Česká zemědělská univerzita v Praze

Technická fakulta

Modelování a testování bezpečnostních systémů pomocí neuronových sítí

Diplomová práce

Vedoucí diplomové práce : Ing. Zdeněk Votruba

Autor práce : Bc. Jan Lexa

PRAHA 2013

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra technologických zařízení staveb

Technická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Lexa Jan

Informační a řídící technika v agropotravinářském komplexu

Název práce

Modelování a testováni bezpečnostních systémů pomocí neuronových sítí

Anglický název

Modeling and testing of security systems using neural networks

Cíle práce

Cílem práce je na základě provedené analýzy navrhnout matematický model obvyklého zabezpečovacího systému. Tento model integrovat na model neuronové sítě a otestovat míru pravdivosti provedené analýzy. Navrženou strukturu neuronové sítě implementovat na možnost predikce bezpečnostní zátěže konkrétního objektu.

Metodika

Na základě logického modelu zabezpečovacího systému se pokusit vytvořit matematický model tohoto systému. Definovat základní matematické vztahy mezi jednotlivými komponenty. Na základě matematického modelu navrhnout vhodnou strukturu neuronové sítě a simulovat na této síti reálné chování bezpečnostního systému.

Osnova práce

1. Úvod
2. Definice bezpečnostních systémů PTZS
3. Neuronové sítě - typy a možné použití. nástroje pro simulování neuronových sítí
4. Vytvoření matematického modelu a definování vazeb a jejich závislostí
5. Model neuronové sítě systému PTZS
6. Výsledky a zhodnocení
7. Závěr

Rozsah textové části
50 stran textu včetně obrázků, grafů a tabulek

Klíčová slova
poplachové systémy, zabezpečovací systémy, neuronové sítě, matematické modelování, Matlab

Doporučené zdroje informací

BISHOP C. M.: Neural Networks for Pattern Recognition. Oxford University Press, NewYork, 1995, 498 s., ISBN 0-38-731073-8

DOŇAR, B., ZAPLATÍLEK, K.,: MATLAB - tvorba uživatelských aplikací 2. díl, BEN - technická literatura, 2004, ISBN 80-7300-133-0.

DOŇAR, B., ZAPLATÍLEK, K.,: MATLAB - začínáme se signály 3. díl, BEN - technická literatura, 2006, ISBN 80-7300-200-0.

FAUSETT, L.: Fundamentals of Neural Networks. Prentice Hall, New York, 1994, 404 s., ISBN 0-13-335186-0.

KŘEČEK, S., et al.: Příručka zabezpečovací techniky. 3.vydání, Blatná : Cricetus, 2006. 313 s. ISBN 80-902938-2-4.

LOVEČEK, T., NAGY, P.: Kamerové bezpečnostné systémy. 1.vydání. Žilina: EDIS -vydavateľstvo ŢU, 2008. 283 s. ISBN 978-80-8070-893-1

MAŘÍK V., ŠTĚPÁNKOVÁ O., LAŽANSKÝ J.: Umělá inteligence 4, Academia, Praha, 2003, 476 s., ISBN 80-200-1044-0

NOVÁK, M., et al.: Umělé neuronové sítě – teorie a aplikace, C.H.Beck, 1998, Praha, 382 s., ISBN 80-7179-132-6.

Vedoucí práce
Votruba Zdeněk, Ing.

Termín zadání
listopad 2012

Termín odevzdání
duben 2013

doc. Ing. Miroslav Přikryl, CSc.
Vedoucí katedry

prof. Ing. Vladimír Jurča, CSc.
Děkan fakulty

V Praze dne 29.3.2013

Prohlášení :

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Zdeňka Votruby a použil jen pramenů citovaných v přiložené bibliografii.

V Praze dne ……. Podpis……

Poděkování:

Děkuji vedoucímu diplomové práce Ing. Zdeňku Votrubovi za rady, připomínky a metodické vedení práce.

Abstrakt:

Cílem této diplomové práce bylo ověřit, zda je možné pomocí neuronových sítí vytvořit matematický model zabezpečovacího systému. V první části byla probrána teoretická část zabezpečovacích systémů, základní rozdělení detektorů, jejich funkce a finanční zhodnocení. Další část práce se věnovala teoretickým základům neuronových sítí, jejich rozdělení, vysvětlení principu funkce neuronů a jejich propojení do sítí. Dále byly uvedeny různé typy neuronových sítí, jejich přednosti a použitelnost pro konkrétní případy. V předposlení části práce byl navržen zabezpečovací systém pro fiktivní objekt, aby bylo možné sestavit matematický model a trénovací soubor pro neuronovou síť. Tento model byl v závěrečné části práce aplikován na síť typu vícevrstvý perceptron. Byl vytvořen trénovací soubor, pomocí kterého došlo k natrénování sítě tak, aby se chovala jako matematický model zabezpečovacího systému.

Klíčová slova: poplachové systémy, zabezpečovací systémy, neuronové sítě, matematické modelování, Matlab

Summary:

The objective of this diploma thesis was verification, if it is possible to create mathematical model of alarm system using neural networks. In the first part of this thesis there was explainted theoretical part of alarm systems, basic distribution of detectors, principals of operations and their financial evaluation. The next part of thesis dedicateds to theoretical basis of neural networks, their basic distribution, explainations of principals of function and their connection into networks. In the next part there were stated some types of neural networks, their preferences and usability for particular instances. In the penultimate part of the thesis there was created a model of the alarm system for fictional building to create mathematical model and testing aggregate to neural network. In the last part of the diploma thesis this model was applied to multi-layered perceptron neural network. In the end there was created testing aggragate, to practise the network learned to behave as the mathematical model of the alarm system.

Keywords: alarms, alarm systems, neural networks, mathematical modelling, Matlab

Obsah

1 Úvod

Doba ve které právě žijeme je čím dál tím více spojována s umělou inteligencí. Chytří roboti dokáží naprosto přesně napodobovat manuální činnosti lidí, jsou schopni samostatně řešit dílčí i celé úlohy, lze jim svěřit precizní úkoly, které se pro člověka staly neřešitelnými. To vše díky chytrým čipům, které v sobě skrývají prvky účinně napodobující lidský mozek.

Neuronové sítě jsou jen jedním z mnoha výpočetních modelů, který dokáže věrně repliklovat způsob řešení problému člověkem. Lze pomocí nich řešit problémy, které by byly jinak řešitelné pomocí složitých statistických metod a to s mnohem vyšší přesností. Oblast použitelnosti je velmi rozsáhlá. Od ekonomických účelů, jako je např. vývoj kurzů měn nebo fondů, přes meteorologii, kde se dá s určitou přesností předpovídat počasí, až k běžným účelům jako je např. rozpoznávání znaků textů, či obličejů z fotografií.

Díky univerzálnosti neuronových sítí je lze prakticky využít v podstatě kdekoli, i v odvětví zabezpečení. Pro přesný návrh sítě, která by dokázala určit míru zabezpečení je potřeba nejprve sestavit vhodný model, který by co nejvýstižněji dokázal simulovat činnost zabezpečovacího systému, jeho slabé i silné stránky, míry rizika a způsoby vylepšení. Následně lze vytvořit optimalizovaný model neuronové síě, která se naučí řešit tento problém a s tímto modelem zabezpečovacího systémů dále pracovat.

Aby bylo možno neuronovou síť naučit řešit danou problematiku, je nejprve nutné vytvořit matematický model s předpokádanými vstupy a na základě výpočtů, probíhajících unvitř neuronové sítě se pokusit co nejpřesněji odhadnout, co očekáváme na jejím výstupu.

2 Cíl práce a metodika

Cílem této diplomové práce je na základě provedené analýzy navrhnout matematický model obvyklého zabezpečovacího systému. Tento model integrovat na model neuronové sítě a otestovat míru pravdivosti provedené analýzy. Navrženou strukturu neuronové sítě implementovat na možnost predikce bezpečnostní zátěže konkrétního objektu.

Metodikou práce je na základě logického modelu zabezpečovacího systému se pokusit vytvořit matematický model tohoto systému. Definovat základní matematické vztahy mezi jednotlivými komponenty. Na základě matematického modelu navrhnout vhodnou strukturu neuronové sítě a simulovat na této síti reálné chování bezpečnostního systému.

3 Definice zabezpečovacích systémů PZTS

3.1 Zkratka PZTS

PZTS je zkratkou poplachových zabezpečovacích a tísňových systémů. Podle nové normy ČSN EN 50131-1 z roku 2009 tato zkratka nahradila dlouho užívanou zkratku EZS (elektrické zabezpečovací systémy). I tak však dochází k používání této staré zkratky, protože je již zažitá a léta používaná v praxi [8].

Zkratka EZS byla používána až do roku 2002 pro Elektrickou Zabezpečovací Signalizaci. Následně se změnila na Elektrické Zabezpečovací Systémy [8]. V květnu 2009 přišla další změna a zkratka EZS byla nahrazena termíny PZTS, resp. PZS, resp. PTS. Nové označení vzniklo díky tomu, že nová norma rozlišuje dvě „odvětví“ poplachových systémů:

1) Poplachové systémy pro detekci vniknutí (IAS-Intruder Alarm System, resp. PZS – Poplachový Zabezpečovací Systém)

2) Poplachové systémy pro detekci přepadení (HAS – Hold Alarm System, resp. PTS – Poplachový Tísňový Systém) [9].

3.2 Rozdělení PZTS podle kategorií

Systém PZTS lze rodělit do několika kategorií ochrany. Jsou to kategorie ochrany perimetrické, plášťové, prostorové, předmětové a také ochrany tísňové.

Bezpečnostní systémy lze podle typu provedení rozdělit na bezdrátové, drátové a sběrnicové.

3.2.1 Prvky perimetrické ochrany

Jedná se o prvky, mající za úkol chránit území okolo střeženého objektu. Tento objekt je střežen plotem nebo jinou mechanickou překážkou. Prvky perimetrické ochrany signalizují narušení právě tohoto prostoru dříve, než se pachatel pokusí dostat k samotnému objektu.

Mezi tyto prvky patří závory (bariéry), které pracují na principu infračerveného záření, mikrovlnného záření a nebo jejich kombinace.

3.2.2 Prvky plášťové ochrany

Plášťová ochrana je taková, která by měla zabránit vstupu útočníka do střežené budovy jakýmkoli stavebním otvorem - oknem, dveřmi atd. K detekci takového narušení slouží např. magnetické kontakty. Pevná část (jazýčkový kontakt) je připevněna na rámu okna nebo dveří a pohyblivá část (permanentí magnet) je přimontována k pohyblivé části, tedy přímo na okno resp. dveře. Po otevření dojde k rozpojení magnetického kontaktu, tudíž i k vyhodnocení narušení zóny a tedy i k poplachu.

Dalším prvkem plášťové ochrany může být i detekce rozbití skla. Při nárazu do skla a jeho tříštění se dočkáme změny tlaku a poté zvuku tříštění, který má své charakteristické vlastnosti. Dnešní kombinované detektory tříštění skla takto dokáží vyhodnotit pokus o vnik bez otevření rámu, okna či dveří.

3.2.3 Prvky prostorové ochrany

Prostorová ochrana je taková, která má za úkol detekovat pohyb pachatele vloupání již po překonání perimetrické a plášťové ochrany do střežených prostor. Hlavním úkolem je detekovat pohyb po hlídaných místnostech. Nejčastěji se k tomuto účelu používají pasivní infračervené detektory (zkráceně PIR), které snímají teplo pachatele a jeho pohyb mezi tzv. Fresnelovými zónami. Tyto zóny se dají upravit pomocí Fresnelových čoček a tím upravit parametry hlídaného prostoru. Tyto PIR čidla se vyrábí i v aktivní podobě a to v ultrazvukové, nebo mikrovlnné. Principy vyhodnocení jsou odlišné. U mikrovlnných detektorů se jedná o princip změny mikrovlnného záření po dopadu na předmět a u ultrazvukového detektoru se jedná o Dopplerův jev.

Při užití aktivních detektorů je vhodné nejprve zvážit vhodnost použití pro konkrétní situace.

3.2.4 Prvky předmětové ochrany

Tyto prvky slouží k přímé ochraně předmětů zvláštní hodnoty, které chceme chránit před odcizením. V této kategorii rozeznáváme čidla kontaktní, kapacitní, tlaková, akustická, trezorová a čidla na ochranu uměleckých děl [3].

Pod pojmem kontaktní čidlo si lze představit senzor, který je se střeženým předmětem pevně spojen. Při sebemenší manipulaci s tímto předmětem poté dojde ke spojení (rozepnutí) kontaktu a tudíž i k vyhlášení poplachu. Jako příklad lze uvést různé typy mikrospínačů, tahová nebo tlaková čidla.

Kapacitní čidla jsou taková, která svým principem pracují na stejné bázi jako kondenzátory. Poplach způsobí změna elektrostatického pole vytvořeného mezi předmětem a zemí, která je před manipulací jasně nastavena.

Akustická čidla se vyrábí většinou v kombinaci s čidli tlakovými a tím dochází k selekci planých poplachů (např. vlivem rozbití skleněného předmětu bez přičinění nežádoucí osoby) od poplachů skutečných. Poplach zde vyvolá současné hlášení tlakové i akustické složky čidla. Tyto čidla lze použít tam, kde máme chráněné předměty umístěné za skleněnými plochami (vitríny, tabulová skla apod.)

Podobně jako u perimetrické ochrany lze i u ochrany předmětů použít různé druhy barier, infračidel, infrabarier nebo podobných optických čidel.

Poslední kategorie spadající do předmětové ochrany jsou speciální čidla „šitá na míru“ každému konkrétnímu předmětů. U vzácných obrazů se může např. jednat o čidla závěsná, která snadno dokáží rozeznat manipulaci se zavěšenými předměty. U sošek nebo jiných vzácných stojících předmětů lze použít např. čidlo váhové.

3.2.5 Prvky tísňové ochrany

Jedná se o hlásiče, pomocí kterých chceme ochránit osoby v případě nebezpečí. Tísňové hlásiče známe skryté, veřejné a osobní. Veřejné hlásiče musí být dostupné každému, aby bylo možno v případě nebezpečí urychleně vyhlásit poplach. Jako příklad lze uvést

tlačítka nouze v prostředcích městské hromadné dopravy (metro) kde jsou pomocí těchto signálů okamžitě alarmovány složky mající odpovědnost za bezpečnost provozu. Tímto způsobem lze upozornit např. na požár, úraz, náhlou nevolnost starších osob, popř. napadení třetí osobou.

Skryté hlásiče jsou používány tam, kde dochází k tokům peněžních prostředků. Jsou důležité pro alarmování bezpečnostní agentury, popř. rovnou policie ČR v případě napadení, případně pokusu o loupež finanční hotovosti a to takovým způsobem, aby nebylo možné odhalit osobu, která alarm spustila.

Osobní tísňové hlásiče jsou ideálním prostředkem pro přivolání okamžité pomoci. Mají podobu klíčenek nebo hodinek, které mají chráněné osoby neustále při sobě. Největší využití našly zejména u starších osob, které si mohou takto přivolat lékařskou pomoc v případě náhlé nevolnosti a předejít tak vážným zdravotním komplikacím.

3.2.6 Možnosti komunikace PZTS

Všechny vyvolané akce (ať už se jedná o prvky kterékoli z výše uvedených kategorií prvků) jsou vyhodnoceny ústřednou PZTS a ta na základě nastavení systému vyhlásí poplach. Poplach se projeví spuštěním sirén (ať se jedná o sirény venkovní, či vnitřní) a tento stav přepošle dále.

První možností je kontaktovat majitele objektu, kde je nainstalovaný zabezpečovací systém. Lze tak učinit pomocí přídavného GSM komunikátoru, který je schopen poslat tísňovou SMS nebo rovnou zatelefonovat na předem nastavené telefonní číslo a upozornit tak majitele na způsob vniknutí, umístění narušené zóny, či jen pokus o sabotáž systému.

Druhou možností je připojení zabezpečovacího systému na pult centrální ochrany. Po vyhlášení poplachu, již firma provozující PCO sama podnike potřebné kroky, např. kontaktuje hlídací agenturu nebo rovnou policii, přivolá pomoc zravotnické služby nebo hasičského sboru v případě požáru.

3.2.7 Ekonomické zhodnocení zabezpečovacích systémů

Vzhledem k tomu, že žádný zabezpečovací systém není stoprocentně bezpečný a jedná se spíše o prevenci, jež má vetřelce v první řadě odradit od pokusu o vloupání, je potřeba zvážit ekonomickou ivestici do onoho zabezpečovacího systému. Zejména je důležité stanovit si, zda má cenu zběsile rozvěsit co nejvíce čidel bez účelu jen pro ten pocit, že se objekt stane „nedobytným“ a nebo zda se pokusit zaměřit pouze na možná místa vniku popř. na místa, kde se nachází kritické zóny, ve kterých je uložen cenný majetek.

