-1.

[15] FOLTÝNEK, Tomáš, 2011. Teorie grafů. Mendelova univerzita v Brně. ISBN 978-80-7375-500-3.

[16] PRINKE, Vladimír, 2003. Mozek jako nástroj. Olomouc. ISBN 80-7346-009-2.

[17] HASTIE, T., TIBSHIRANI, R. a FRIEDMAN, J. H., 2001**.** The Elements of Statistical Learning. Springer. ISBN 978-0387952840.

[18] BOCÁKOVÁ, Milada [online]. Dostupné z: http://www.upol.cz/fileadmin/user_upload/PdF-katedry/KPR/bocakova/Tkane/Nerv/nerv.html

[19] GORLICHOVÁ, Lucie. Umělé neuronové sítě v lékařské diagnostice. Brno, 2006. Diplomová práce. Masarykova univerzita v Brně, Fakulta přírodovědecká, Katedra analytické chemie.

[20] ŠEBKOVÁ, Tereza. Kompoziční data a statistický software R. Olomouc, 2009. Bakalářská práce. Univerzita Palackého v Olomouci, Fakulta přírodovědecká, Katedra matematické analýzy a aplikací matematiky.

[21] KULICH, Michal. [online] 2004. R pro samouky stručný úvod. Dostupné z: http://samba.fsv.cuni.cz/~rysava/doc/uvodr.pdf.

[22] DROZD, Pavel. [online] 2007. Cvičení z biostatistiky – základy práce se softwarem R. Dostupné z: http://cran.r-project.org/doc/contrib/CviceniR1.pdf

[23] HORNIK, K., M. STICHCOMBE a H. WHITE, 1989. Multilayer feedforward network are universal approximators. Technická univerzita ve Vídni. Pp 359-366.

[24] ANASTASIADIS, A., MAGOULAS, G. a VRAHATIS, M., New globally convergent training scheme based on the resilient propagation algorithm. Inflow: Elsevier [online]. 2005. ISSN 253-270.

[25] RAJASEKARAN, S., PAI VIJAYALAKSHMI G. A., 2004. Neural Networks, Fuzzy Logic and Genetic Algorithms. PHI Learning Pvt. Ltd., ISBN 8120321863, 9788120321861.

[26] MARCENO-CEDEŃO A., QUINTANILLA-DOMÍNGUEZ J. a ANDINA D., WBCD breast cancer database classification applying artificial metaplasticity neural network. Inflow: Elsevier [online]. 2011. Svazek 38, vydání 8, ISSN 0957-4174.

[27] FRITSCH, S., GUNTHER F., 2010. Neuralnet training Neural Networks. The R journal vol. 2/1. ISSN 2073 - 4859.

[28] SPREACHER, A., MUELLEMAN R., WADMAN M., A neural network model analysis to identify victims of intimate partner violence. Inflow: Elsevier [online]. 2004. Svazek 22, vydání 2.

[29] KLÍMA David, dostupné z http://www.volny.cz/klimad/skola/spa/skrbek%20web/rbf/rbf.html#rbfneurony.

[30] VÍT David dostupné z http://www.lemonway.com/research/nan-clanek-final.pdf.

[31] KLASCHKA, J., KOTRČ E., 2004. Klasifikační a regresní lesy. Robust 2004.

[32] KOMPRDOVÁ Klára, 2012. Rozhodovací stormy a lesy. Brno. ISBN 978-80-7204-785-7.

[33] MELOUN M., MILITÍK J., 2004. Statistická analýza experimentálních dat. Academia. ISBN 80-200-1254-0.

[34] ZVÁRA Karel, 2008. Regrese. Praha: Matfyzpress. ISBN 978-80-7378-041-8.

[35] MÍŠA, Jakub, 2006. Umělé neuronové sítě. Dostupné z: http://statnice.e-misa.info/C2-01.pdf.

[36] Dostupné z: http://www.brianomeara.info/tutorials/aic.

[37] CIAMPI, A., COUTURIER, A., a SHAOLIN Li, 2002. Prediction trees with soft nodes for binary outcomes. John Wiley. 1145-1165.

[38] AUGET, Jean-Louis, 2007. Advances in Statistical Methods for the Health Sciences: Applications to Cancer and AIDS Studies, Genome Sequence Analysis, and Survival Analysis, *Statistics for Industry and Technology*. Springer. ISBN 97880817645427.

[39] HOSMER, D. W. a LEMESHOW, S., 2004. Applied Logistic Regression. John Wiley & Sons. ISBN 9780471654025.

[40] Český statistický úřad, http://www.czso.cz/.