Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Vizualizace učení neuronové sítě

2017

Michal Ratajský

Vedoucí práce: Mgr. Petr Osička, Ph.D.

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Michal Ratajský
Název práce:	Vizualizace učení neuronové sítě
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	Mgr. Petr Osička, Ph.D.
Počet stran:	40
Přílohy:	1 DVD
Jazyk práce:	český

Bibliograhic info

Author:	Michal Ratajský
Title:	Neural Network Learning Visualization
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Applied Computer Science, combined form
Supervisor:	Mgr. Petr Osička, Ph.D.
Page count:	40
Supplements:	1 DVD
Thesis language:	Czech

Anotace

Umělé neuronové sítě patří mezi významné algoritmy z oblasti strojového učení. V rámci práce byly popsány vybrané modely umělých neuronových sítí a byla vytvořena aplikace pro vizualizaci jejich učení. Předpokládané využití aplikace je pro účely výuky.

Synopsis

Artificial neural networks play significant role among machine learning algorithms. This paper describes selected types of neural networks and for the purpose of illustrating their learning processes, a graphical desktop application was developed. The application is intended for educational use.

Klíčová slova: neuronová síť; perceptron; adaline; kohonenova samoorganizační mapa; radial basis function; c++; qt

Keywords: neural network; perceptron; adaline; kohonen self-organizing map; radial basis function; c++; qt

Děkuji Mgr. Petru Osičkovi, Ph.D. za vedení této práce a své rodině za podporu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Studium problematiky umělých neuronových sítí a obecně oborů umělé inteligence a strojového učení nabývá v posledních letech zcela zásadního významu. Pokrok ve vývoji teoretických modelů a dostupnost výkonných výpočetních prostředků pomáhají řešit problémy, které se až donedávna zdály být doménou výhradně lidského způsobu uvažování. Tím přirozeně stoupá atraktivita těchto oborů a poptávka po jejich studiu.

První umělé neuronové sítě začaly vznikat již ve 40. letech minulého století. Během následujících desetiletí pak vznikla řada modelů, které položily matematické základy a mnohé nalezly uplatnění i v praktických úlohách klasifikace, aproximace, získávání znalostí z dat, nebo v roli asociativních pamětí.

Vývoj neuronových sítí v novém století souvisí zejména se vznikem oboru *hlubokého učení* (*deep learning*). Význačným rysem novodobých modelů je však využití již dříve objevených konceptů, které tyto modely využívají a vhodně je rozšiřují. Osvojení původních konceptů je tedy nezbytné také pro pochopení podstaty fungování moderních *konvolučních* a *LSTM* sítí, se kterými se setkáváme například v oblastech rozpoznávání hlasu a strojového překladu [3].

Cílem této práce bylo vyvinout grafickou aplikaci pro vizualizaci učení vybraných typů umělých neuronových sítí. Základním požadavkem přitom byla možnost využití pro výukové účely, s čímž také souvisí zaměření na významné modely jednodušších neuronových sítí, konkrétně na jednovrstvé a vícevrstvé perceptronové sítě, model ADALINE, Kohonenovy samoorganizační mapy a sítě typu radial basis function.

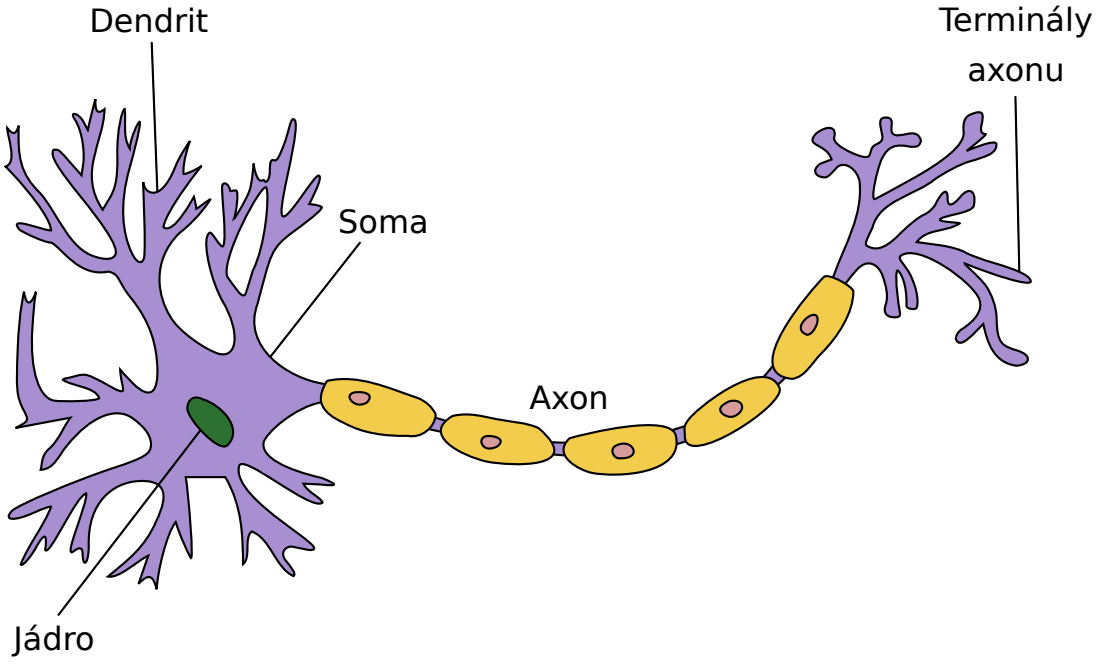
V úvodních částech práce je představena biologická inspirace, teoretická podstata fungování vybraných modelů a popsány jsou také nejběžnější učící algoritmy, které jsou v těchto modelech využívány.

V dalších kapitolách je popsán způsob fungování a ovládání vytvořené aplikace, technické aspekty vývoje, instalace a také možnosti budoucích rozšíření. Závěrečná kapitola je věnována dalším vizualizačním nástrojům.

2 Biologický pohled

Biologické nervové buňky, neboli *neurony*, jsou základní stavební jednotkou nervové soustavy živých organismů. Jejich úkolem je šíření vzruchů z receptorů k dalším nervovým buňkám a výkonným orgánům organismu [12]. Významné množství neuronů se nachází v mozku. Průměrný mužský mozek o váze 1,5 kg obsahuje 86 miliard neuronů [5].

Nervová buňka je svojí stavbou přizpůsobená k přenosu informací. K tělu buňky, tzv. *somatu*, jsou připojeny vstupní přenosové kanály *dendrity* a výstupní kanál *axon*, který může být přes své terminály připojen k dendritům jiných neuronů (viz obrázek 1). Spojení mezi neurony, kde dochází k přenosu informace, nazýváme *synapse*. Synapse disponují různou mírou propustnosti a tím mohou napomáhat nebo potlačovat šíření vzruchů. Právě míra propustnosti synapsí je nositelem všech význačných informací během života organismu [12].