Pro ideální investici je vždy nutné stanovit si vyvážený počet a typ detektorů podle účelu. Tím, že budeme nad rozmístěním detektorů více přemýšlet, lze ušetřit nejen na počtu detektorů, ale rovněž i na typu ústředny. Při rozumné redukci např. nebudeme potřebovat vysoké počty zón v ústředně nebo zónové expandéry.

Při návrhu zabezpečovacího systému je velmi důležité, pokusit se uvědomit si, jak by pachatel postupoval v případě pokusu o vloupání (zaměřit se na místa, která jsou nejvíce ohrožená). Může se jednat např. o skleněné tabule oken a dveří, dosažitelné ze země, nebo o případá místa vniku špatně viditelná z okolí.

V takovýchto případech je důležité, zaměřit se na vstupní otvory do objektu a pokusit se je zabezpečit pokud možno co nejlépe. Jedná se například o magnetické kontakty na rámech dveří či oken, popř. tříštivé detektory, rozeznávající rozbití skleněných tabulí. Pro případ selhání nebo sabotáže těchto základních čidel je možné pojistit se prostorovou ochranou pomocí PIR detektorů. Už takto zabezpečený objekt je poměrně obtížně napadnutelný a pokud se elektrický zabezpečovací systém doplní o vhodné mechanické zabezpečení, získáme tím kvalitní stupeň zabezpečení za poměrně nízkou cenu. U většiny standartních rodinných domků pak stačí zabezpečení sladit s požadavky konečného uživatele pomocí tísňového nebo odchodového tlačítka, či GSM komunikátoru. Nejedná se však o nezbytnosti, nýbrž o navýšení komfortu používání a samozřejmě se to projeví i v konečném rozpočtu.

Celková cena zabezpečovacího systému se odvíjí od požadovaného stupně zabezpečení (Podle ČSN EN 50131-1 rozlišujeme 4 stupně zabezpečení), typu vybrané ústředny, počtu a druhu detektorů a také množství doplňků pro snažší ovládání systému.

Co se týče ceny, nejnižšími položkami na celkovém rozpočtu jsou čidla typu magnetických kontaktů, které se pohybují (v závislosti na stupni zabezpečení) v řádu desítek až stovek korun. Samozřejmě existují i dražší bezdrátové verze, které se mohou vyšplhat k cenám v řádech tisíců korun v závislosti na stupni zabezpečení. O něco málo dražší jsou PIR detektory. Jejich cena se odvíjí jednak rovněž od stupně zabezpečení, ale také na druhu. Vzhledem k tomu, že PIR detektory si můžeme pořídit v provedení aktivním (mikrovlnné a ultrazvukové) nebo pasivním, je velmi důležité zhodnotit vhodnost použití daného typu pro hlídání za konkrétních podmínek. Cena PIR detektorů se může pohybovat v řádu stovek (levné pasivní drátové detektory), až tisíců korun (bezdrátová řešení s vysokým stupněm zabezpečení). Nejvyšší položkou celkového ropočtu bývá ústředna společně se základním vybavením, které je nezbytné pro správný chod zabezpečovacího systému. Jedná se např. o klávesnice, napájecí transformátor, box na ústřednu, nebo také akumulátor, který musí být navržen tak, aby jeho kapacita splňovala požadavky pro danou bezpečnostní třídu. Cena ústředny se podle provedení může pohybovat v řádu jednotek až desítek tisíců korun. V rozpočtu také nesmíme opomenout spojovací a pomocný materiál, kabeláž a další možné doplňkové moduly.

Kvalitní zabezpečovací systém pro běžný rodinný domek lze pořídit řádově do 20 000 Kč. cena se samozřejmě může navýšit v případě, že požadujeme vyšší komfort při používání (IP modul, GSM modul) a nebo pokud chceme tento systém doplnit o kamerový systém.

Dnes jsou na trhu známy i kompletní domovní systémy, které v sobě obsahují zabezpečovací systém, požární ochranu, systémy docházky a vstupů nebo také prvky domácí automatizace. Tyto „inteligentní“ systémy jsou v současnosti ještě neustále ve vývoji a jejich cena je znatelně vyšší.

4 Neuronové sítě

Aby bylo možné aplikovat zabezpečovací systémy (dále jen PZTS) na model neuronové sítě, je nejprve nutno se důkladně seznámit se samotnými neuronovými sítěmi, jejich různými typy a zhodnotit jejich výhody a použitelnost v daném případě.

Nejprve je nutné seznámit se s tímto pojmem, který se v souvislosti s informatikou poprvé objevil zhruba v polovině 20. století.

Podstata umělých neuronových sítí je založena na racionálním napodobení struktury a principů činnosti biologických neuronových sítí (tvořících základ informačních systémů všech živých organismů) pomocí umělých technických nebo programových prostředků [12].

Neuronové sítě jsou složeny z jednotlivých neuronů, které jsou vzájemně pospojovány pomocí synaptických spojů. Existuje mnoho typů neuronových sítí lišících se strukturou, matematickým popisem, přechodovými funkcemi, způsobem učení a v neposlední řadě i účelem, pro který byly navženy. Známe sítě, které jsou vhodné pro regresní úlohy, kde požadujeme tzv. učení s učitelem abychom dosáhli u každého nového příkladu předem požadovaný výsledek, na druhou stranu existují i sítě, které se samostatně rozhodují a pomocí kterých lze řešit problémy, u kterých si sami nejsme jisti výsledkem řešení. Nejpoužívanější neuronové sítě budou detailně rozebrány v následující kapitole, počínaje samotným neuronem, přes způsoby učení až k účelu využití v konkrétních případech.

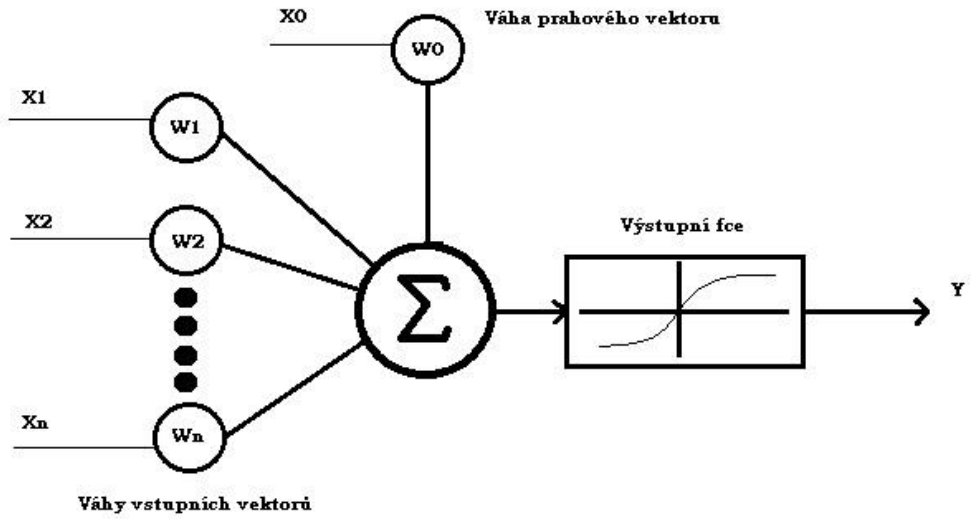
4.1 Neuron

Fyziologický neuron je základní jednotkou nervové soustavy. Nervová soustava člověka (obecně živých organismů) zprostředkuje vztahy mezi vnějším prostředím a organismem, a zajišťuje tak reakci na vnější podněty i na vnitřní stavy organismu [11].

Pro účely výpočtů pomocí neuronových sítí bylo potřeba nejprve vytvořit matematický model neuronu, který by nejlépe simuloval chování neuronu fyziologického pomocí matematických funkcí.

4.1.1 Matematický model neuronu

Obrázek. 1 Formální neuron



[*Zdroj: upraveno dle (10)*]

Na obr.1 je znázorněna matematická interpretace fyziologického modelu neuronu. Na vstup neuronu je přivedeno obecně n vstupů x1…xn s váhami w1…wn. Prahový vektor x0 má definovanou váhu w0 jako tzv. prahové napětí θ. Suma součinů vstupů a jejich vah je aplikována na výstupní funkci, která bývá nelineární. Následně je výsledek porovnáván s prahovým napětím. V případě, že vypočtená hodnota překročí prahové napětí, dochází následně ke změně vah na vstupech a výpočet se opakuje. V případě, že je výpočet nižší než prahové napětí, konfigurace neuronu se nemění.

Matematicky lze funkci neuronu popsat následovně:

Matematický popis neuronu

$$\boldsymbol{y = F\left(\sum_{i=1}^{n} x_i\, w_i + \theta\right)} \qquad \mathbf{(1)}$$

kde:
x_i - je hodnota na i-tém vstupu
w_i - je váha i-tého vstupu
θ - je prahová hodnota
n - je celkový počet vstupů

F - je obecná nelineární funkce
y - je hodnota výstupu [10].

4.1.2 Přechodové funkce

Přechodové funkce se liší podle typu použitých neuronů a volíme je tak, aby byly neurony schopné interpretovat výsledky co možná nejblíže našemu očekávání. Setkáváme se např. s těmito přechodovými (aktivačními) funkcemi.:

Ostrá nelinearita (skoková funkce)

Jedná se o funkci, mající ostrý průběh. Hodnota aktivační fce $\delta(x)= 1$, v případě, že $x \geq 1$. Pokud je $x = 0$, pak $\delta(x)= 0$. Tato funkce je charakteristická pro binární neuron a je nespojitá

Logistická singmoida

Pro tuto funcki známe první a druhou derivaci. Jedná se o funcki spojitou a rostoucí a je interpretována vztahem:

Logistická sigmoida

$$y(x) = \frac{1}{1 + e^{-kx}} \tag{2}$$

Hyperbolický tangens

Podobně jako logistická sigmoida i hyperbolický tangens je rostoucí, spojitá funkce. Rozdílem oproti logistické sigmoidě je, že v nule platí $y(x) = 0$. Pro hyperbolický tangens máme vztah:

Hyperbolický tangens

$$y(x) = \frac{1 - e^{-kx}}{1 + e^{-kx}} \tag{3}$$

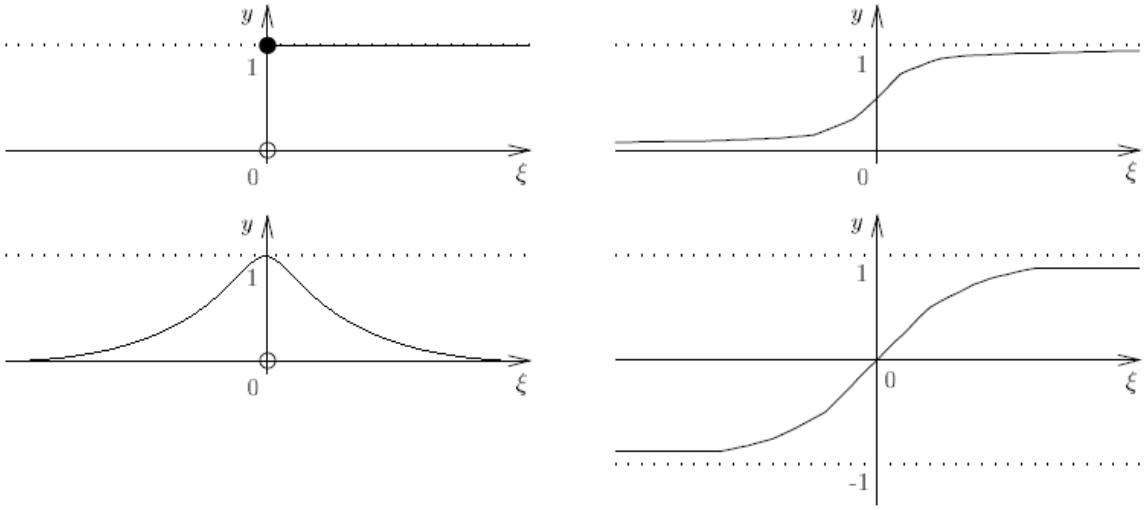
Funkce radiální báze

Tato funkce je specifická tím, že její hodnoty se v $-\infty$ i v ∞ rovnají 0. A v nule se hodnota funkce radiální báze $y(x) = 1$. Pro tuto funkci máme vztah:

Fce radiální báze

$$y(x) = e^{-kx} \tag{4}$$

Obrázek 2 Přechodové fce (Skoková fce, logistická sigmoida, fce radiální báze,hyperbolický tangens)



[Zdroj: upraveno dle (11)]

4.2 Typy neuronových sítí

V současnosti známe již několik typů neuronových sítí. Neuronové sítě se liší způsobem výpočtu výsledku, svou vnitřní strukturou nebo i počtem a typem použitých neuronů. Mohou se lišit také způsobem učení. Typově lze neuronové sítě rozdělit na: jednoduchý perceptron, vícevrstvý perceptron, Hopfieldovu síť, kompetiční neuronové sítě (Kohonenova síť) a další.

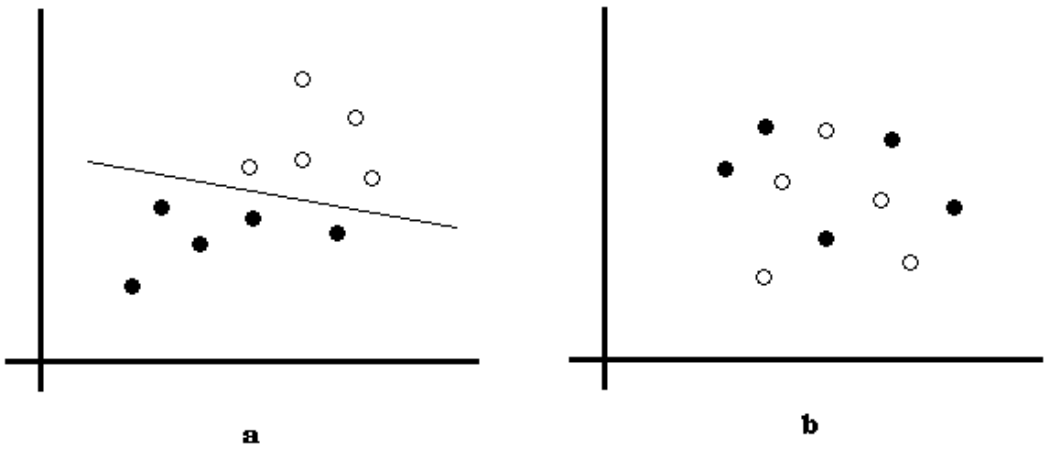
4.2.1 Jednovrstvý perceptron

Jedná se o nejjednodušší neuronovou síť, schopnou řešit ty nejjednodušší operace. Jedná se zejména o řešení problémů, které jsou lineárně separabilní.

Lineární separabilita

Dvě kategorie X^+ a X^- se nazývají lineárně separabilní (obr.3a), pokud je lze oddělit nadrovinou. Pokud jsou X^+ a X^- lineárně separabilní, existuje neuron, který je klasifikuje s nulovou chybou. V opačném případě (X^+ a X^- nejsou lineárně separabilní (obr. 3b)) takový lineární neuron neexistuje [6].

Obrázek 3 Příklad lineární separability



[Zdroj: vlastní]

Pomocí jednovrstvého perceptronu lze realizovat různé (lineárně separabilní) booleovské funkce jako jsou např. OR nebo AND. záleží přitom na nastavení vstupních vah a na přechodové funkci. Např. funkce OR lze dosáhnout vstupním nastaveních vah, prahového napětí a výběru aktivační funkce podle následující tabulky:

Tabulka 1 Funkce OR pomocí perceptronu

Váhový vektor w1	0.5
Váhový vektor w2	0.5
Vektor prahového napětí w0	<0;0.5)
Zvolená aktivační funkce	Sign(x)

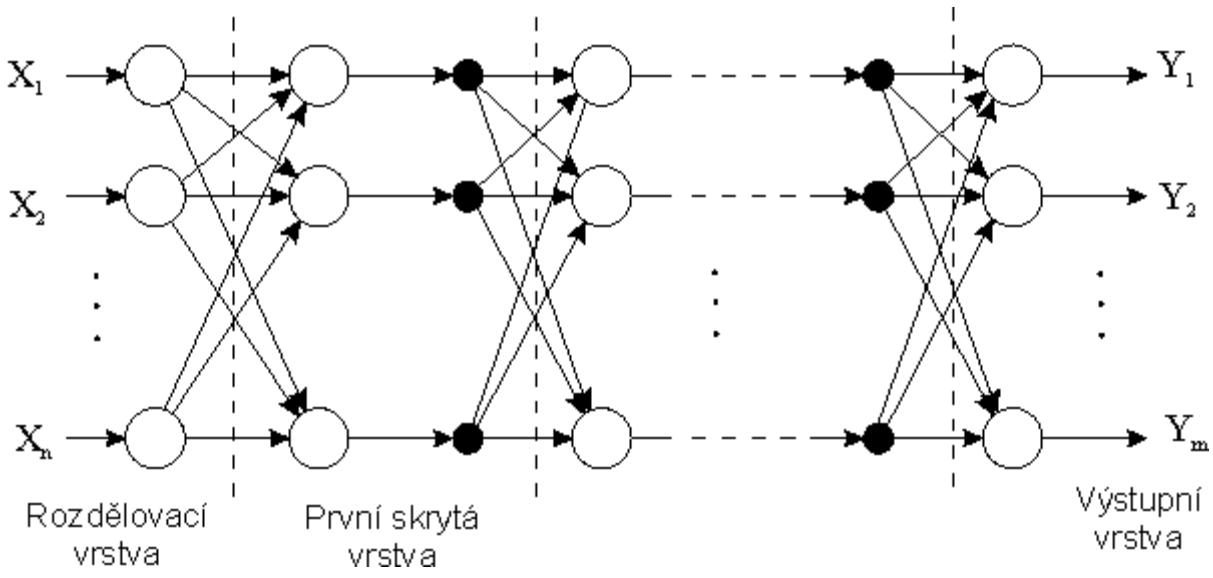
[Zdroj:vlastní]

Pro realizaci funkce AND stačí pozměnit váhu vektoru w0 na libovolnou hodnotu z intervalu <0.5;1> .Tím dojde k posunutí prahu a funkce bude na výstupu y(x) rovna jedné pouze v případě, že na oba vstupy x1 a x2 budou přivedeny hodnoty odpovídající logické 1 .