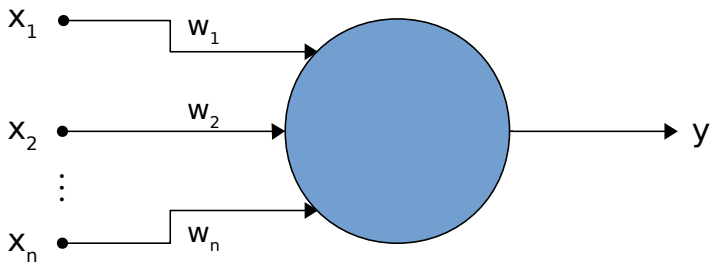
Obrázek 1: Biologický neuron [9]

3 Umělé neuronové sítě

Za počátek vzniku oboru umělých neuronových sítí je považována práce z roku 1943, ve které Warren McCulloch a Walter Pitts navrhli jednoduchý matematický model biologického neuronu [12].

Obecná struktura umělého neuronu, ze které vycházejí všechny dále popsané modely, je zachycena na obrázku 2. Umělý neuron má n obecně reálných vstupů x_1 až x_n, kterými jsou modelovány dendrity biologického neuronu. Vstupy

jsou ohodnoceny reálnými *synaptickými vahami* w_1 až w_n s excitačním nebo inhibičním charakterem. Výstup neuronu y je dán aplikací funkce příslušného modelu na vahami ohodnocené vstupy a může být, ve shodě s biologickou představou, propagován směrem k dalším neuronům.



Obrázek 2: Umělý neuron

3.1 Typy umělých neuronových sítí

Na základě biologické inspirace vznikla řada matematických modelů, lišících se architekturou, vnitřní funkcí neuronů, nebo metodami učení.

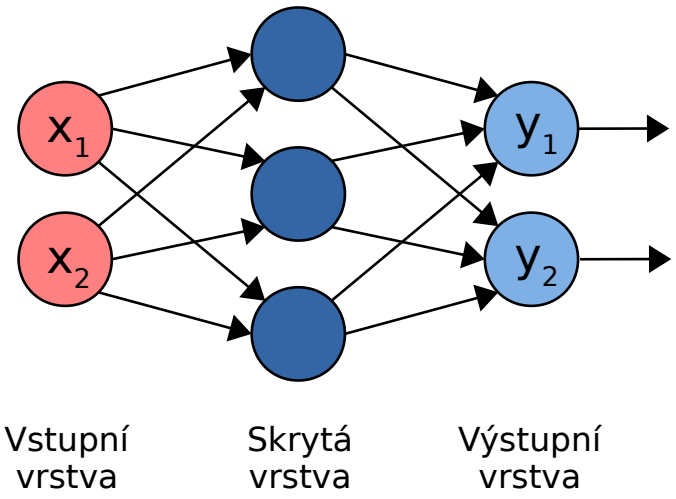
3.1.1 Dopředné sítě

Pro dopředné modely neuronových sítí je typická jejich vrstevnatost, tedy rozložení neuronů do několika oddělených *vrstev*, plnících různou funkci. V případě *úplně propojených* sítí je každý neuron propojen se všemi neurony v přímo sousedících vrstvách (viz obrázek 3). Vstupní hodnoty jsou přiváděny na neurony *vstupní* vrstvy, ze kterých je signál dále šířen v jednom směru. Model může být:

- *jednovrstvý* s jedinou vrstvou nevstupních neuronů, v němž jsou neurony vstupní vrstvy přímo propojeny s neurony *výstupní* vrstvy a ty určují výstup neuronové sítě;
- *vícevrstvý*, kde se mezi vstupní a výstupní vrstvou nachází jedna nebo více *skrytých* vrstev a signál je ze vstupu šířen postupně přes jednotlivé skryté vrstvy až k vrstvě výstupní.

3.1.2 Rekurentní sítě

V rekurentních sítích jsou některé neurony propojeny cyklicky. Oproti dopředným modelům se zde signál nešíří pouze jedním směrem, ale dochází i ke zpětnovazebnému přenosu informace. Rekurentní sítě jsou typicky modelovány tak, aby odezva sítě závisela nejen na aktuálních vstupních hodnotách, ale aby odrážela i vliv dříve předložených vstupů [17].



Obrázek 3: Úplně propojená dopředná neuronová síť

Mezi významné zástupce rekurentních sítí patří zejména *Hopfieldova síť* a sítě typu *LSTM* (*Long Short-Term Memory*), které se ukázaly být nesmírně úspěšné v úlohách zpracování obrazu, ručně psaného textu a strojového překladu [3].

3.1.3 Způsoby učení

Učení neuronových sítí spočívá v postupném předkládání vzorů z trénovací sady, na základě nichž jsou v síti adaptovány synaptické váhy. Různé modely využívají různé způsoby učení. Rozlišujeme:

- *učení s učitelem* (*supervised learning*): trénovací sada obsahuje ke každému vstupu sítě také požadovaný výstup (správnou odpověď), váhy jsou adaptovány tak, aby se minimalizoval rozdíl mezi vypočtenou hodnotou a hodnotou požadovanou;
- *učení bez učitele* (*unsupervised learning*): trénovací sada obsahuje pouze množinu vstupních dat, cílem je síť adaptovat na tuto množinu tak, aby na stejné, resp. podobné vzory reagovala stejným způsobem.

3.2 Perceptron

3.2.1 Jednoduchý perceptron

Jednoduchý perceptron je jednovrstvá dopředná neuronová síť, obsahující jediný neuron s binárním výstupem ve své výstupní vrstvě. Tento model vytvořil v roce 1957 Frank Rosenblatt a současně pro něj navrhl učící algoritmus [12]. Tento model je schopen klasifikovat pouze lineárně separovatelné množiny vstupů.

Vstupem perceptronu je n reálných hodnot x_1 až x_n, které jsou ohodnoceny synaptickými vahami w_1 až w_n. Vážený součet vstupních hodnot tvoří *vnitřní potenciál* neuronu. Dle biologické představy budeme chtít, aby byl neuron aktivní v případě dosažení určité prahové hodnoty b. K tomu využijeme *aktivační funkci* $\varphi : \mathbb{R} \to \{0, 1\}$, kterou aplikujeme na vnitřní potenciál a tím získáme binární výstupní hodnotu perceptronu:

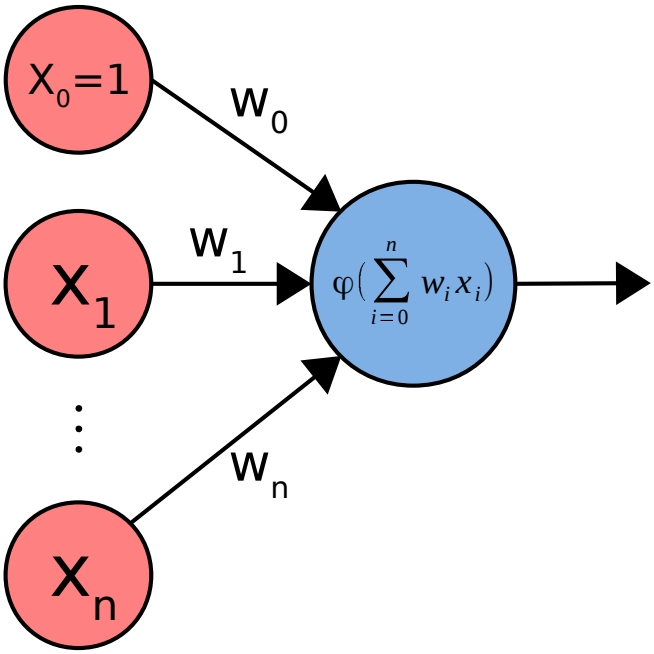
$$y = \varphi\left(-b + \sum_{i=1}^{n} w_i x_i\right) .$$

Aktivační funkce $\varphi(x)$ má tvar *ostré nelinearity*. Přičtením záporné prahové hodnoty $-b$ před aplikací funkce můžeme funkci definovat s nulovým prahem:

$$\varphi(x) = \begin{cases} 1 & \text{pro } x \geq 0 \\ 0 & \text{pro } x < 0 \end{cases} .$$

Pro zjednodušení vztahu pro výpočet výstupu je vhodné zavést fixní vstupní hodnotu $x_0 = 1$ a váhu $w_0 = -b$, kterou nazveme *bias*. Vztah můžeme poté zapsat (viz také obrázek 4):

$$y = \varphi\left(\sum_{i=0}^{n} w_i x_i\right) .$$



Obrázek 4: Jednoduchý perceptron s n vstupními neurony

3.2.2 Učení sítě

Před začátkem učení zvolíme počáteční hodnoty synaptických vah. V praxi typicky nastavujeme váhy náhodně, např. na hodnoty blízké nule [12]:

$$w_i \in [-1, 1] \quad i = 0, \ldots, n \ .$$

V každém učícím kroku nejprve vybereme trénovací vzor z trénovací množiny a nastavíme hodnoty vstupních neuronů dle vybraného vzoru. Výstup sítě y pro daný vzor získáme dle uvedeného vzorce. Následně můžeme provést adaptaci synaptických vah pro přiblížení k požadovanému výstupu t:

$$w_i = w_i - \alpha(y - t) \quad i = 0, \ldots, n \ ,$$

kde $\alpha \in (0, 1]$ je *koeficient učení*, určující míru adaptace vah. Koeficient má vliv na rychlost učení a pro dosažení optimálních výsledků je vhodné jej během učení upravovat. Je zřejmé, že pokud výstup y odpovídá požadovanému výstupu t, bude v daném kroku zachováno původní nastavení vah.

Pokud je množina trénovacích vzorů lineárně separovatelná dle požadované výstupní hodnoty t, učící algoritmus konverguje po konečně mnoha krocích, tedy nezávisle na počátečním nastavení synaptických vah [10, str. 88].

3.2.3 Jednovrstvá síť perceptronů

V jednovrstvé síti perceptronů se ve výstupní vrstvě nachází m výstupních neuronů. Tato neuronová síť je úplně propojená, tedy každý z n vstupních neuronů je vstupem každého z m výstupních neuronů. Synaptická váha w_{ij} označuje váhu synaptického spojení z i-tého vstupního neuronu do j-tého výstupního neuronu. Váha w_{0j} označuje bias j-tého výstupního neuronu. Výstup j-tého neuronu definujeme vztahem:

$$y_j = \varphi\left(\sum_{i=0}^{n} w_{ij} x_i\right) \quad j = 1, \ldots, m \ .$$

Učení jednovrstvé perceptronové sítě odpovídá učení jednoduchého perceptronu, aplikované zvlášť na každý z m výstupních neuronů. Trénovací vzory musí obsahovat požadované výstupy t_1 až t_m pro všechny výstupní neurony. Adaptace synaptické váhy w_{ij} je tedy dána vztahem:

$$w_{ij} = w_{ij} - \alpha(y_j - t_j) \quad i = 0, \ldots, n; \ j = 1, \ldots, m \ .$$

3.3 ADALINE

Model ADALINE (Adaptive Linear Element) vyvinul Bernard Widrow se svými studenty krátce po objevu perceptronu [12]. Tento model svojí strukturou odpovídá modelu perceptronu, má tedy n reálných vstupů a výstupní hodnotu y ovlivňují bias w_0 a hodnoty synaptických vah w_1 až w_n. Oproti jednoduchému

perceptronu zde však není uplatňována skoková aktivační funkce, ale výstup tvoří přímo lineární kombinace vstupních hodnot:

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

Výstupy jsou tedy obecně reálné. ADALINE je však, stejně jako jednoduchý perceptron, binárním klasifikátorem. Uvažujme vstupní hodnoty x_1 až x_n, tedy bod $[x_1, \ldots, x_n]$ v n-rozměrném prostoru. Nadrovina s koeficienty w_1 až w_n daná rovnicí

$$w_0 + \sum_{i=1}^{n} w_i x_i = 0$$

rozděluje tento prostor na dva poloprostory, ve kterých má hodnota y opačné znaménko [12, str. 65].

Při učení neuronu typu ADALINE budeme minimalizovat chybu sítě, kterou vzhledem k aktuálnímu trénovacímu vzoru můžeme vyjádřit vztahem:

$$E = \frac{1}{2}(y - t)^2 \ .$$

Adaptaci vah provedeme gradientní metodou [12, str. 67]:

$$w_i = w_i - \alpha \frac{\partial E}{\partial w_i} = w_i - \alpha(y - t)x_i \quad i = 0, \ldots, n \ ,$$

kde α je koeficient učení. Úpravu vah provádíme postupně pro každý předložený trénovací vzor a proces učení lze ukončit po dosažení požadované úrovně chyby sítě. Toto učící pravidlo je často nazýváno *Least Mean Square* (LMS) [4, str. 150] nebo delta pravidlo [17, str. 8].