4.2.2 Vícevrstvý perceptron

Tento typ sítě je složen z vrstev jednoduchých perceptronů. Každá vrstva má svou vlastní přechodovou funcki a výstupy z neuronů jsou vzájemně propojeny se vstupy všech neuronů obsažených ve vrstvě následující. První vrstva perceptronů se nazývá vstupní a perceptrony v ní obsažené se nazývají receptory. Zvláštností receptorů je to, že mají pouze jeden vstup. Poslední vrstva neuronů se nazývá vrstva výstupní. Všechny další vrstvy, které jsou obsažené mezi vrstvami vstupními a výstupními se nazývají skryté. Počet neuronů v jednotlivých vrstvách se volí v závislosti na zadání řešeného problému a celá síť může mít libovolné množství vrstev. Celkové uspořádání sítě se pak nazývá topologie sítě. Všechny možné konfigurace vytvářejí tzv. váhový prostor neuronové sítě [11].

Obrázek 4 Vícevrstvá perceptronová síť



[Zdroj: (10)]

Pomocí vícevrstvé perceptronové sítě lze řešit již složitější problémy a funkce, které nejsou lineárně separabilní (např. fce XOR). Topologie (architektura sítě) je způsob uspořádání a propojení neuronů. Pod pojmem konfigurace sítě nalezneme topologii sítě

společně s přiřazením vah. Výpočtem (nebo také relaxací) sítě rozumíme stanovení hodnot na výstupních neuronech na základě zadání hodnot vstupních [6].

Učení vícevrstvého perceptronu

Existuje mnoho algoritmů, podle kterých se provádí výpočet sítě. V první řadě lze učení rozdělit na učení s učitelem a učení bez učitele.

Během tzv. učení s učitelem se využívá algoritmu zpětného šíření chyby (backpropagation of error). Celé síti je předkládán testovací soubor, který obsahuje vstupní a předem definované výstupní hodnoty. Po každé změně vah se kontroluje, s jakou chybou síť výpočet provedla a následně se výpočty opakují, dokud se nesplní ukončovací podmínka. Tou může být např. předložení max. počtu trénovacích prvků, nebo dosažení max. dovolené chyby. Tento algoritmus učení je nejpoužívanější díky vysoké přesnosti. Nevýhodou je však to, že změny vah probíhají v malých rozsazích, proto přesné výpočty trvají značně déle, než u algoritmu učení bez učitele.

Učení bez učitele se používá u dopředných vícevrstvých sítí. Neuronové síti jsou postupně předkládány prvky trénovacího souboru (podobně jako u učení s učitelem) a po každém dalším prvku se změní váhy jednotlivých neuronů. U tohoto postupu se nekontroluje výstup a algoritmus končí po předložení všech prvků trénovacího souboru.

Použití vícevrstvého perceptronu

Vícevrstvý perceptron je nejuniverálnější neuronovou sítí. Lze ji použít pro řešení libovolné regresní úlohy. Jako příklad lze třeba uvést úspěšné aplikování této sítě v oblasti komprese dat, prognózování vývoje kurzu nebo postupu počasí.

4.2.3 Hopfieldova síť

Na rozdíl od vícevrstvého perceptronu, v Hopfieldově síti hraje důležitou roli i čas. Časový úsek musí být obsažen i v popisu neuronu. Proto se postsynaptický potenciál u Hopfieldovy sítě vypočítá podle následujícího vztahu:

Postsynaptický potenciál Hopfieldovy sítě

$$h(t) = \sum x_i(t) w_i \qquad (5)$$

Topologie sítě je zvláštní tím, že všechny neurony jsou zároveň vstupní i výstupní. Výstupy každého neurony jsou propojeny se vstupy všech neuronů ostatních, kromě vstupu vlastního. Na vstup neuronů se přivádí pouze hodnoty -1 a 1, popř. 0 a 1.

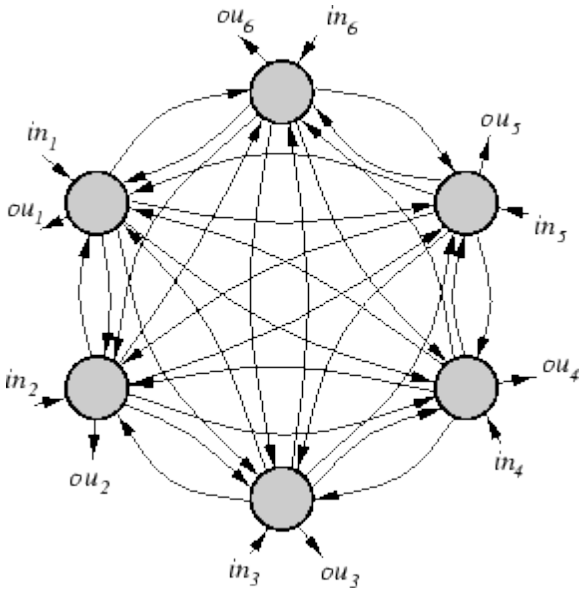
Hopfieldova síť se využívá jako asociativní paměť. Rozdíl oproti běžné paměti je ten, že nepotřebujme znát adresy dat pro vyhledávání, ale stačí nám jen fragment požadovaných dat.

Asociativní paměť lze rozdělit na autoasociativní a heteroasociativní. Princip autoasociativní paměti spočívá v upřesnění (popř. zúplnění) vstupní informace. Heteroasociativní paměť spočívá v tom, že na základě vstupní asociace si lze „vybavit“ určitou sdruženou informaci [11].

Učení Hopfieldofy sítě probíhá ve 2 fázích a to - 1) Změna synaptických vah
2) Testování

Změna synaptických vah probíhá na základě vložení vstupních dat a tím také jejich uchování v asociativní paměti, kterou síť představuje. Testování poté spočívá v předkládání vzroků na vstupy a je hodnocena jejich shoda se vzorkem uloženým v paměti.

Obrázek 5 Hopfieldova síť se šesti neurony



[Zdroj: (1)]

Učení Hopfieldovy sítě

Učení Hopfieldovy sítě může probíhat buď synchronním nebo asynchronním způsobem. Zásadní rozdíl mezi synchronním a asynchronním výpočtem je v počtu neuronů, které se v daném kroku přepočítávají. U synchronního výpočtu se v čase t přepočítávají všechny neurony najednou, zatímco u asynchronního výpočtu se přepočítává pouze jeden náhodně vybraný neuron.

Postup učení

Pro učení Hopfieldovy sítě lze použít Hebbův zákon o učení asociativní paměti. Nejprve je nutné stanovit čas **t**. V čase **t = 0** jsou všechny váhové vektory nulové. Poté jsou síti předkládány trénovací prvky a po předložení každého následujícího prvku se **t** zvýší na **t+1**. Učící proces končí v momentě, kdy jsou síti předloženy všechny prvky trénovacího souboru. Výslednou konfiguraci lze vyjádřit vztahem [11]:

Konfigurace Hopfieldovy sítě

$$w_{ji} = \sum_{k=1}^{p} d_{kj}\, x_{kj} \qquad (6)$$

Kde **p** je počet kroků, **d** je výstup, **x** je vstup a **w** jsou jednotlivé váhy.

Dalším důležitým vzorcem pro Hopfieldovu síť je vztah pro výpočet energie sítě. Ta je závislá na momentálních stavech výstupů sítě. Energie sítě klesá při asynchornním výpočtu, zatímco během synchronním výpočtu energie sítě klesat nemusí. Pro výpočet energie Hopfieldovy sítě se používá vztahu [6]:

Energie Hopfieldovy sítě

$$E(t) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\, y_i(t) y_j(t) \tag{7}$$

Použití Hopfieldovy sítě

Hopfieldova síť nachází nejširší oblast uplatnění v oblasti korekce špatně čitelných nebo zašuměných obrazů. Díky asoiciativní paměti je schopna rozpoznat obrazy které jsou nejen poškozené či jen špatně čitelné, ale také potoočené, či převrácené. Pomocí Hopfieldovy sítě je také možno řešit optimalizační problémy, jako např. optimalizace cesty (problém obchodního cestujícího).

4.2.4 Kompetiční sítě

Na rozdíl od Hebbovského učení, v kompetičních sítích jde o to, že na rozdíl od ostatních sítí je aktivní vždy pouze jeden výstupní neuron - neurony spolu soutěží o aktivitu (*competitive learning*) [11].

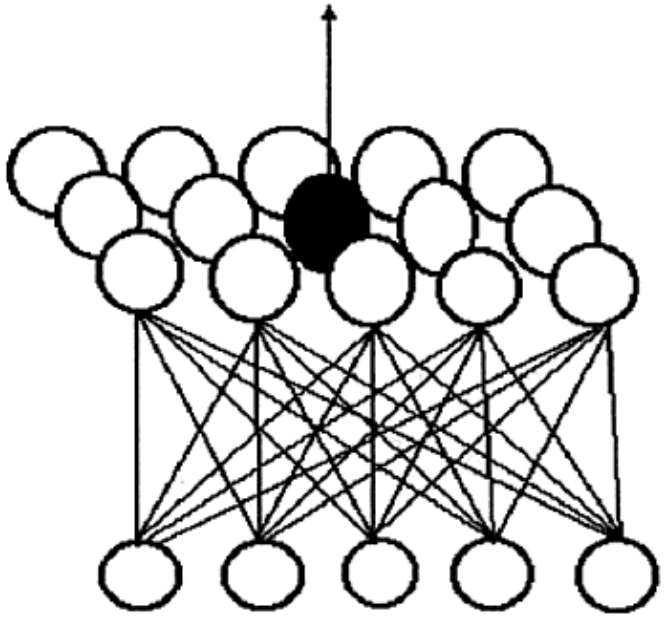
Kompetiční sítě se skládají vždy ze dvou vrstev neuronů - receptorů a výstupních neuronů. Všechny receptory jsou jednotlivě propojeny se všemy výstupními neurony a každému takovému spojení je přiřazena váha. Během výpočtu se jednotlivě síti předkládají vektory **x** a aktivní bude ten výstupní neuron, který bude mít nejnižšší rozdíl svého váhového vektoru **w** od vstupního vektoru **x** [6].

Nejznámějším a nejpoužívanějším typem samooganizační (kompetiční sítě) je tzv. Kohonenova mapa, která byla popsána v r. 1982 T.Kohonenem [7].

Kohonenova mapa

Kohonenova mapa je síť, která je tvořena jednou vrstvou neuronů, seřazených do čtvercové matice. Všechny jsou navzájem propojené. Vazby mezi jednotlivými neurony jsou buď excitační (vazby se sousedními neurony) nebo inhibiční (vazby se vzdálenými neurony) [14].

Obrázek 6 Kohonenova mapa



[Zdroj: (14)]

Učení Kohonenovy sítě

Tzv. Kohonenovo učení vychází podstatou z Lloydova algoritmu. Lloydův algoritmus popisuje učení kompetičních sítí jako postup po sobě následujících kroků. Nejprve se náhodně zvolí reprezentanty dat. Poté se musí určit, která data jsou jednotlivými reprezentanty reprezentována. Jsou to ta data, která jsou danému reprezentantu nejblíže. Následující krok spočívá v nalezení těžiště v jednotlivých množinách a ta data, která se nachází těžišti nejblíže, se pak stávají reprezntanty. Tento postup se opakuje, dokud nedojde k úplnému seřazení.

Kohonenovo adaptační pravidlo

$$w_r(t) = w_r(t-1) + \alpha * \left(x - w_r(t-1)\right) \quad (8)$$

$\mathbf{w_r}$- váhový vektor reprezentantu x

x - vstupní vektor

α - plasticita [5].

Při učení Kohonenovy sítě požadujeme, aby celkový rozptyl všech shluků byl minimální a aby byla zachována topologie vstupního prostoru [6].

Na rozdíl od perceptronu, u SOM (self-organising map) učení probíhá bez učitele, na bázi soutěžního učení. Mapě se postupně předkládají vzory. Své váhy přibližují vzoru právě ty neurony, které od tohoto vzoru měly nejkratší Eukleidovskou vzdálenost. Aby bylo možno co nejpřesněji napodobit určený vzor, známe u těchto samoorganizačních map dva různé typy topologie. Jsou to topologie, určující tvar okolí reprezentantu a to - čtvercové a šestiúhelníkové.

LVQ (Learning vector quantization)

LVQ neboli Learning Vector Quantization v překladu znamená *učící vektorová kvantizace*. Vektorová kvantizace je statistická metoda, která se snaží klasifikovat vstupní vektory do kategorií. [13].

LVQ je algoritmus, který slouží k „doučení“ sítě, která byla v předchozím kroce organizována dle Kohonenova učení. Tentýž autor také specifikoval tři typy LVQ algoritmu. Tyto algoritmy jsou označeny jako LVQ1, LVQ2 a LVQ3.Tyto algoritmy byly vyvinuty pro zpřesnění samoorganizačních map za pomoci učení s učitelem [11].

Použití kompetičních sítí

Jak už bylo uvedeno, kompetiční sítě (vyjma algoritmu LVQ) se učí bez učitele, neboli bez vnější informace. Proto je jejich použití vhodné zejména tam, kde neznáme konkrétní hodnoty výstupů. Jedná se zejména o úlohy, kde je zapotřebí, aby se síť samostatně rozhodovala, byla schopná rozlišovat a třídit jednotlivé objekty nebo signály. Nejčastější

uplatnění tohoto typu sítě lze nalézt v oblastech úpravy zvuku, rozpoznávání fotografií, odstraňování šumů, automatického třídění mnoha dalších příkladech [15].

4.3 Nástroje pro modelování neuronových sítí

Existuje mnoho nátrojů od různých výrobců softwaru, které nám umožňují vytvářet, trénovat a také simulovat chování různých typů neuronových sítí. Může se jednat o komerční úspěšné produkty nebo i produkty opensourcové.

Zřejmě nejpoužívanějším komerčním nástrojem pro simulaci neuronových sítí v prostředí windows je *Matlab*, respektive nástroj *Matlab Neural Network Toolbox*. Tomuto softwaru však velice zdatně konkuruje i *Statistica* a nástroj pro neuronové sítě *Statistica Neural Network*. Zvláštní knihovnu věnující se neuronovým sítím obsahuje také *Simulink*, jež je součástí matlabu. Pomocí tohoto nástroje lze snadno vytvořit schema neuronové sítě libovolného typu i použití a lze ji rovnou v tomto prostředí otestovat.

Z opensourcových programů za zmínku stojí např. *Sage, JmathLib, Maxima, Octave* nebo *FreeMat*. I tyto programy nabízejí možnost práce s neuronovými sítěmi (např. *Octave's Neural Network Package*), co se však jejich možností týče, zdaleka se komerčním softwarům nemohou rovnat.

V následující kapitole se blíže seznámíme právě s nástroji *Matlab Neural Networks Toolbox* a *Simulink*, jelikož v těchto prostředích proběhlo vytvoření a testování modelu, který je hlavní nápní této práce.

4.3.1 Prostředí matlab

Název MATLAB vznikl ze spojení dvou aglických slov Matrix a Laboratory. Byl vyvinut firmou The Mathworks v roce 1984. Cílem bylo vytvoření nástroje, který v sobě obsahuje univerzální prostředí pro matematické a fyzikální výpočty, věděcké analýzy, 2D a 3D grafiku a to vše bez znalosti klasického programování.