3.4 Vícevrstvý perceptron

Rozšířením jednovrstvé perceptronové sítě o jednu nebo více skrytých vrstev a vhodnou úpravou učícího pravidla získáme vícevrstvý perceptron, který je schopen klasifikovat prvky i lineárně neseparovatelných množin.

Pro použití učícího algoritmu *backpropagation* (viz dále) musí být aktivační funkce $\varphi(x)$ spojitá, diferencovatelná a monotónně neklesající [16]. Nejčastěji využívanými funkcemi jsou standardní (logistická) sigmoida a hyperbolický tangens [10, 16]:

$$\varphi(x) = \frac{1}{1 + e^{-x}} \quad x \in \mathbb{R} \ ,$$

$$\varphi(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad x \in \mathbb{R} \ .$$

Hodnotu nevstupního neuronu získáme, stejně jako v případě jednovrstvé sítě, aplikací aktivační funkce na vážený součet hodnot neuronů v předchozí vrstvě.

Předpokládejme vícevrstvou síť s jednou skrytou vrstvou s l neurony. Označme h_k^p vnitřní potenciál k-tého skrytého neuronu a h_k jeho výstupní hodnotu. Platí:

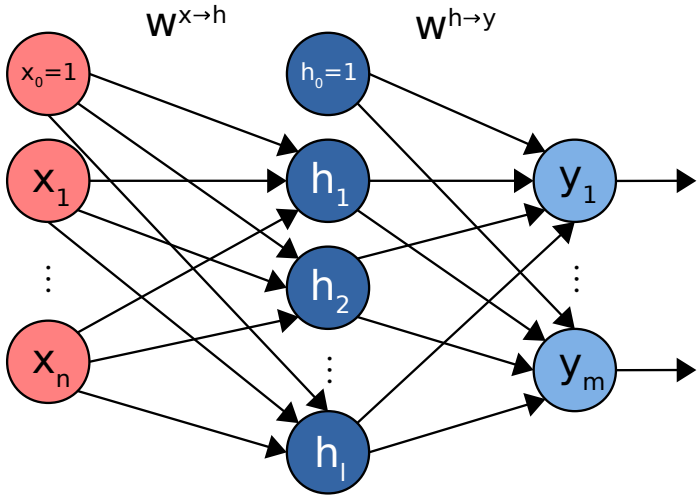
$$h_k^p = \sum_{i=0}^{n} w_{ik}^{x \to h} x_i \quad k = 1, \ldots, l \ ,$$

$$h_k = \varphi\left(h_k^p\right) \quad k = 1, \ldots, l \ ,$$

kde synaptická váha $w_{ik}^{x \to h}$ je váha mezi i-tým neuronem vstupní vrstvy a k-tým neuronem skryté vrstvy, $x_0 = 1$ a $w_{0k}^{x \to h}$ je bias k-tého neuronu. Způsob značení je také znázorněn na obrázku 5. Obdobně platí pro výstupní vrstvu:

$$y_j^p = \sum_{k=0}^{l} w_{kj}^{h \to y} h_k \quad j = 1, \ldots, m \ ,$$

$$y_j = \varphi\left(y_j^p\right) \quad j = 1, \ldots, m \ .$$



Obrázek 5: Vícevrstvý perceptron s jednou skrytou vrstvou

3.4.1 Učení sítě

Základním algoritmem pro učení vícevrstvého perceptronu je *backpropagation*, neboli algoritmus zpětného šíření chyby. Učení neuronové sítě probíhá ve třech fázích:

1. dopředné šíření trénovacího vzoru (*feed-forward*),
2. zpětné šíření chyby (*backpropagation*),
3. adaptace synaptických vah pro minimalizaci chyby.

Během dopředného šíření vzoru postupně počítáme hodnoty neuronů skrytých vrstev a výstupní vrstvy dle uvedených vztahů. Pro chybovou funkci sítě vzhledem k aktuálnímu vzoru platí [16, str. 46]:

$$E = \frac{1}{2}\sum_{j=1}^{m}(y_j - t_j)^2 \ .$$

Adaptací vah budeme obecně rozumět jejich změnu o hodnotu Δw_i:

$$w_i = w_i + \Delta w_i \ .$$

Pro minimalizaci chyby použijeme, podobně jako v případě modelu ADALINE, gradientní metodu [12, str. 54]:

$$\Delta w_i = -\alpha \frac{\partial E}{\partial w_i} \ ,$$

kde α je koeficient učení. Pro synaptické váhy mezi neurony skryté a výstupní vrstvy platí [12, str. 56]:

$$\frac{\partial E}{\partial w_{kj}^{h\to y}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial y_j^p}\frac{\partial y_j^p}{\partial w_{kj}^{h\to y}} \quad k = 0, \ldots, l;\ j = 1, \ldots, m \ ,$$

$$\frac{\partial E}{\partial y_j} = y_j - t_j \ ,$$

$$\frac{\partial y_j}{\partial y_j^p} = y_j(1 - y_j) \ ,$$

$$\frac{\partial y_j^p}{\partial w_{kj}^{h\to y}} = h_k \ ,$$

kde předpokládáme, že $\varphi(x)$ je standardní sigmoida a tedy

$$\varphi'(x) = \varphi(x)[1 - \varphi(x)] \ .$$

Dále je vhodné zavést chybu j-tého výstupního neuronu δ_j^y [10, str. 168]:

$$\delta_j^y = (y_j - t_j)y_j(1 - y_j) \quad j = 1, \ldots, m \ .$$

Ve vzorci pro výpočet změny synaptických vah později užijeme vypočtený vztah

$$\frac{\partial E}{\partial w_{kj}^{h\to y}} = (y_j - t_j)y_j(1 - y_j)h_k = \delta_j^y h_k \ .$$

Obdobně budeme postupovat v případě synaptických vah $w_{ik}^{x\to h}$ mezi neurony vstupní a skryté vrstvy:

$$\frac{\partial E}{\partial w_{ik}^{x\to h}} = \frac{\partial E}{\partial h_k}\frac{\partial h_k}{\partial h_k^p}\frac{\partial h_k^p}{\partial w_{ik}^{x\to h}} \quad i = 0, \ldots, n;\ k = 1, \ldots, l \ .$$

Výpočet parciální derivace $\frac{\partial E}{\partial h_k}$ pro skrytý neuron h_k převedeme na výpočet parciálních derivací $\frac{\partial E}{\partial y_j}$ pro výstupní neurony y_j, které jsou synapticky spojeny s neuronem h_k [12, str. 57]:

$$\frac{\partial E}{\partial h_k} = \sum_{j=1}^{m} \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial y_j^p} \frac{\partial y_j^p}{\partial h_k} = \sum_{j=1}^{m} \delta_j^y w_{kj}^{h \to y} \ .$$