Matlab nabízí inženýrský, věděcký a matematický jazyk pro vyjadřování problémů a jejich matematické i grafické řešení. V Matlabu jsou integrovány výpočty, vizualizace a programování ve flexibilním otevřeném prostředí [4].

Neural network toolbox

Tento nástroj implementovaný do prostředí matlabu je velmi užitečný pro tvorbu neuronových sítí kteréhokoli z výše uvedených typů, jejich trénování, simulace, či testování. Rovněž lze přehledně vytvořit datové soubory pro vstupy nebo výstupy neuronových sítí a vše lze možno exportovat jako proměnné přímo do workspacu a dále s těmito objekty pracovat v rámci programování. Ve vyšších verzích matlabu je toolbox již implementován jako standartní součást balíčku a už nikoli jako přídavná knihvna.

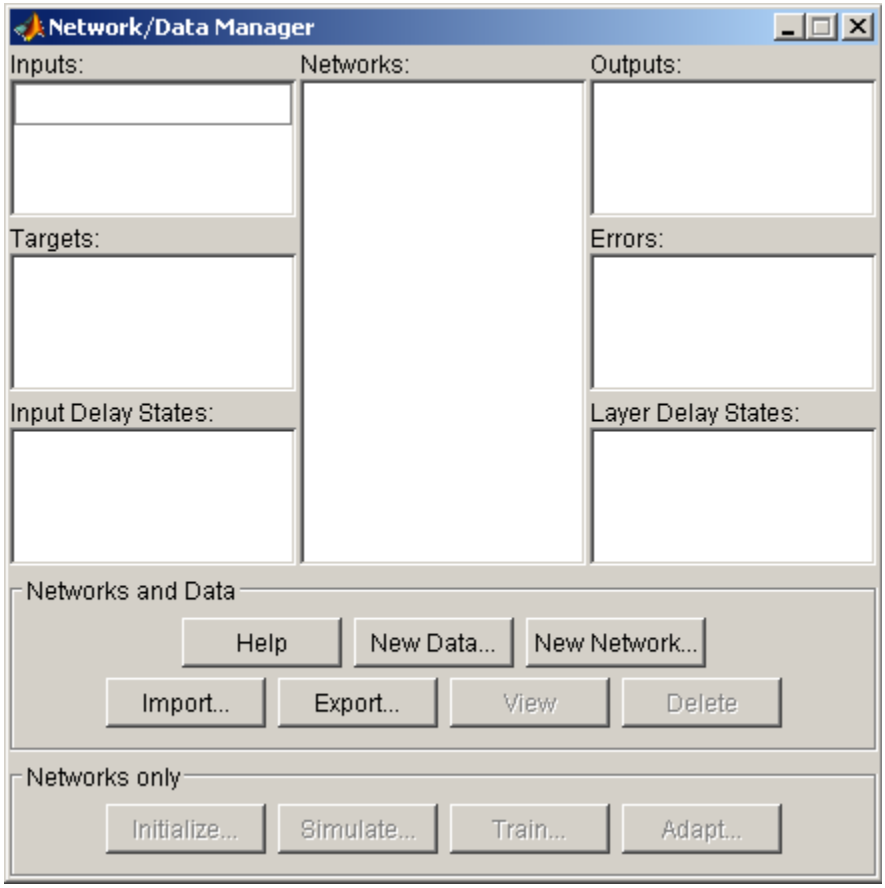
Pro vyvolání nástroje neural network toolboxu v matlabu stačí napsat příkaz **nntool** a potvrdit. Zobrazí se nabídka (Obr. 7), která nám umožní rovnou vytvářet, editovat, či importovat datové soubory nebo již hotové sítě. Pro některé sítě je možné zobrazit i jejich grafické znázornění.

Vstupní a výstupní vektory se zadávají formou matic, kde každý řádek je oddělen středníkem a každý sloupec je oddělen mezerou. Např. zadáním vektoru *[1 2 1;3 2 3; 1 2 1]* získáme vzor, který obsahuje matici:

$$\begin{matrix} 1 & 2 & 1 \\ 3 & 2 & 3 \\ 1 & 2 & 1 \end{matrix}$$

Při volbě vstupních a výstupních vektorů je nutno dávat pozor na to, aby byly reprezentovány maticí stejných rozměrů. Např. je- li vstupní vektor o rozměrech *m* x *n*, pak i výstupní vektor musí mít stejný rozměr.

Obrázek 7 Okno neural network toolboxu



[Zdroj:vlastní via Matlab]

Vytvoření sítě pomocí toolboxu je však jen jednou z několika možností. Další způsob, jak docílit vytvoření sítě, je za pomoci kódu. Kódy ke konkrétním sítím lze nalézt v nápovědě matlabu, kterou lze vyvolat pomocí tlačítka **Help** nebo po stisku klávesy **F1.**

Při vytváření sítě tímto způsobem je důležité neopomenout důležité parametry sítě, kterou chceme pomocí kódu naprogramovat. Mezi nejdůležitější parametry patří např. počet neuronů, jejich porpojení (topologie), přechodové funkce a u vícevrstvých perceptronů také počet vrstev.

Prvním krokem je vždy vytvoření sítě a nastavení základních parametrů. Dalším krokem je vždy inicializace. Pomocí inicializace se provede základní nastavení vah a prahů neuronů. Poté se můžeme pustit do trénování. Trénování má mnoho parametrů, které je potřeba nastavit (např. ukončující podmínka, nastavení parametrů chyby apod.). Během trénování předkládáme prvky trénovacího souboru, dokud není splněna ukončující podmínka (max. počet epoch, konečná chybovost atd.).

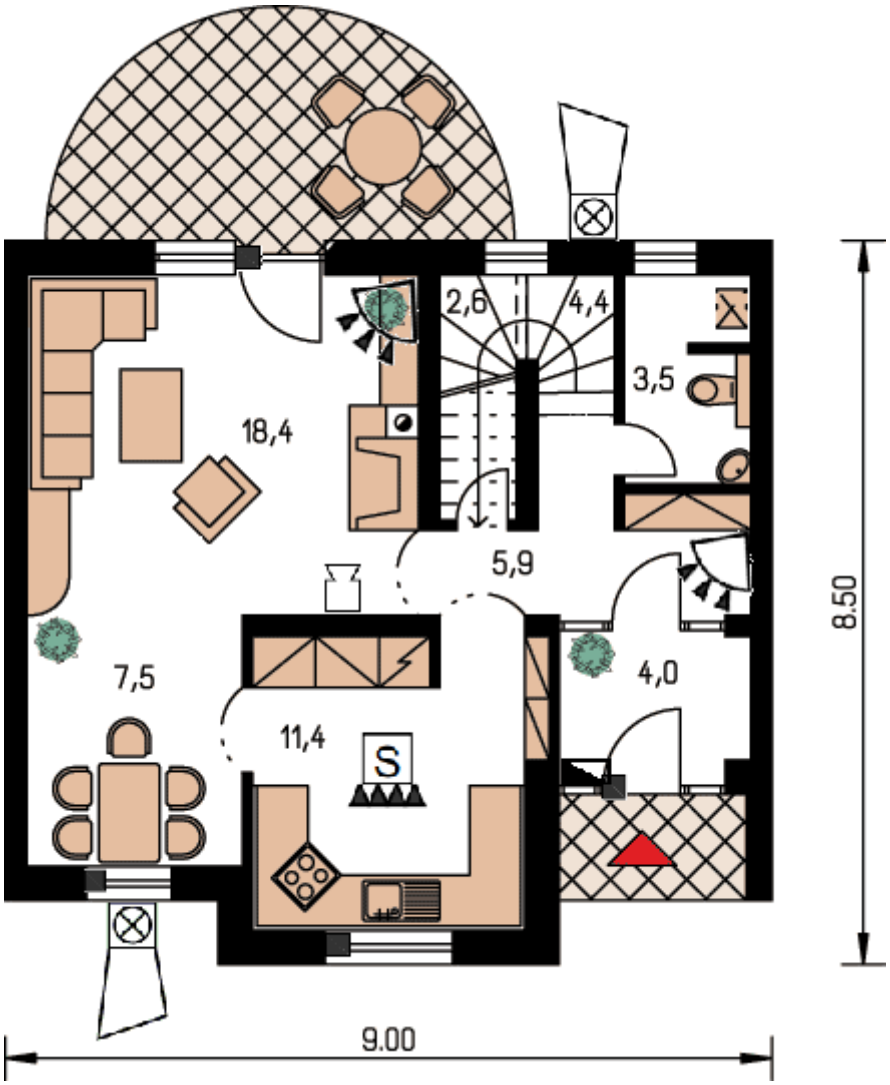
Další možností, jak v matlabu simulovat neuronovu síť, je vytvořit si její model pomocí schemat v prostředí **Simulink**. Zde se nachází knihovny, pomocí kterých lze sestavit prakticky libovlnou síť. Výhodou je, že vše lze rovnou testovat a pracovat tak s celou sítí jako modelem, který se rovnou ladí. A to vše v uživatelsky příjemném prostředí, a bez hlubší znalosti programování.

5 Vytvoření matematického modelu, definování vazeb a jejich závislostí

Aby bylo možné vytvořit neuronovu síť reprezentující zabezpečovací systém, musíme nejprve tento konkrétní systém nadefinovat. Jako příklad poslouží zabezpečovací systém fiktivního objektu, navržený pro účely testování.

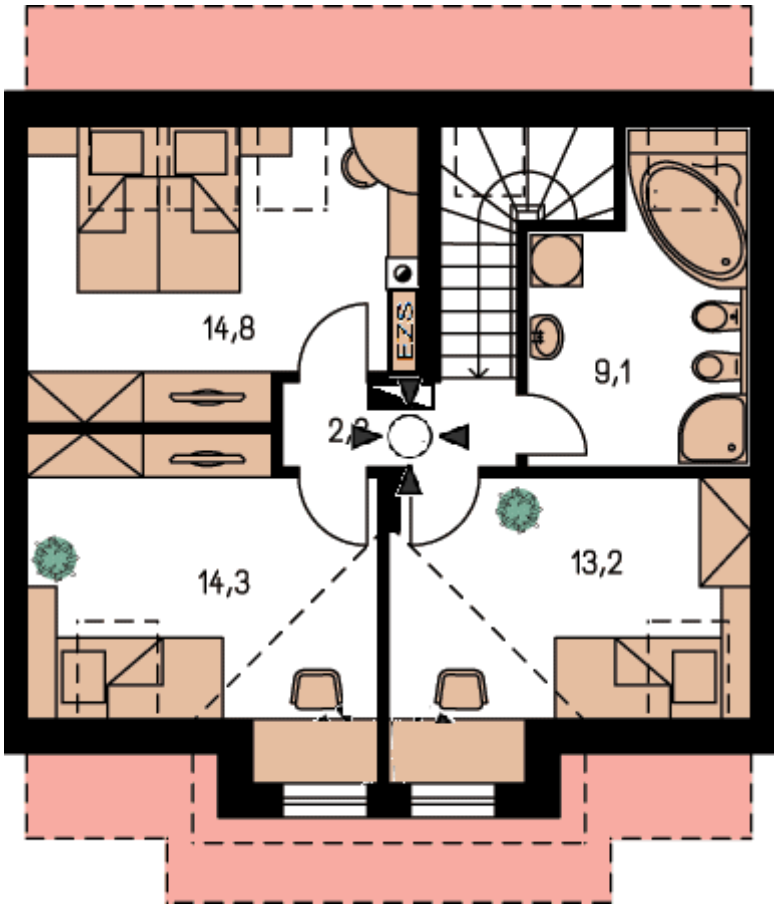
Tento objekt je dvoupatrový řadový rodinný domek se dvěma možnými vstupy do objektu. Půdorys přízemí je zobrazen na obrázku č. 8 a první patro na obrázku č. 9.

Obrázek 8 Přízemí rodinného domu



[Zdroj: vlastní]

Obrázek 9 První patro rodinného domu



[Zdroj: vlastní]

V přízemí (obr. 8) lze vidět dvoje vstupní dveře. Jedná se o hlavní vchod, opatřený magnetickým kontaktem, za nímž je umístěna klávesnice pro aktivaci a deaktivaci zabezpečovacího systému a vchod vedlejší, sloužící ke vstupu na terasu. V tomto případě uvažujeme, že je terasa situována tak, že není přístupná z ulice. Dveře vedoucí na terasu jsou opatřeny magnetickým kontaktem a jsou hlídány PIR detektorem, který zároveň monitoruje prakticky celý obývací pokoj a část jídelny. Pro případ vniku pachatele oknem nad jídelním stolem, je rám rovněž okna opatřen magnetickým kontaktem. Stejně jako okno v jídelně je magnetem opatřeno i okno v kuchyni, kam PIR detektor „nevidí“. V kuchyni je také naplánováno požární čidlo. Poslední čidlo v přízemí je PIR detektor, monitorující chodbu směrem ke schodišti a tím i výstup z koupelny. Co se týče zvukové signalizace, v přízemí je umístěna vnitřní siréna tak, aby byl její poplach slyšitelný po celém domě. Dvě vnější sirény spolehlivě zaručí dostatečnou zvukovou signalizaci do ulice. Ústředna EZS (PZTS) je umístěna v prvním nadzemním podlaží podlaží (Obr. 9) a aby nebyla snadno přístupná, je

schodiště a celá chodba v 1. podlaží hlídána všesměrovým stropním čidelm, hlídající zároveň vstup přes pokoje v případě pokusu o vloupání střešními okny.

Celý objekt bude rozdělen do čtyř zón a dvou podsystémů. První podsystém bude obsahovat stropní všesměrový PIR detektor, hlídající v prvním nadzemním podlaží rodinného domu a bude aktivován pouze tehdy, bude-li dům opuštěný úplně. V případě, že bude dům plný lidí a pokoje ve druhém podlaží budou obsazeny, není tutno nadzemní podlaží zabezpečovat. Velká ložnice (obr. 9) vlevo nahoře je nepřístupná, jelikož střešní okno nebude možné otevřít zvenku. Od schodiště nebude přístup možný, neboť chodba i část schodiště je hlídána právě tímto stropním PIR detektorem.

Druhý podsystém se bude skládat ze tří zón, které se nachází v přízemí. První zónou je magnetický vstupní kontakt, po jehož rozepnutí je nutno zadat deaktivační kód zabezpečovacího systému do této zóny lze s pomocí zapojení ATZ připojit i PIR čidlo, umístěné nad dveřmi, které následují. Druhá zóna je prostor obývacího pokoje,jídelny a kuchyně, hlídaný PIR detektorem umístěným u vstupu na terasu, magnetickým kontaktem dveří směrem k terase a magnetickými kontakty v oknech. Pro současné hlídání na jedné zóně lze magnetické kontakty zapojit do série.(Obr. 8) Třetí zónu bude tvořit kouřový detektor a temper ústředny.

Přehled značení detektorů dle ČSN EN 50131-1/Z1 viz. příloha č.1.

5.1 Definice matematických vazeb

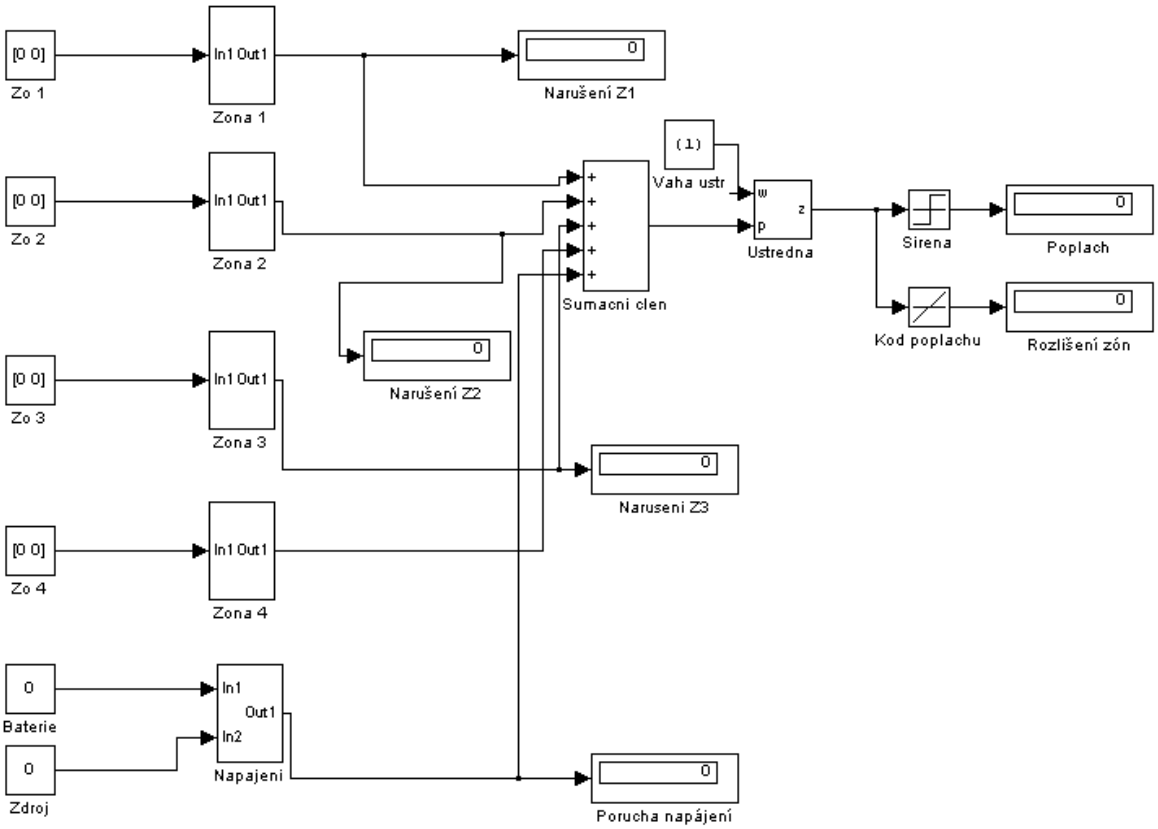
Pro definici matematického modelu zabezpečovacího systému je nejprve nutné určit, jakých hodnot mohou čidla dosahovat. Vycházejme z předpokladu, že čidla mohou být buď ve stavu log 0, neboli stav v klidu (zavřený) a stav log 1, který můžeme nazvat jako stav poplachu (otevřený). Takto nadefinovné stavy odpovídají zapojení NC (normaly closed).

Aby bylo možno detekovat přesné místo narušení objektu, musí mít každá zóna (potažmo každý detektor) svůj unikátní kód (váhu). Prakticky je tento problém vyřešen právě rozdělením detektorů do předem nadefinovaných zón a případně jsou detektory zapojeny s pomocí ATZ. ATZ nám umožňuje na jednu zónu připojit a identifikovat dva unikátní detektory.

5.2 Blokové schema zabezpečovacího systému

Následné (obr.10) schema bylo vytvořeno v simulinku, v matlabu. Schema bylo navrženo tak, aby co nejobecněji interpretovalo funkci zabezpečovacího systému.

Obrázek 10 Schema zabezpečovacího systému (simulink)



[Zdroj: vlastní via Simulink]

Bloky, které jsou pojmenované **Zo1**, **Zo2**, **Zo3**, **Zo4**, **Baterie** a **Zdroj**, představují vstupní vektory (stavy) detektorů, připojených k ústředně a částí zdroje tj. záložní baterie a napájecího transformátoru. Vektory o dvou prvcích obsažených v těchto blocích představují stav detektoru (0 znamená klid, 1 poplach). Zbylé dva jednoprvkové vektory představují stav funkce baterie a napájecího transformátoru (0 znamená odpojeno, 1 připojeno).

Následující bloky, označené jako **Zona 1, Zona 2, Zona 3, Zona 4** a **Napájení** jsou reprezentovány neurony, přesněji perceptrony. Vstupní bloky tvoří vstupní informaci pro daný neuron a na základě těchto vstupních informací dochází k vyhodnocení poplachu. Pro každý prvek vstupního vektoru je přiřazena unikátní váha, která je později vynásobena 1 nebo 0

(podle stavu vstupního detektoru) a vyhodnocena přechodovou funkcí perceptronu. Tím dojde ke generování unikátního kódu poplachu.

Váhy u každého z perceptronů jsou navrženy tak, aby každá zóna generovala poplachy v řádu, který jí byl přidělen. Každý prvek vstupního vektoru má váhu navrženu právě tak, aby bylo možné rozlišit, který z těchto prvků je právě ve stavu log 1 (vyvolání poplachu) a zároveň umožnil zjistit, zda spolu s ním není ve stavu poplachu i druhý detektor na téže zóně. Perceptron, který interpretuje napájení, má za úkol vyslat signál poplachu v případě, že napájecí trafo bude odpojeno a zároveň záložní akumulátor bude téměř vybitý. V tomto případě záleží na interpretaci, jelikož hodnota 0 u baterie bude ve skutečnosti vyšší, resp. tato hodnota by měla být popsána jako procentuální nabití akumulátoru. Pro případ totohoto modelování si však prozatím vystačíme s hodnotami 0 a 1. Stejně tak v tomto modelu není zohledněno to, že by ústředna měla vyslat pouze zprávu o poruše zdroje bez charakteru poplachu v případě, že dojde např. k vypadnutí pojistek a systém se na pár minut ocitne bez primárního napájení. Pro zpřesnění modelu je potřeba definovat podmínky chování v jiném programovacím jazyce.

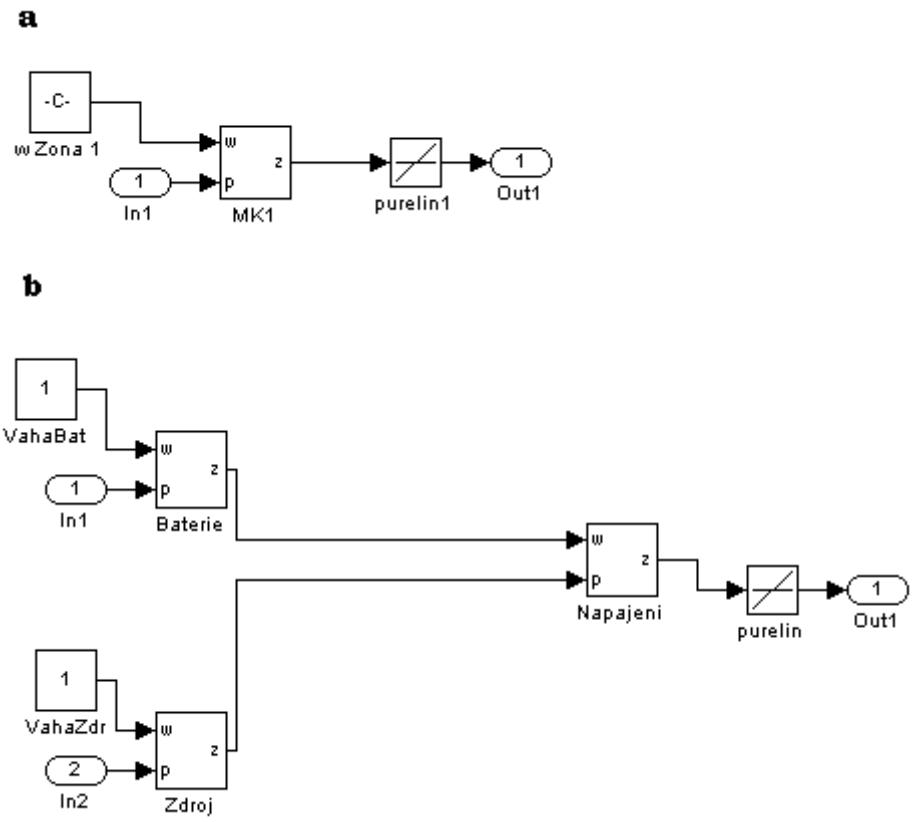
Bloky **Sumační člen** a **ústředna** mají za úkol simulovat chování a ústředny zabezpečovacího systému a vyhodnovcovat vstupní signály. Přechodová funkce ústředny **purelin** je lineární funkce, která nám umožní vyhodnocovat stavy o libovolné hodnotě.Této přechodové funkce je použito u generování tzv. kódu poplachu, který nám má za úkol přesně určit místo detekce poplachu právě díky unikátním vahám detektorům přiděleným. Ostrá nelineární funkce, která rozlišuje pouze stavy 0 a 1 je zde použita pro aktivaci a deaktivaci sirény.

Poslední nezmíněné bloky jsou kontrolní displeje pro zpřehlednění simulace a operace s modelem. Na nich lze přečíst hodnoty vah po vyvolání poplachu na zónách a s těmito hodnotami polachu lze dále pracovat. Je možné je upravit právě tak, aby bylo zřejmé, která zóna, popř. který detektor je právě ve stavu poplachu.

Detailní rozkreslení schematu modelu perceptronu, který má za úkol reprezentovat chování jednotilvých zón, je znázorněn na následujícím obrázku (obr. 11a). Toto schema je dodrženo pro zóny 1 - 4 a kromě pozměněných vah vstupních vektorů se ničím neliší. Podobně, však s drobnými změnami je řešen perceptron, který reprezentuje napájení. Tento perceptron je složen ze dvou sumačních bloků (kvůli odlišení vektoru **Zdroje** a **Baterie)**

a teprve potom je finální hodnota hodnocena posledním sumačním členem a lineární funkcí (obr. 11b).

Obrázek 11 Schemata neuronů (simulink)



[Zdroj: vlastní via Simulink]

Díky tomuto modelu vytvořeného v simulinku lze získat tabulku stavů, kterou lze následně použít jako součást trénovacího souboru pro vybranou neuronovou síť a tím tak danou síť naučit, jak hodnotit poplachy.

Nastavení vah je uvedeno v následující tabulce:

Tabulka 2 Přidělené váhy

	Z1	Z2	Z3	Z4
Detektor č.1	0.1	0.01	0.001	0.0001
Detektor č.2	0.2	0.02	0.002	0.0002

[Zdroj: vlastní]

Výstupem tohoto modelu je kód obsahující informaci o místě popachu. Např. pokud nastane poplach současně na detektoru č. 1 v zóně Z1 a na 2. detektoru v zóně Z3, výstup bude dán vztahem : $1 \times 0.1 + 1 \times 0.002 = \mathbf{0.102}$

Váhy byly rozvrženy tak, aby bylo možno výstupní kód podle této vzorové tabulky opět rozložit. Např. pokud výstupní kód bude obsahovat hodnotu **0.021**, lze pak zcela přesně stanovit, že poplach nastal na 2. detektoru na zóně Z2 a na 1. detektoru zóny Z3.

Tento model byl sestaven tak, aby ověřil zda, a v jaké míře je možné pomocí vícevrstvého perceptronu simulovat chování zabezpečovacího systému podle zadaných kriterií. Podle výsledků tohoto modelu v simulinku lze usoudit, že perceptronová síť bude vhodným typem neuronové sítě pro další testování.

Díky schematickému modelu byla získána data pro trénovací soubor (viz. příloha č.2), který bude aplikován na neuronovou síť typu vícevrstvého perceptronu, abychom zjistili, do jaké míry bude síť schopna zabezpečovací systém napodobit.

6 Model neuronové sítě

Tato kapitola se věnuje samotnému modelu neuronové sítě. Prvním krokem bude výběr typu sítě a odůvodnění proč právě tento typ, definicie základních parametrů, dále bude následovat samotné vytvoření sítě pomocí nástroje **nntool** a pomocí zdrojového kódu. Dalším krokem bude aplikace trénovacího souboru na síť a následný proces učení.

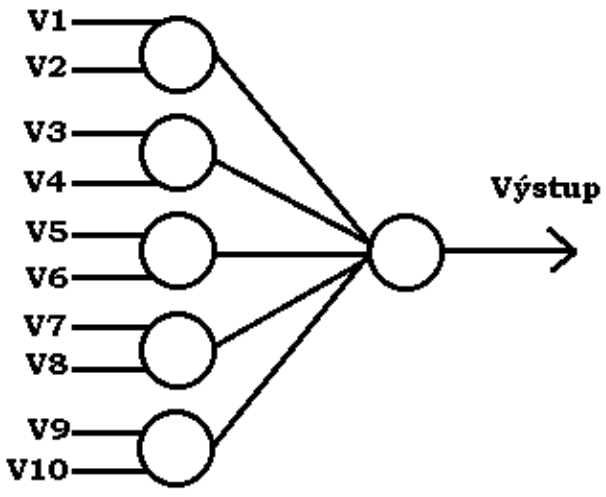
6.1 Výběr typu neuronové sítě

Jak již bylo popsáno v kapitole 4, existuje mnoho typů sítí, které mají své přednosti ale bohužel také některé nedostatky. Nemusí to být vždy záležitost kvality výpočetního algortimu, většinou záleží na vhodnosti použití modelu v konkrétním případě.

Pro tento příklad byl vybrán model neuronové sítě typu vícevrstý perceptron. Tento typ sítě má nespornou výhodu v univerzálnosti, protože topologii sítě lze nakonfigurovat přesně podle požadavků na funkci, výkonnost a možnosti učení.

Vzhledem k tomu, že máme vytvořený trénovací soubor, je vhodné použít učení s učitelem. Prvky trénovacího souboru budou postupně předkládány síti a bude docházet k postupné změně vah, dokud nedojde k „naučení“ všech prvků testovacího souboru.

Obrázek 12 Model neuronové sítě



[Zdroj: vlastní]

Na obr. 12 je model neuronové sítě pro realizaci zabezpečovacího systému, který se skládá z pěti neuronů ve vstupní vrstvě a jedním neuronem ve vrstvě výstupní.Tento model neobsahuje žádné skryté vrstvy. Při návrhu modelu se vycházelo z funkčního schematu ze simulinku (obr. 10). V1 až V10 jsou logické vstupy do zón a na výstupu se bude vyhodnocovat poplašný kód.

Obecný vzorec pro popis vícevrstvého perceptronu

$$a_j = \sum_i w_{ij}\, x_i + w_{j0} \tag{9}$$

$\boldsymbol{a_j}$- výstup

$\boldsymbol{w_{ij}}$- nastavení vah

$\boldsymbol{x_i}$- vstupní hodnota

$\boldsymbol{w_{j0}}$- bias [2].

Vzhledem k tomu, že v našem případě není uvažována funkce biasu, lze tuto rovnici pro náš příklad upravit následujícím způsobem:

Matematický popis modelu

$$Vy = w1 * (V1 + V2) + w2 * (V3 + V4) + w3 * (V5 + V6) + w4 * (V7 + V8) + w5 * (V9 + V10) \tag{10}$$

Vy - výstupní hodnota výpočtu sítě

w1 - **w5** - Váhy vstupních neuronů, odpovídající tabulce č. 2

V1 - **V10** - vstupní hodnoty detektorů a napájení, odpovídající hodnotám **0** a **1**

Prahové hodnoty nejsou v tomto případě definovány.

Při výpočtech neuronové sítě typu vícevrstvý perceptron se využívá zpětného šíření chyb (backpropagation of error). U tohoto algortimu se vychází z funkce , interpretující rozdíl mezi skutečnou a žádanou hodnotou výstupu neuronové sítě, kterou lze popsat pomocí vztahu:

Backpropagation

$$\delta_l = g^{'}(a_l) \sum_s w_{sl} \delta_s \tag{11}$$

$\boldsymbol{\delta_l}$- odchylka posledního (výstupního) neuronu

$\boldsymbol{g^{'}(a_l)}$ - derivace přechodové funkce

$\boldsymbol{w_{sl}}$- nastavení vah

$\boldsymbol{\delta_s}$- odchylka předchozího (vstupního) neuronu [2].

6.2 Vytvoření sítě

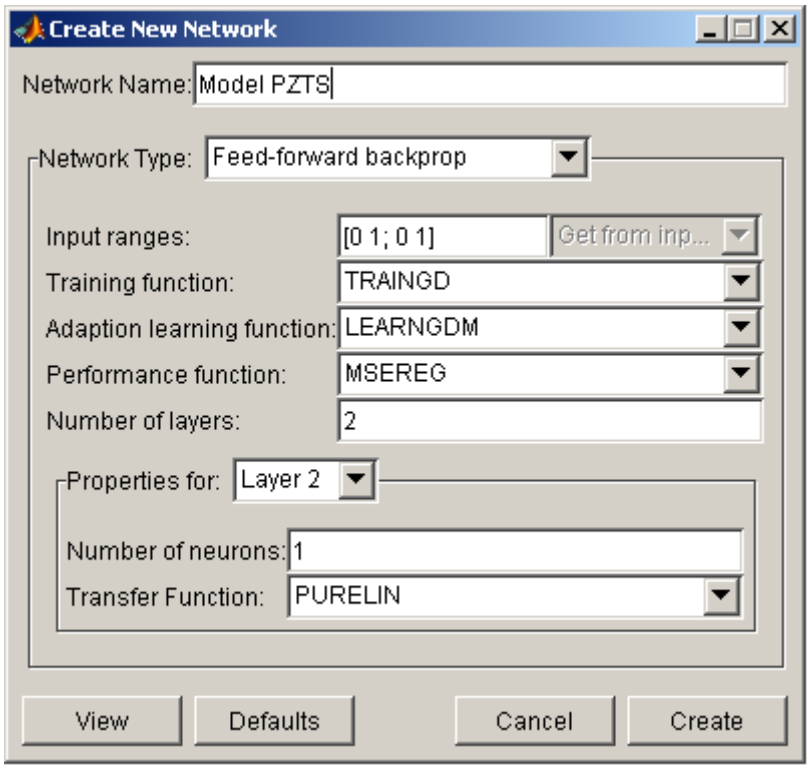
Nyní, když máme vybraný typ sítě a máme nadefinovanou topologii, lze síť snadno vytvořit pomocí nástroje neural network toolboxu (viz následující postup) nebo pomocí zdrojového kódu matlabu.