Pro další parciální derivace v součinu platí:

$$\frac{\partial h_k}{\partial h_k^p} = h_k(1 - h_k) \ ,$$

$$\frac{\partial h_k^p}{\partial w_{ik}^{x \to h}} = x_i \ .$$

Pro neurony skryté vrstvy zavedeme chybu k-tého neuronu δ_k^h, kterou později použijeme při adaptaci vah:

$$\delta_k^h = \left(\sum_{j=1}^{m} \delta_j^y w_{kj}^{h \to y} \right) h_k(1 - h_k) \quad k = 1, \ldots, l \ ,$$

$$\frac{\partial E}{\partial w_{ik}^{x \to h}} = \left(\sum_{j=1}^{m} \delta_j^y w_{kj}^{h \to y} \right) h_k(1 - h_k) x_i = \delta_k^h x_i \ .$$

Posledním krokem je adaptace synaptických vah v celé neuronové síti. Jelikož výpočty δ_k^h byly ovlivněny hodnotami $w_{kj}^{h \to y}$, je adaptace vyčleněna do samostatného závěrečného kroku.

$$w_{kj}^{h \to y} = w_{kj}^{h \to y} - \alpha \delta_j^y y_j \quad k = 0, \ldots, l; \ j = 1, \ldots, m \ ,$$

$$w_{ik}^{x \to h} = w_{ik}^{x \to h} - \alpha \delta_k^h h_k \quad i = 0, \ldots, n; \ k = 1, \ldots, l \ .$$

Učení je ukončeno po dosažení požadované chyby nebo pokud již nedochází k významnému snižování chyby. Minimalizace chybové funkce gradientní metodou je úkolem nalezení lokálního minima funkce, které však nemusí být minimem globálním (pro více informací viz např. [12, str. 55]). Na výsledek učení má vliv řada faktorů, včetně počátečního nastavení vah, volby koeficientu učení a pořadí výběru trénovacích vzorů.

3.5 Kohonenova samoorganizační mapa

Kohonenova samoorganizační mapa je neuronová síť, která byla poprvé popsána v roce 1982 a patří mezi úplně propojené dopředné sítě s jednou vstupní a jednou výstupní vrstvou, jejíž učení probíhá bez učitele.

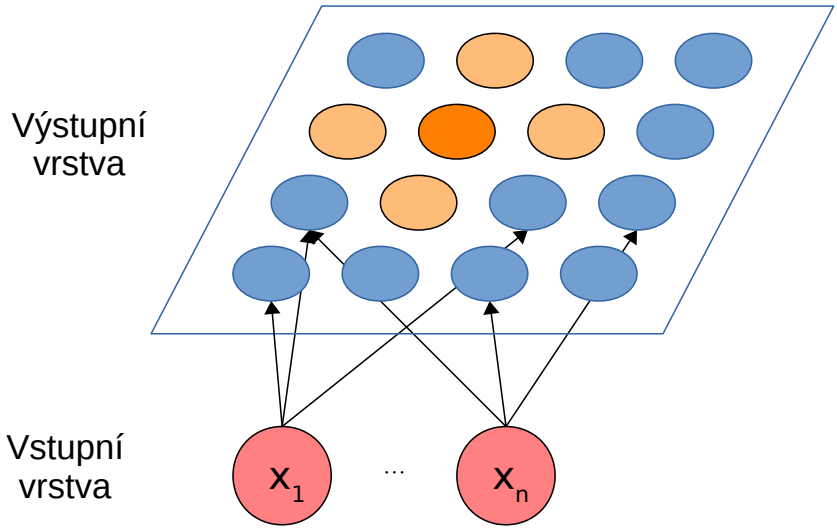
Síť obsahuje ve své vstupní vrstvě n vstupních neuronů x_1 až x_n. Neurony ve výstupní vrstvě jsou uspořádány do určité topologické struktury, nejčastěji do dvourozměrné mřížky (obrázek 6), nebo jednorozměrné řady jednotek [12].

Při vyhodnocování výstupu na základě předložených vstupních hodnot hledáme ve výstupní vrstvě neuron, který je nejvíce podobný vstupním hodnotám. Pro každý z m výstupních neuronů tedy provádíme výpočet:

$$D(j) = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})^2} \quad j = 1, \ldots, m \ ,$$

kde w_{ij} je váha synaptického spojení mezi i-tým vstupním a j-tým výstupním neuronem. Neuron s nejnižší hodnotou $D(j)$ nazveme *vítězný neuron* nebo také *Best Matching Unit*. Pro hodnotu j-tého výstupního neuronu platí [12]:

$$y_j = \begin{cases} 1 & \text{pro } j = \arg\min_k D(k) \\ 0 & \text{jinak} \end{cases} \ .$$



Obrázek 6: Kohonenova mapa s vyznačeným vítězným neuronem a jeho okolím

3.5.1 Učení sítě

Při učení Kohonenovy mapy je pro daný tréninkový vzor nejdříve nalezen vítězný neuron. Dále dojde k adaptaci vah tohoto neuronu tak, aby se neuron svými vahami přiblížil vstupním hodnotám. Provádí se však adaptace nejen vítězného neuronu, ale také neuronů v jeho okolí, kde vliv okolí je pro vítězný neuron c a j-tý výstupní neuron definován *funkcí sousedství* $h_c(j)$. Velikost okolí přitom není konstantní, ale obvykle se během učení zmenšuje tak, že na konci učení do okolí patří pouze vítězný neuron [12]. Vítězný neuron s vyznačeným okolím je znázorněn na obrázku 6. Pro adaptaci synaptických vah platí vztah:

$$w_{ij} = w_{ij} + \alpha h_c(j)(x_i - w_{ij}) \quad i = 1, \ldots, n; \ j = 1, \ldots, m \ ,$$

kde α je koeficient učení. Funkci sousedství $h_c(j)$ můžeme volit různými způsoby. Vhodnou volbou jsou funkce se spojitým přechodem mezi nulovými a nenulovými hodnotami, což odpovídá biologickým interakcím. Typicky používanou funkcí je Gaussova funkce [12]:

$$h_c(j) = \exp\left(-\frac{d_c(j)^2}{\sigma^2}\right) \quad j = 1, \ldots, m \ ,$$

kde funkce $d_c(j)$ vyjadřuje vzdálenost j-tého neuronu od vítězného neuronu, která je obecně dána topologickou strukturou. Ve zmíněném případě dvourozměrné mřížky je vhodnou volbou euklidovská vzdálenost. Hodnota σ udává šířku okolí, kterou je vhodné v průběhu učení zmenšovat.

3.6 Síť typu radial basis function

Neuronová síť typu radial basis function (RBF) je úplně propojenou dopřednou sítí, obsahující vstupní, skrytou a výstupní vrstvu.