Pro spuštění nástroje neural network toolbox stačí v prostředí matlab zadat příkaz **nntool.** Po potvrzení tohoto příkazu se zobrazí okno s názvem **Network/Data manager** (viz Obr.7). Pomocí tohoto základního okna lze snadno nadefinovat novou síť, vstupní či trénovací data nebo výstupy. Všechna data lze poté exportovat do worksapcu matlabu a lze s nimi dále pracovat a upravovat je. Postup je možný i obráceně, tedy nejprve ve workspacu nadefinovat požadovaná data a teprve poté je importovat do toolboxu.

Novou síť vytvoříme kliknutím na tlačítko **New network.** Po kliku se nám zobrazí okno, které nám nabízí mnoho možností konfigurace od typu sítě a topologie, trénovací, adaptační a přechodové funkce až po rozsah vstupních hodnot. V našem případě zvolíme síť typu **Feed-forward backprop** (dopředná síť se zpětným šířením chyby) a nastavíme počet neuronů a vrstev. Tento model odpovídá dvěma vrstvám, kde v první vrstvě je nutno nastavit počet neuronů na pět a ve druhé na jeden neuron. Přechodové funkce jsou zvoleny v obou vrstvách na **purelin** (lineární), aby bylo možné rozpoznat hodnoty v rozsahu 0.0001 až 1. Vstupní i výstupní rozsah hodnot tedy odpovídá rozsahu [0 1; 0 1]. Jako trénovací funkce byla zvolena fce **TRAINGD** a jako adaptační fce byla použita **LEARNGD**. Bližší informace o těchto funkcích lze lalézt v nápovědě matlabu. Na závěr je vhodné pro lepší přehled síť

pojmenovat. V tomto případě byla síť nazvána jako **Model PZTS.** Celé nastavení sítě lze vidět na následujícím obrázku (Obr.13).

Obrázek 13 Nastavení parametrů sítě



[Zdroj:vlastní via Matlab]

Nyní se podíváme, jak lze tu samou síť vytvořit pomocí příkazů zdrojového kódu. Nejprve je nutno nadefinovat opět vrstvy sítě a počet neuronů v nich obsažených. Pro tento případ tedy použijeme příkaz **Layers = [5 1];** . Je zřejmé, že síť bude mít 2 vrstvy, přičemž v první vrstvě je neuronů pět a ve druhé pouze jeden (stejně jako v případě nastavení pomocí toolboxu). Dalším krokem je definování přechodových funkcí. Pro každou vrstvu se přechodová funkce definuje zvlášť a tak je pro tento případ použit příkaz **TransFcns = {'purelin' 'purelin'};**. Nyní, když máme definované vrstvy a přechodové funkce, je nutné stanovit trénovací, adaptační a výkonovou funkci. Parametry trénovací funkce se zadávají pomocí příkazu **BTF**, adaptační fce pomocí příkazu **BLF** a výkonová fce pomocí příkazu **PF**. Následující příkaz tedy bude vypadat takto:

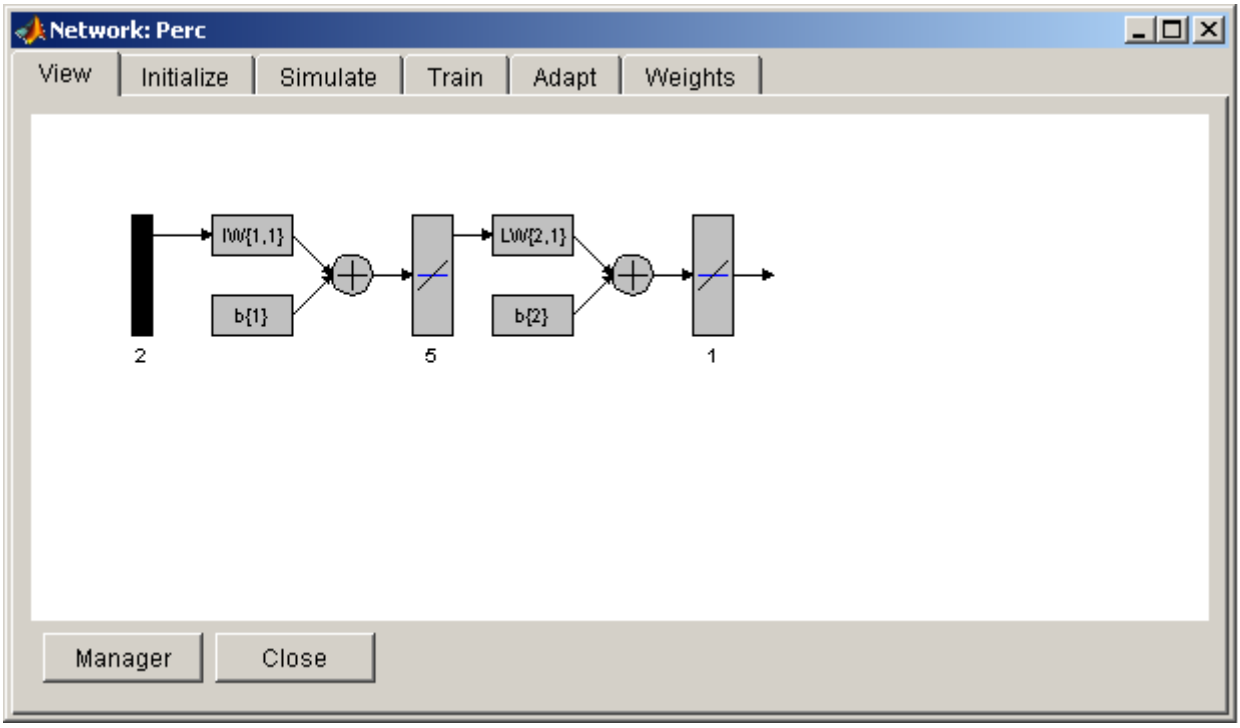
BTF = 'traingd'; BLF = 'learngdm'; PF = 'mse'.

Nyní už máme nakonfigurované parametry sítě a stačí síť vytvořit. Pro dopřednou síť se používá příkaz **newff** a k tomuto modelu se pomocí příkazu připojí nakonfigurované parametry. Model si nazveme jako **PZTS**. Příkaz pro vytvoření sítě je tedy:

PZTS = newff(PR, Layers,TransFcn, BTF, BLF,PF); kde parametr **PR** určuje rozsah hodnot ze vstupních dat.

Náhled právě vytvořené sítě je zobrazen na obr. 14.

Obrázek 14 Náhled sítě v prostředí matlab



[Zdroj:vlastní via Matlab]

6.3 Učení sítě

V okamžiku, kdy máme síť vytvořenou, lze začít s procesem učení. Jak již bylo zmíněno, učení tohoto modelu proběhne za pomoci učitele. Testovací soubor viz. (příloha 2) je nutné rozdělit na vstupní data a tzv. target data (data, která očekáváme na výstupu).

Vstupní data mají tvar vektorové matice o velikosti 5×2 vstupních hodnot. Každá zóna představuje jeden sloupec (2 hodnoty) a nabývá hodnot 0 a 1. Target data jsou matice

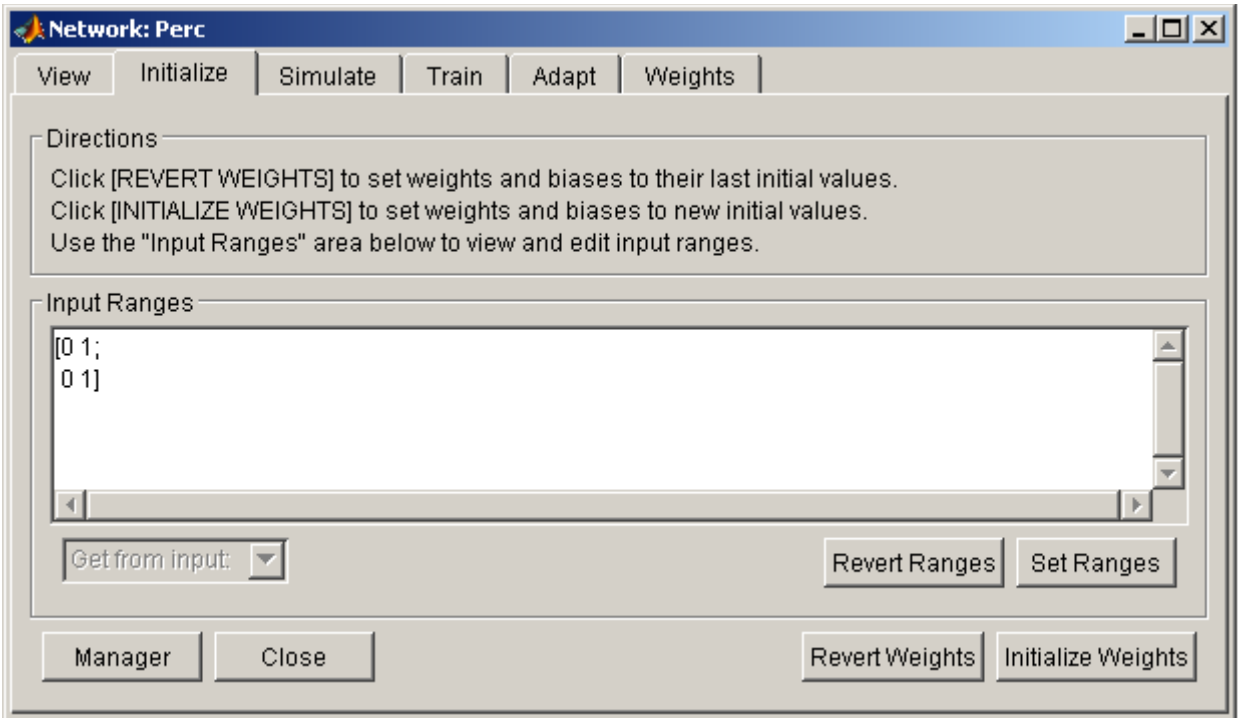
o velikosti 5×1 hodnot a rozsahem se pohybují od 0.0001 do 1 (v závislosti na umístění detektoru a zóny).

Během učení bude docházet k postupnému předkládání prvků trénovacího souboru, dokud neproběhne naučení všech základních prvků. Neuronová síť bude po procesu učení schopna určit i poplachy, které nebyly obsaženy v souboru trénovacích prvků, jako např. kombinovaný poplach na více detektorech.

Před samotným procesem učení je zapotřebí nově vytvořenou síť inicializovat. Během inicializace dojde k prvnímu náhodnému nastavení vah, které se podle parametrů učení budou postupně upravovat.

Inicialiuzaci lze provést opět pomoci toolboxu, kde je potřeba otevřít náhled sítě a v kartě **Initialize** a inicializaci provést pomocí tlačítka **Initialize Weights (**viz obr. 14**).**

Obrázek 15 Inicializace vah



[Zdroj:vlastní via Matlab]

Nyní jsme připraveni postupovat podle učebního algoritmu učení s učitelem. Pro tento model je použito funkce **TRAINGD**, což znamená učení za pomoci snižování gradientu chyby. U tohoto postupu se porovnává výstup s požadovanou hodnotou, určí se nastavení vah a výpočet výstupu sítě.Výstup sítě je porovnáván s požadovanou hodnotou výstupu a určí se

chyba.Tato chyba se následně šíří zpětně od výstupu ke vstupu. Opakováním tohoto postupu dojde ke snížení chyby na požadovanou úroveň a tím k optimálnímu nastavení synaptických vah.

Metodu zpětného šíření chyby lze popsat základním vztahem jako chybovou funkci E:

Výpočet chybové funkce E

$$E^n = \frac{1}{2}\sum_{k=1}^{c}(y_k - t_k)^2 \tag{12}$$

E^n - hodnota chybové funkce

y_k - výstupní hodnota

t_k - target (požadovaná) hodnota [2].

Dosazením parciální derivace získáme vztah pro výpočet zpětného šíření chyby:

Výpočet zpětného šíření chyby

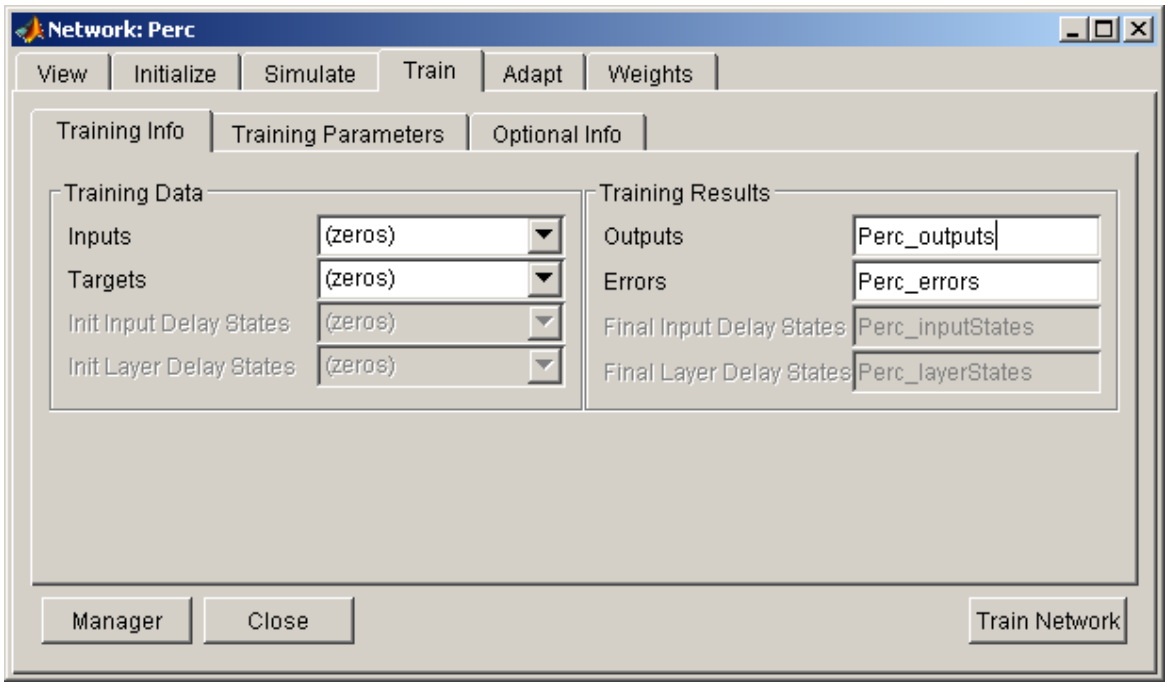
$$\frac{\partial E}{\partial w_{ij}} = \sum_{n}\frac{\partial E^n}{\partial w_{ij}} \tag{13}$$

E - chybová funkce

w_{ij} - váhy mezi i-tými a j-tými prvky [2].

Po inicializaci vah začneme předkládat prvky trénovacího souboru. Pro trénování sítě je potřeba kliknout na kartu **train** u vytvořené sítě. Na této kartě (obr. 16) lze vidět možnosti trénování. Jako první je nutné vybrat vstupní data **Input** a požadovaný výstup **Target.** Rovněž si lze zvolit název výstupního souboru, kde budou zapsána data z procesu učení **Errors** (chyby) a **Outputs** (výstupy).

Obrázek 16 Trénování sítě

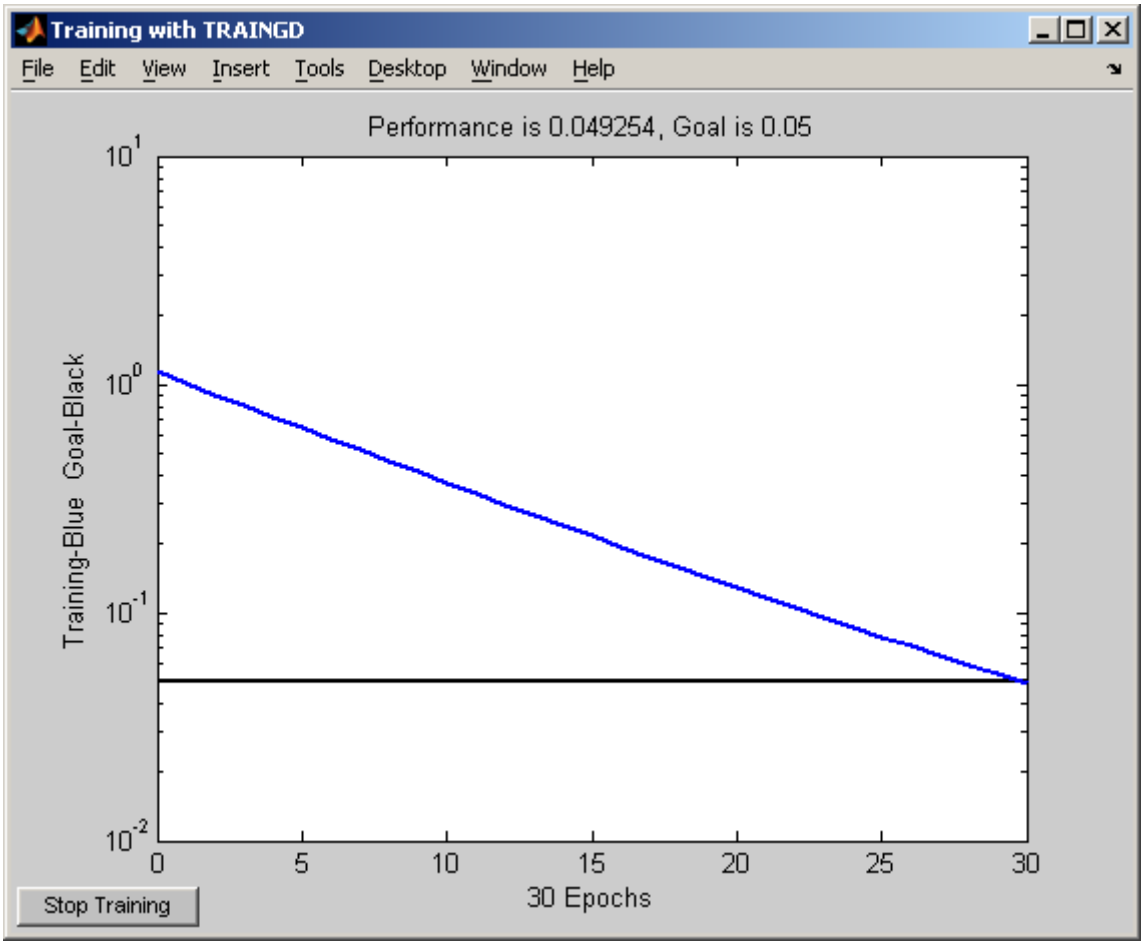


[Zdroj:vlastní via Matlab]

Podle předchozího obrázku vybereme vstupy **Inputs** a výstupy **Targets**. V kartě **Training parameters** nastavíme parametry trénování jako např. počet epoch, požadovanou maximální odchylku, max. povolenou chybu nebo např. čas trénování.

Trénování sítě spočívá v postupném předkládání prvků trénovacího souboru. Po předložení každého dalšího prvku dochází ke změně synaptických vah. Tím se síť adaptuje a její reálný výstup se postupně přibližuje našim požadovaným hodnotám. Proces učení modelu neuronové sítě PZTS je znázorněn na následujících grafech, kde je zobrazeno postupné přiblížení křivky výkonnosti sítě k požadovaným hodnotám. Čím více se tato křivka přibližuje požadované hodnotě, k tím menším změnám dochází během epoch učení.

Graf 1 Proces trénování sítě - 1. krok

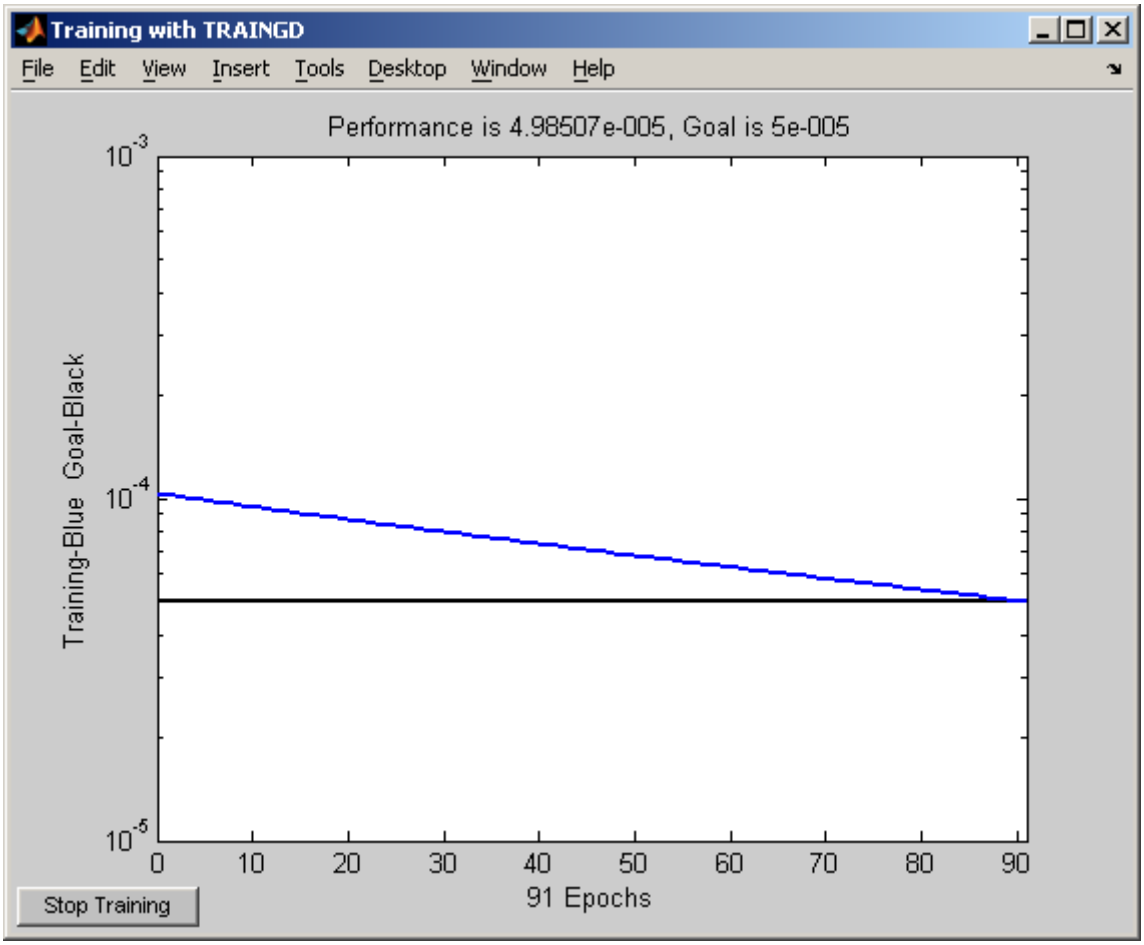


[Zdroj:vlastní via Matlab]

Během prvního kroku dochází k nejvyšší změně synaptických vah z defaultního nastavení směrem k našim požadavkům. Výkonnost trénování neuronové sítě závisí na složitosti řešeného problému a je potřeba s tímto údajem experimentovat. Pokud by se výkonnost sítě (performance) zvolila příliš vysoká, došlo by k nestabilitě učícího algoritmu a k přeučení sítě. V opačném případě, pokud je výkonnost zvolena příliš malá, potřebovali bychom k dostatečné změně vah zbytečně mnoho iteračních cyklů (učících epoch).

Abychom učinili první krok učení, byla výkonnost trénovacího algortimu nastavena na hodnotu 0.05. Při této výkonnosti trvala změna vah 30 učebních epoch (viz graf. č. 1).

Graf 2 Proces trénování sítě - 2. krok



[Zdroj:vlastní via Matlab]

V tomto kroce se váhy změnily jen nepatrně, oproti kroku prvnímu. Výkonnost trénovací fce tohoto kroku je nyní pouze 5×10^{-5}. Protože zde byla nastavena výrazně nižší výkonnost trénovacího algoritmu, bylo k upravení vah zapotřebí více iteračních cyklů.

Nyní provedeme stejný trénovací proces pomocí zdrojového kódu matlabu:

Prvním krokem, stejně jako u postupu pomocí toolboxu je inicializace sítě. Ta se provede pomocí příkazu **PZTS = init(PZTS);**. **PZTS** je názvem sítě, kterou jsme si nadefinovali v předchozím kroce vytvoření.

Samotné trénování nadefinované neuronové sítě probíhá pomocí příkazu **[net,tr,Y,E,Pf,Af] = train(net,P,T,Pi,Ai,VV,TV);** kde **net** je název naší sítě, **tr** je tzv. training record, sloužící k reprezentaci průbehu učení sítě. **Y** jsou výstupy sítě, **E** je výstupní chyba sítě, **Pf** a **Af** jsou ukončující podmínky. Tyto hodnoty budou vráceny sítí jako výstupy.

Vstupy do sítě jsou následující: **P** jsou vstupy do sítě (trénovací soubor), **T** je target, neboli požadovaný výstup, **Pi** a **Ai** jsou vstupní podmínky, defaultně nastaveny na hodnotu **0.** Poslední dva parametry jsou **VV** validační vektor a **TV**, které jsou defaultně nastaveny jako prázdné hodnoty **[ ].**

Nyní upravíme příkaz pro naši síť. Za **net** dosadíme jméno **PZTS** a za **P** a **T** dosadíme vstup **I1** a target **T1.** Ostatní parametry není potřeba zapisovat, stačí tedy použít jejich defaultní hodnoty. Trénovací příkaz bude mít poté tuto podobu: **[PZTS,tr,Y,E]=train(PZTS,I1,T1);**.

Tento příkaz lze aplikovat za sebou tolikrát, kolik máme prvků v trénovacím souboru. Tím dojde k postupnému natrénování sítě. Po každém kroce nám navíc pole matlabu vrátí hodnoty **E** (chyby) a **Y** (výstupu sítě).

K nastavení trénovací funkce slouží příkaz **PZTS.trainFcn**. V našem případě se jedná o funkci **Traingd,** která již byla pospána výše. Všechny nastavitelné parametry trénování lze zobrazit a případně upravit po napsání příkazu **PZTS.trainParam.** Jedná se o stejné parametry jako u trénování sítě pomocí toolboxu, tedy o parametry **epochs** (počet učících epoch), **goal** (maximální povolená chyba) a další.

6.4 Simulace sítě

Nyní, když máme síť vytvořenou, inicializovanou a natrénovanou, můžeme vyzkoušet, jak bude reagovat na doposud neznámé podněty (vektory, které se síť neučila). Simulační vektory jsou uvedeny v následující tabulce. Předpokládejme, že síť by po naučení měla být schopná na základě vstupních vektorů vyhodnotit, na kterém detektoru, či zóně se poplach vyskytl.

6.4.1 Definice simulace simulačních vektorů

Tabulka 3 Simulační vektory

Z1		Z2		Z3		Z4		Napájení	
Det.1	**Det.2**	**Det.1**	**Det.2**	**Det.1**	**Det.2**	**Det.1**	**Det.2**	**Baterie**	**Zdroj**
1	1	0	0	0	0	0	0	1	1
0	0	1	0	1	0	0	0	1	1
0	0	0	0	0	1	1	0	1	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0

[Zdroj: vlastní]

V tabulce č.3 jsou definovány náhodné situace, které nebyly síti předloženy jako prvek trénovacího souboru. Každý řádek představuje narušení určitých detektorů v zónách. V prvním řádku je představena situace, kdy pachatel způsobí poplach současně na obou detektorech, které jsou nadefinovány v první zóně. Další řádky simulují narušení 2 detektorů v odlišných zónách a poslední 2 řádky simulují potíže s napájením.

Aby bylo možné tyto simulační stavy předložit síti, je nutné je upravit jako matici vstupu. Stejně jako trénovací prvky, i tyto simulační prvky musí být obsaženy v matici o 5×2 prvcích, kde první řádek matice představuje každý lichý řádek tabulky č. 3. Druhý řádek matice bude obsahovat všechny sudé řádky tabulky č. 3.

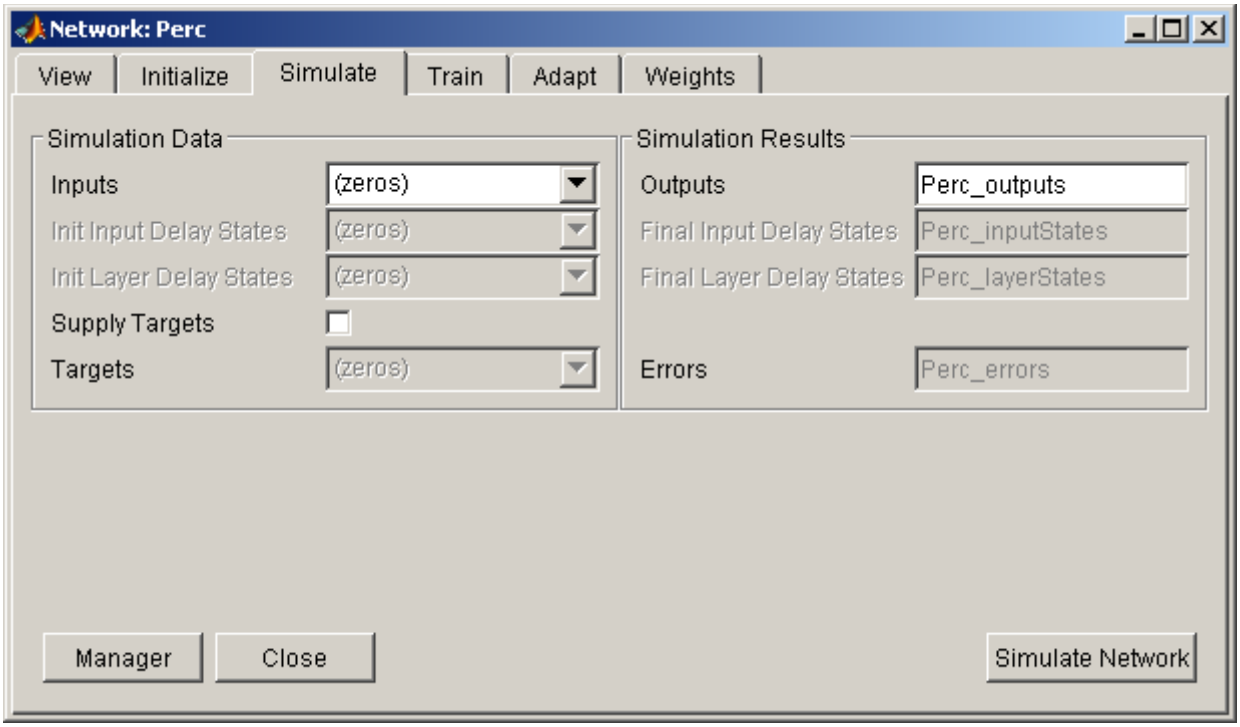
Jako názorný příklad je použit přepis prvního řádku z tabulky do matice, která bude mít následující tvar: [1 0 0 0 1;1 0 0 0 1].

6.4.2 Proces simulace

Obdobně jako vytvoření i trénování, také simulaci sítě je možné provést oběma způsoby, tedy za pomoci toolboxu i za pomoci zdrojového kódu.

Jako první bude simulace demonstrována v toolboxu. Abychom mohli zahájit proces simulace sítě, je nutné v základní nabídce otevřít síť. Po otevření nabídky sítě je nutné otevřít záložku **Simulate** (viz obr. 17)

Obrázek 17 Simulace sítě



[Zdroj:vlastní via Matlab]

Nyní již stačí vybrat vstup, který chceme simulovat. V našem případě použijeme výše uvedený vektor podle tabulky č.3.

Během procesu simulace sítě dochází k výpočtu výstupu v závislosti na nastavení vah, který byl vypočítán na základě trénování sítě. Hodnota výstupu bude zapsána do proměnné, v tomto případě do proměnné s názvem **Perc_outputs** a v základním okně ji lze jednoduše otevřít.

V případě, že chceme simulaci spustit pomocí zdrojového kódu, je nutné použít příkaz **sim(net,p)**. Tento příkaz je vhodné doplnit o přiřazení do proměnné, jako např. **a = sim(net,p)**.

Net zde zastupuje síť, kterou chceme simulovat a **p** je vstupní vektor (matice) simulace. V našem případě tedy použijeme příkazy **Sim1 = [1 0 0 0 1;1 0 0 0 1];**

a = sim(PZTS,Sim1). Hodnoty zapsané v proměnné **a** lze vypsat do workspacu nebo zobrazit v grafu pomocí příkazu **plot(a)**.

7 Výsledky a zhodnocení

Po úspěšné simulaci lze shrnout, k čemu jsme došli. Výstupní hodnota je matice ve tvaru 5×1 hodnot. Tyto hodnoty interpretují stav zabezpečovacího systému. V případě, že je pomocí vstupů zaznamenán poplach, je možné pomocí výstupního vektoru snadno identifikovat jeho zdroj.

7.1 Výsledky simulace

Nejprve je nutné vyzkoušet, jak systém zareaguje na stav, kdy je zabezpečovací systém v klidu, tedy všechny zóny mají na vstupech přivedeny stavy log. 0 a napájení baterie i zdroje log.1.