Vstupní vrstva obsahuje n vstupních neuronů x_1 až x_n. Skrytou vrstvu nazveme *RBF vrstvou* a v ní obsažené neurony h_1 až h_l *RBF jednotky*. RBF jednotky realizují *radiální funkce*. Radiální funkci si lze představit jako funkci, určenou významným bodem (středem), která pro argumenty se stejnou vzdáleností od daného středu dává stejné funkční hodnoty. Měříme-li vzdálenost pomocí euklidovské metriky a uvažujeme-li dvourozměrný vstupní prostor, pak množiny se stejnou funkční hodnotou tvoří kružnice [12].

Vnitřní potenciál h_k^p k-té RBF jednotky je určen vzdáleností vstupu od středu RBF jednotky:

$$h_k^p = \sqrt{\sum_{i=1}^{n} (x_i - c_i)^2} \quad k = 1, \ldots, l \ ,$$

kde c_i jsou hodnoty synaptických vah mezi vstupní a RBF vrstvou (obrázek 7), které určují střed RBF jednotky. Výstupní hodnotu h_k získáme aplikací aktivační funkce $\varphi(x)$:

$$h_k = \varphi\left(h_k^p\right) \quad k = 1, \ldots, l \ .$$

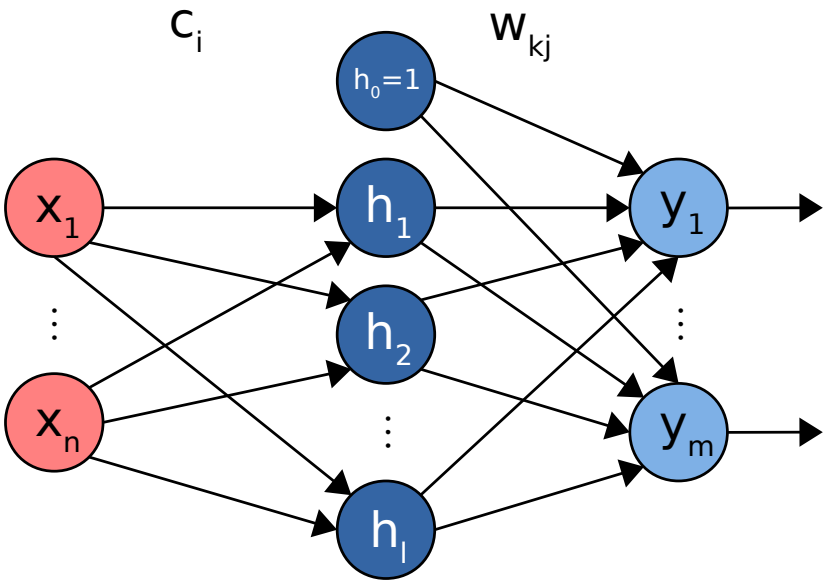
Aktivační funkce je typicky realizována pomocí Gaussovy funkce:

$$\varphi(x) = \exp\left(-\frac{x^2}{\sigma^2}\right) \quad x \in \mathbb{R}, \sigma > 0 \ .$$

Parametr σ je šířka, určující dosah radiální oblasti kolem středu RBF jednotky a je běžně určen během učení sítě. Výstupní vrstva RBF sítě je lineární:

$$y_j = \sum_{k=0}^{l} w_{kj} h_k \quad j = 1, \ldots, m \ ,$$

kde w_{kj} je synaptická váha mezi k-tou RBF jednotkou a j-tým výstupním neuronem, $h_0 = 1$ a w_{0j} je bias j-tého neuronu.



Obrázek 7: Síť typu radial basis function

3.6.1 Učení sítě

V první fázi učení RBF sítě probíhá volba středů RBF jednotek, neboli adaptace synaptických vah c_i. Tato fáze probíhá učením bez učitele. Úkolem rozmístění RBF jednotek je aproximace výskytu trénovacích vzorů, k čemuž můžeme využít např. Kohonenovu samoorganizační mapu nebo algoritmus *k-means* [12].

Druhá fáze učení nastavuje další parametry RBF jednotek. Obvykle jednotkám přiřazujeme parametr, určující dosah, ve kterém mají jednotky relevantní výstup. Ve výše uvedeném případě Gaussovy aktivační funkce je jím šířka σ. Šířka se často nastavuje úměrně průměru euklidovských vzdáleností q nejbližších sousedů dané jednotky. Často se přitom volí $q = 1$ [12, str. 123-126].

Třetí fáze je učení s učitelem, během nějž jsou adaptovány váhy mezi skrytou a výstupní vrstvou. Postup adaptace je stejný jako v případě vícevrstvého perceptronu, nedochází zde však k propagaci do nižších vrstev:

$$w_{kj} = w_{kj} - \alpha(y_j - t_j)h_k \quad k = 0, \ldots, l;\ j = 1, \ldots, m\ .$$

4 Vizualizace učení neuronových sítí

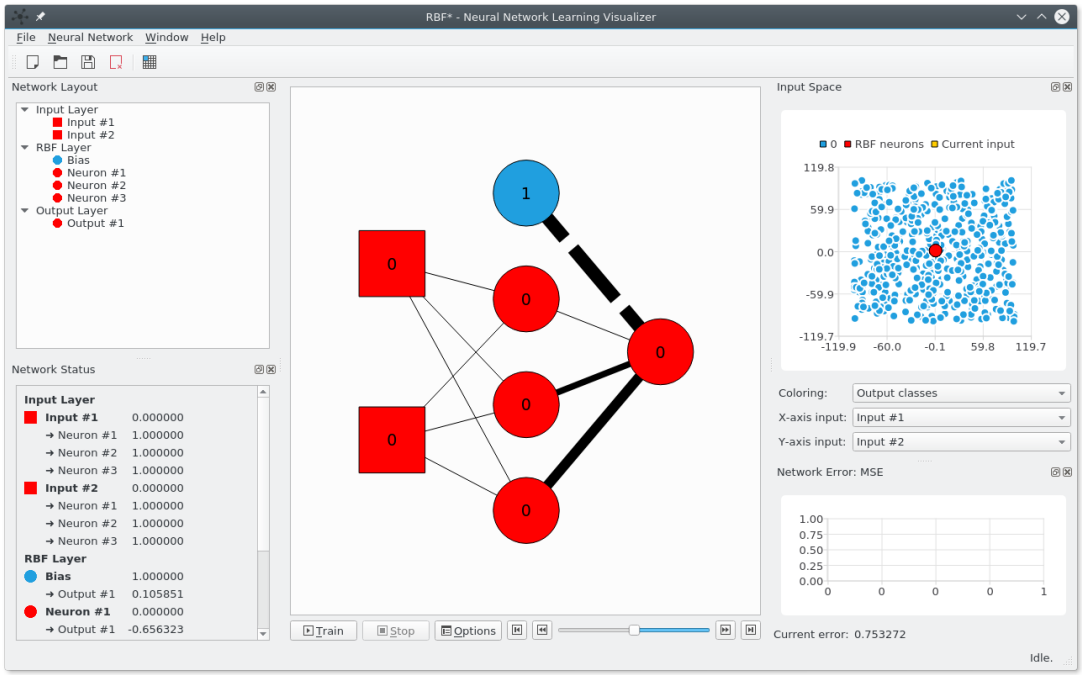
Pro účely vizualizace učení neuronových sítí byla vytvořena multiplatformní grafická aplikace *Neural Network Learning Visualizer*. Aplikace umožňuje vizualizaci učení perceptronových sítí, modelu ADALINE, Kohonenových map a sítí typu radial basis function.

4.1 Vzhled aplikace

Primární okno aplikace obsahuje centrální část, ve které je zobrazen diagram sítě s nástroji pro ovládání procesu učení. Podoba diagramu neuronové sítě obecně odpovídá dříve uvedeným nákresům (např. na obrázku 4). Kromě intuitivně použitelných tlačítek pro zahájení a zastavení učení obsahuje také vodorovný posuvník, který posunem doprava zmenšuje prodlevu mezi iteracemi učícího algoritmu a tím zrychluje průběh učení. Pomocí tlačítka *Options* lze vyvolat dialog pro nastavení podmínek pro automatické zastavení učení a úpravu nastavení učících parametrů, které byly zvoleny během vytváření sítě.

Levá a pravá strana primárního okna poskytují dokovací oblasti, do kterých jsou (v závislosti na typu sítě) umisťovány vizualizační panely a doplňkové nástroje. Panely lze v případě potřeby z dokovací oblasti odstranit, změnit jim umístění, velikost a proporce, vyčlenit je do samostatných oken, nebo je sjednotit pod společnou komponentu se záložkovými kartami. Odstraněné panely lze vyvolat zpět kliknutím pravým tlačítkem myši do oblasti hlavní nabídky nebo hlavičky libovolného jiného panelu.

Primární okno také obsahuje panel nástrojů a ve spodní části okna stavový řádek, na jehož pravé straně jsou zobrazovány informace o průběhu učení. Typická podoba okna je znázorněna na obrázku 8.



Obrázek 8: Primární okno aplikace

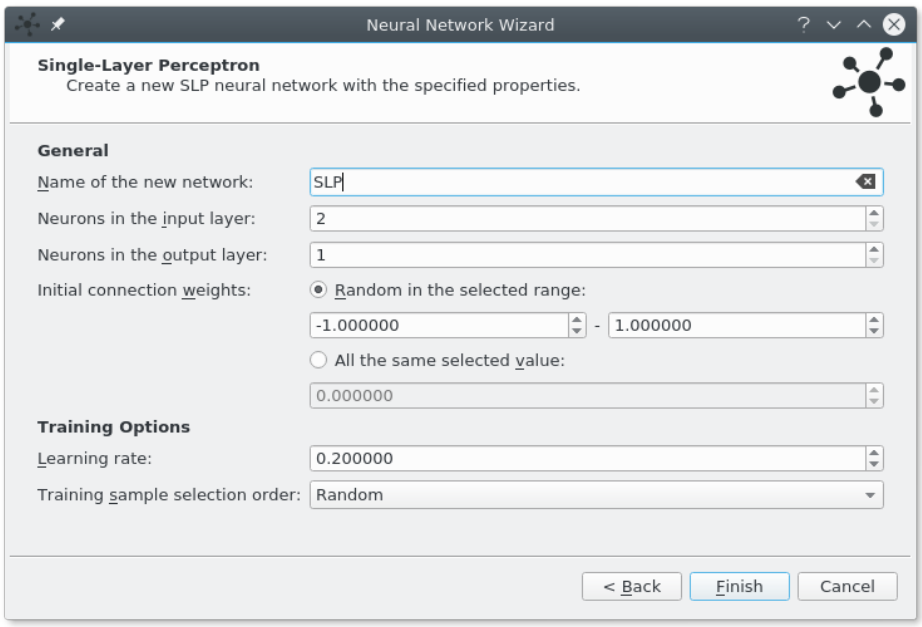
4.2 Vytvoření sítě

Po spuštění programu je zobrazen průvodce, který vybízí k vytvoření nové sítě, resp. k načtení sítě existující. Pro každý podporovaný typ sítě je k dispozici několik hotových příkladů s předvolenými parametry a trénovacími sadami.

4.3 Perceptron

Jednovrstvá perceptronová síť je v aplikaci označována termínem *Single-Layer Perceptron*. Tímto modelem lze vyjádřit jednoduchý perceptron i síť perceptronů. Průvodce vytvořením sítě (obrázek 9) umožňuje pro novou síť zadat následující možnosti přizpůsobení:

- *Name of the network*: libovolné pojmenování sítě;
- *Neurons in the input layer*: počet neuronů ve vstupní vrstvě;
- *Neurons in the output layer*: počet neuronů ve výstupní vrstvě;
- *Initial connection weights*: počáteční nastavení synaptických vah, jejichž hodnoty mohou být vybrány náhodně ze zvoleného intervalu, nebo může být všem vahám přiřazena zadaná hodnota;
- *Learning rate*: koeficient učení;
- *Training sample selection order*: volba metody výběru trénovacích vzorů.