Použijeme tedy příkaz simulace, který je uveden výše. Simulační vektor **Sim1** změníme tak, aby jeho hodnoty obsahovaly matici **[0 0 0 0 1; 0 0 0 0 1]**. Tato matice odpovídá prvnímu prvku trénovacího souboru. Spuštěním simulačního příkazu a vyvoláním proměnné **a** se dočkáme odpovědi na tento příkaz která bude zobrazena jako: **a = [0.19 0.19 0.19 0.19 0.086]**.

Hodnota **0.19** tedy odpovídá stavu, kdy je daná zóna v klidu a nevykazuje žádné známky narušení. Poslední hodnota matice (hodnota **0.086**) představuje stav, kdy je napájení v pořádku.

Nyní již můžeme testovat reakce výstupu sítě na simualční vektory, které jsou uvedeny v tabulce č. 3. Jako první si vezmeme případ, kdy zavedeme poplach na oba detektory první zóny, který odpovídá prvnímu řádku této tabulky. Vektor **Sim1** tedy bude mít hodnoty **[1 0 0 0 1; 1 0 0 0 1]**. Po provedení simulace dostaneme výsledek v podpbě výpisu proměnné **a = [0.086 0.19 0.19 0.19 0.086]**. V případě narušení zón tedy hodnota **0.086** znamená, že byly současně narušeny oba detektory v dané zóně (umístění k-tého prvku v matici).

Abychom zjistili, jak se výstup změní v případě, že bude narušen pouze první detektor jedné zóny a pouze druhý detektor zóny jiné, dosadíme do vektoru **Sim1** hodnoty matice odpovídající 3. řádku tab. č.3. Tedy jedná se matici ve tvaru **[0 0 0 1 1; 0 0 1 0 1].** Odpovědí na tento simulační krok nám bude hodnota **a = [0.19 0.19 0.22 0.05 0.086].** Hodnota **0.22** nám

interpretuje stav, kdy je narušen pouze druhý detektor dané zóny a hodnota **0.05** je vrácena při narušení prvního detektoru ze zóny odlišné.

Celý soubor simulačních vektorů a výsledky simulace jsou zobrazeny v tabulce č. 4.

Tabulka 4 Výsledky simulace

Vstup	Narušené zóny	Narušené detektory	Výstup
[0 0 0 0 1;0 0 0 0 1]	0	0	[0.19 0.19 0.19 0.19 0.086]
[1 0 0 0 1;1 0 0 0 1]	1	1 a 2	[0.086 0.19 0.19 0.19 0.086]
[0 0 0 1 1;0 0 1 0 1]	3 a 4	2 a 1	[0.19 0.19 0.22 0.05 0.086]
[0 0 0 0 0;0 0 0 0 1]	Napájení	Baterie	[0.19 0.19 0.19 0.19 0.22]
[0 0 0 0 0;0 0 0 0 0]	Napájení	Baterie +Zdroj	[0.19 0.19 0.19 0.19 0.19]

[Zdroj: vlastní]

7.2 Zhodnocení simulace

Jak je patrné z tabulky č. 4, hodnoty výstupu se mění podle změny vstupních vektorů a celý model je proto funkční. Co se týče vyhlášení poplachu při poruše napájení, tam je hodnoty potřeba chápat jako opačné, vůči hodnotám výstupů reagující na spuštění poplachu zón. Výstupní vektor je možné chápat jako stavový kód ústředny na základě kterého by ústředna měla vyhlašovat poplachy. Pro vyšší přehlednost jsou kódy uvedeny v následující kódovací tabulce:

Tabulka 5 Kódovací tabulka

Kódy zón				Kódy napájení		
Kód	Význam	Poplach		Kód	Význam	Poplach
0.19	Bez poplachu	NE		0.19	Porucha baterie i zdroje	ANO
0.22	Poplach 2. detektoru	ANO		0.22	Porucha baterie	NE
0.05	Poplach 1. detektoru	ANO		0.05	Porucha zdroje	NE
0.086	Poplach obou detektorů	ANO		0.086	Napájení v pořádku	NE

[Zdroj: vlastní]

Hodnoty výstupů, jsou závislé na topologii sítě a struktuře trénovacího souboru. Trénovací soubor byl vytvořen tak, aby byla síť schopna rozeznat 4 základní stavy - viz. tabulka č. 5.

Tento model byl navržen jako nejjednodušší možná varianta zabezpečovacího systému. Použitím obdobného postupu by bylo možné tento model rozšířit např. o bezdrátový modul, různé druhy kódování nebo např. připojení vstupů PGM.

8 Závěr

Problematika neuronových sítí je známá již poměrně dlouhou dobu. Metody algoritmů řešící tuto problematiku se však stále vyvíjejí a vylepšují. Stejně tak vznikají nové příležitosti pro tyto výpočetní modely. Oblasti využití neuronových sítí jsou velice široké, od regresních úloh, přes rozpoznávání znaků nebo obrázků až po prognostické metody.

V této práci byl zpracován model zabezpečovacího systému nejprve jako matematický model, u kterého byly zavedeny vstupy simulující chování detektorů v jednotlivých zónách. Byly stanoveny předpokádané výstupy, které by měl model splňovat, aby mohl být označen za funkční. Tento matematický model byl aplikován na model perceptronové sítě, která se skládala ze dvou vrstev, přičemž první vrstva obsahovala pět a druhá vrstva jeden perceptron. Přechodové funkce u perceptronů byly zvoleny jako lineární.

Po vytvoření trénovacího souboru došlo k jeho úpravě tak, aby bylo možné jej implementovat na síť. Trénovací soubor byl rozdělen na vstupní data a cílová (target) data tak, aby došlo k postupnému naučení všech prvků tohoto souboru. Proces učení byl popsán v kapitole 6.3 a znázorněn pomocí grafů.

Když byla síť natrénovaná, bylo důležité vyzkoušet, do jaké míry je tento model funkční. Tohoto testu bylo docíleno pomocí simulace stavů, které nebyly síti předloženy jako prvky trénovacího souboru. Výstupy sítě odpovídaly předpokladům a hodnoty výstupu byly popsány v tabulce č. 5 v kapitole 7.2.

Model neuronové sítě zpracovaný v této práci je funkční. Jedná se o základní model, který by bylo možné dále rozšířit např. o bezdrátový modul (detekce bezdrátových detektorů), různé druhy režimů pro zakódování nebo několik typů poplachu.

Zároveň tento model otevírá nové možnosti pro testování zabezpečovacích systémů pomocí ostatních typů neuronové sítě. Každá neuronová síť má své klady i zápory, více či méně vhodné pro řešení této problematiky. Účelem dalších výzkumů může být vznik upraveného modelu, který bude schopen sám určovat míry zabezpečení daných objektů, nebo v opačném případě na základě zadané míry zabezpečení a požadovaných typů použitých detektorů navrhnout strukturu zabezpečovacího systému, způsoby používání dle

individuálních potřeb uživatelů, nebo ovládání prvků se zabezpečovacím systémem spojených.

9 Seznam literatury

[1] ANDREJKOVÁ, G., SINČÁK, P. *Neuron-ai.tuke* [online]. 9.9. 1998 [cit. 2013-01-24]. Neurónové siete Inžiniersky prístup. Dostupné z WWW: <http://neuron-ai.tuke.sk/cig/source/publications/books/NS1/html/index.html>.

[2] BISHOP, CH, M. *Neural networks for pattern recognition*. 2. vydání Oxford: Oxford University Press, 1995, 482 s. ISBN 978-0-19-853864-6.

[3] BURANT, J., CSIRIK, V., DUDÁŠ, J., DVOŘÁČEK, K., DVOŘÁČEK, V., FENCL, F., KUNC, J., LAIFR, J., MACHÁČEK, V., MAIXNER, T., URBAN, Z., VOTRUBA, Z. *Elektrotechnické a telekomunikační instalace.* Praha: Verlag Dashöfer, 2008. ISSN 1803-0475.

[4] DAVIS, T, A. *Matlab Primer.* 8*th* edition Boca Raton: CRC Press, 2010. 232 s. ISBN 978-1-4398-2862-5.

[5] HOŘEJŠ, J., KUBÁT, M., LAŽANSKÝ, J., MAŘÍK, V., ŠTĚPÁNEK, P., ŠTĚPÁNKOVÁ, O., ZDRÁHAL, Z. *Umělá inteligence.* 1. vydání Praha: Academia, 1993. 264 s. ISBN 80-200-0502-1.

[6] JIROUŠEK, R., VESELÝ, A. *Metody umělé inteligence.* 1. vydání Praha: Reprografické studio provozně ekonomické fakulty ČZU v Praze, 2012. 100 s. ISBN 978-80-213-2295-0.

[7] KOHONEN, T. *Self-organising Maps.* 3. vydání: Springer, 2001, 501 s. ISBN 978-3-54-067921-9.

[8] KŘEČEK, S. *Příručka zabezpečovací techniky.* 3.vydání Blatná: CRICETUS, 2006. 350 s. ISBN 80-902938-2-4.

[9] MIKULA, T. Orsec [online]. 10.11. 2010 [cit. 2012-12-26]. Konec EZS v Čechách !?!. Dostupné z WWW: <http://www.orsec.cz/cs/informacni-servis/clanky-a-komentare/konec-ezs-v-cechach_38-435/>.

[10] MOLNÁR, K. Elektrorevue [online]. 22.2.2000 [cit. 2013-01-24]. Úvod do problematiky umělých neuronových sítí. Dostupné z WWW: <http://www.elektrorevue.cz/clanky/00013/index.html>.

[11] NERUDA, R., ŠÍMA, J. *Teoretické otázky neuronových sítí.* 1. vydání Praha: MATFYZPRESS, 1996. 390 s. ISBN 978-80-858-6318-5.

[12] NOVÁK, M. a kol. *Umělé neuronové sítě. Teorie a aplikace*. 1. vydání Praha: C.H.Beck, 1998. 382 s. ISBN 80-7179-132-6.

[13] SIVANANDAN, S. N., DEEPA, S. N. *Introduction to Neural Networks Using Matlab 6.0.* 2nd reprint New Delhi: Tata McGraw-Hill, 2006. 656 s. ISBN 0-07-059112-1.

[14] SKLENÁK,V. *Data, informace, znalosti a Internet.* 1. vydání Praha : C. H. Beck, 2001. 507 s. ISBN 978-80-717-9409-7.

[15] VOJÁČEK, A. *Automatizace.hw* [online]. 14.5. 2006 [cit. 2013-02-05]. Samoučící se neuronová síť - SOM, Kohonenovy mapy. Dostupné z WWW: <http://automatizace.hw.cz/clanek/2006051401>.

Seznam obrázků

Seznam tabulek

Seznam vzorců

Seznam grafů

Seznam příloh

Příloha 1Schematické značky PZTS dle normy

Schématické značky dle ČSN EN 50131-1/Z1		
ZKR.	ZNAČKA	POPIS
MG		Magnetické čidlo otevření
MGT		Magnetické čidlo otevření odolné
DST		Čidlo rozbití skla
DTS AM		Čidlo rozbití skla antimasking
PI		Čidlo kontaktní PIEZO
PIR		PIR vějíř
PIR V		PIR vějíř venkovní
PIR AM		PIR vějíř antimasking
PIR D		PIR dlouhý dosah
PIR Z		PIR záclona

Schématické značky dle ČSN EN 50131-1/Z1		
ZKR.	ZNAČKA	POPIS
PIR ZAM		PIR záclona antimasking
PIR ID		čidlo s vlastní adresou
UZ		Ultrazvikové čidlo
PIR DV		PIR záclona dveřní
PIR S		PIR stropní čidlo
PIR/DTSS		PIR stropní kombinované s DTS
IZ		IR infrazávora
IZV		IR infrazávora - vysílač
IZP		IR infrazávora - přijímač
MW		Mikrovlnné čidlo

Schématické značky dle ČSN EN 50131-1/Z1		
ZKR.	**ZNAČKA**	**POPIS**
PIR/MW		duální čidlo
PIR/MWS		Duální čidlo stropní
VIB		Otřesové čidlo
MC		Čidlo poslední bankovky
TH		Tísňový hlásič tlačítkový
TL		Tísňový hlásič lišta
TC		Technologický hlásič
GH		Hlásič úniku plynu
PH		Hlásič požáru
SG		Signalizace optická

Schématické značky dle ČSN EN 50131-1/Z1		
ZKR.	**ZNAČKA**	**POPIS**
SZ		Signalizace optická a akustická
SI		Výstražné zařízení siréna vnitřní s blikačem
SS		Výstražné zařízení siréna vnitřní
SE		Výstražné zařízení siréna vnější s blikačem
SB		Výstražné zařízení siréna vnější bez blikače
MJ		Výstražné zařízení maják
US	EZS	Ústředna EZS
PS		Napájecí zdroj
EXP	EXP	Expandér, linkový modul, koncentrátor
TAB	TAB	Tablo EZS

Schématické značky dle ČSN EN 50131-1/Z1		
ZKR.	**ZNAČKA**	**POPIS**
ATS		Přenosové zařízení, komunikátor
TR		Transformátor TR 230/16 V
Z		Modul zdroje PS
AKU		Záložní akumulátor
WLV		Bezdrátový vysílač
WLP		Bezdrátový přijímač
KS		Klíčový spínač
ZE		Propouštěcí zámek
KL		Ovladač EZS (klávesnice)
PCO		Poplachové přijímací zařízení

Schématické značky dle ČSN EN 50131-1/Z1		
ZKR.	**ZNAČKA**	**POPIS**
VVM		Vstupně výstupní modul
REL		Reléový modul

Příloha 2 Trénovací soubor

	Det 1	0	1	0	0	0	0	0	0	0	0
Zona 1	Det2	0	0	1	0	0	0	0	0	0	0
	Det1	0	0	0	1	0	0	0	0	0	0
Zona 2	Det2	0	0	0	0	1	0	0	0	0	0
	Det1	0	0	0	0	0	1	0	0	0	0
Zona 3	Det2	0	0	0	0	0	0	1	0	0	0
	Det1	0	0	0	0	0	0	0	1	0	0
Zona 4	Det2	0	0	0	0	0	0	0	0	1	0
	Baterie	1	1	1	1	1	1	1	1	1	1
Napájení	Zdroj	1	1	1	1	1	1	1	1	1	1

Z1	Z2	Z3	Z4	1	
1	0	0	0	1	Vstup 1
0	0	0	0	1	Vstup 2
0.1	**0**	**0**	**0**	**0**	Target

Z1	Z2	Z3	Z4	2	
0	0	0	0	1	Vstup 1
1	0	0	0	1	Vstup 2
0.2	**0**	**0**	**0**	**0**	Target

Z1	Z2	Z3	Z4	3	
1	0	0	0	1	Vstup 1
1	0	0	0	1	Vstup 2
0.3	**0**	**0**	**0**	**0**	Target

Z1	Z2	Z3	Z4	4	
0	1	0	0	1	Vstup 1
0	0	0	0	1	Vstup 2
0	**0.01**	**0**	**0**	**0**	Target

Z1	Z2	Z3	Z4	5	
0	0	0	0	1	Vstup 1
0	1	0	0	1	Vstup 2
0	**0.02**	**0**	**0**	**0**	Target

Z1	Z2	Z3	Z4	6	
0	1	0	0	1	Vstup 1
0	1	0	0	1	Vstup 2
0	**0.03**	**0**	**0**	**0**	Target

Z1	Z2	Z3	Z4	7	
0	0	1	0	1	Vstup 1
0	0	0	0	1	Vstup 2
0	**0**	**0.001**	**0**	**0**	Target

Z1	Z2	Z3	Z4	8	
0	0	0	0	1	Vstup 1
0	0	1	0	1	Vstup 2
0	**0**	**0.002**	**0**	**0**	Target

Z1	Z2	Z3	Z4	9	
0	0	1	0	1	Vstup 1
0	0	1	0	1	Vstup 2
0	**0**	**0.003**	**0**	**0**	Target

Z1	Z2	Z3	Z4	10	
0	0	0	1	1	Vstup 1
0	0	0	0	1	Vstup 2
0	**0**	**0**	**0.0001**	**0**	Target

Z1	Z2	Z3	Z4	11	
0	0	0	0	1	Vstup 1
0	0	0	1	1	Vstup 2
0	**0**	**0**	**0.0002**	**0**	Target

Z1	Z2	Z3	Z4	12	
0	0	0	1	1	Vstup 1
0	0	0	1	1	Vstup 2
0	**0**	**0**	**0.0003**	**0**	Target

Z1	Z2	Z3	Z4	13		
0	0	0	0	0	Vstup 1	Trafo
0	0	0	0	1	Vstup 2	Bat
0	**0**	**0**	**0**	**0**	Target	

Z1	Z2	Z3	Z4	14		
0	0	0	0	1	Vstup 1	Trafo
0	0	0	0	0	Vstup 2	Bat
0	**0**	**0**	**0**	**0**	Target	

Z1	Z2	Z3	Z4	15		
0	0	0	0	0	Vstup 1	Trafo
0	0	0	0	0	Vstup 2	Bat
0	**0**	**0**	**0**	**1**	Target	