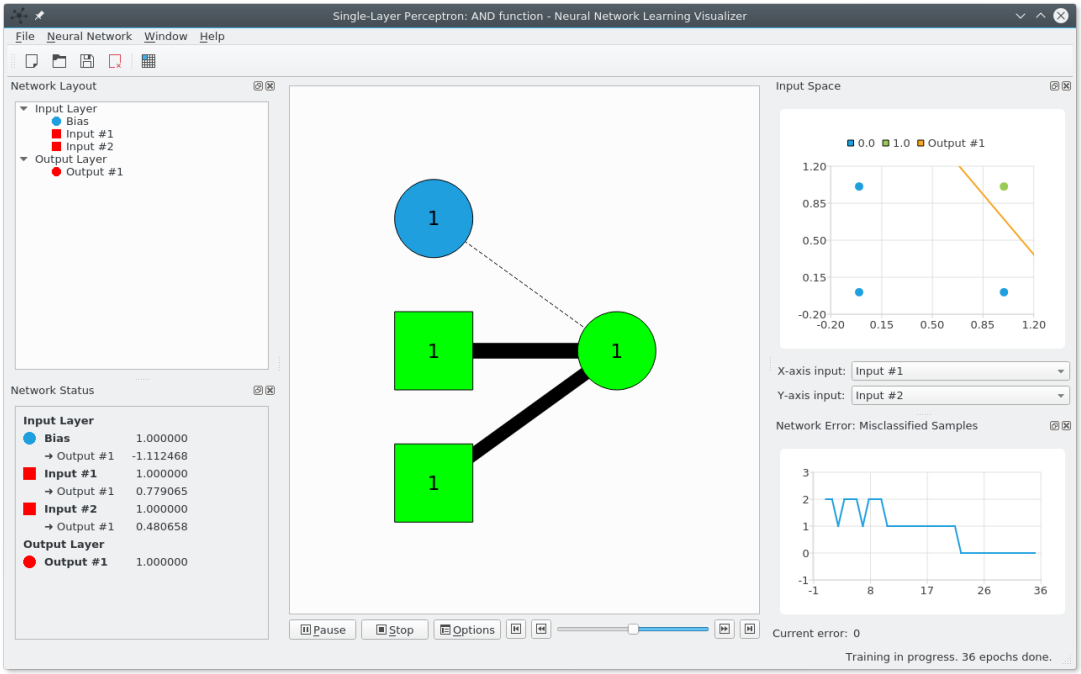


Obrázek 9: Průvodce vytvořením nové perceptronové sítě

Pro odlišení vstupních a nevstupních neuronů je pro zobrazení vstupních neuronů v diagramu použit tvar čtverce. Neurony znázorňují svůj aktuální stav pomocí barvy pozadí a vepsané aktuální hodnoty. Nízké hodnoty jsou podbarveny červeně, s rostoucími hodnotami barva přechází od HSV tónu 0 až k tónu 120[1], tedy přes žlutou k zelené. Pro bias je použito modré podbarvení. Vztahy mezi hodnotami synaptických vah jsou znázorněny pomocí různých tlouštěk čar. Tenká čára odpovídá nízké hodnotě váhy, tlustá čára hodnotě vysoké. Výpočet tloušťky čar je prováděn vždy v kontextu aktuální mezivrstvy.

Vizualizace perceptronové sítě je znázorněna na obrázku 10. Standardně jsou zobrazeny následující vizualizační panely:

- *Network Status*: aktuální stav sítě ve formě číselných hodnot neuronů a synaptických vah mezi nimi, včetně informace o změně hodnoty oproti předchozí iteraci učícího algoritmu;
- *Input Space*: dvourozměrné zobrazení vstupního prostoru, ve kterém jsou vyznačeny trénovací vzory a přímka, kterou perceptron rozděluje rovinu na různě klasifikované poloroviny;
- *Network Error*: graf vývoje chyby sítě, kde vzhledem k binárnímu charakteru výstupních neuronů je chyba vyjádřena počtem nesprávně klasifikovaných vzorů.



Obrázek 10: Vizualizace perceptronu pro logickou funkci AND

[1] viz např. http://doc.qt.io/qt-5/qcolor.html#the-hsv-color-model

4.4 ADALINE

Vizualizace modelu ADALINE je takřka identická jako vizualizace jednovrstvého perceptronu. Chyba sítě je pro tento model vyjádřena pomocí střední kvadratické chyby přes všech q trénovacích vzorů:

$$E = \frac{1}{q} \sum_{p=1}^{q} \sum_{j=1}^{m} \left(y_j^{(p)} - t_j^{(p)} \right)^2 ,$$

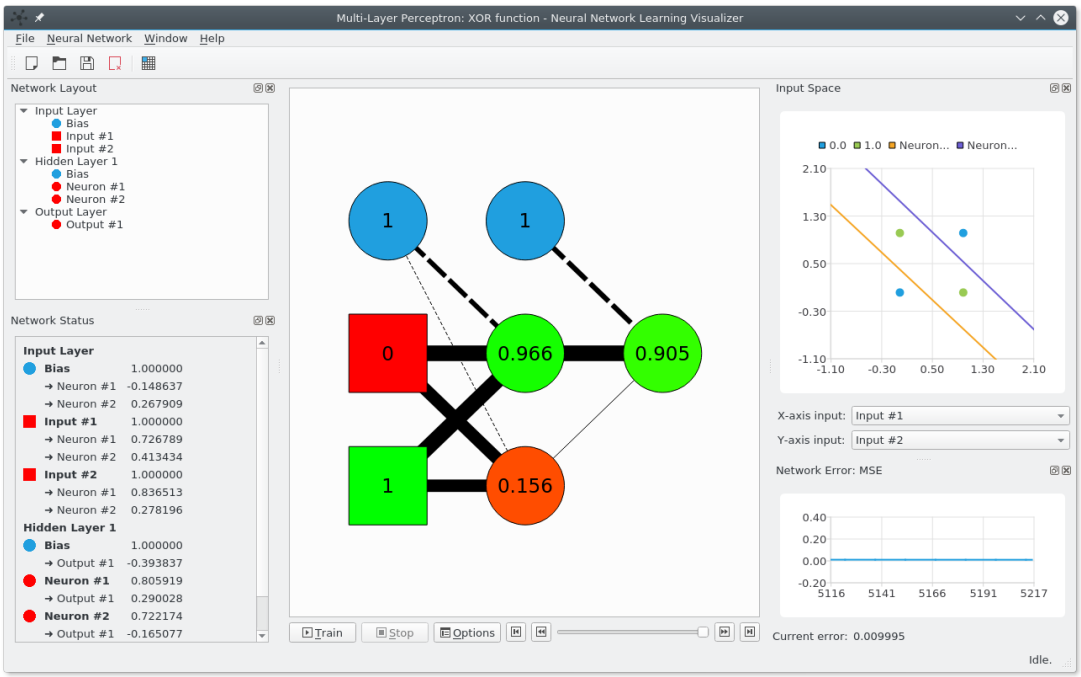
kde $y_j^{(p)}$ je hodnota j-tého výstupního neuronu pro p-tý předložený tréninkový vzor a $t_j^{(p)}$ je požadovaný výstup.

4.5 Vícevrstvý perceptron

Vizualizace vícevrstvého perceptronu (obrázek 11) je rozšířenou variantou vizualizace jednovrstvé sítě. Mimo již zmíněných možností obsahuje průvodce vytvořením sítě následující volby:

- *Number of hidden layers*: počet skrytých vrstev, s možností volby počtu neuronů v příslušných vrstvách;
- *Activation function*: volba aktivační funkce pro všechny nevstupní neurony.

Chyba sítě je i v případě tohoto modelu vyjádřena ve formě střední kvadratické chyby.

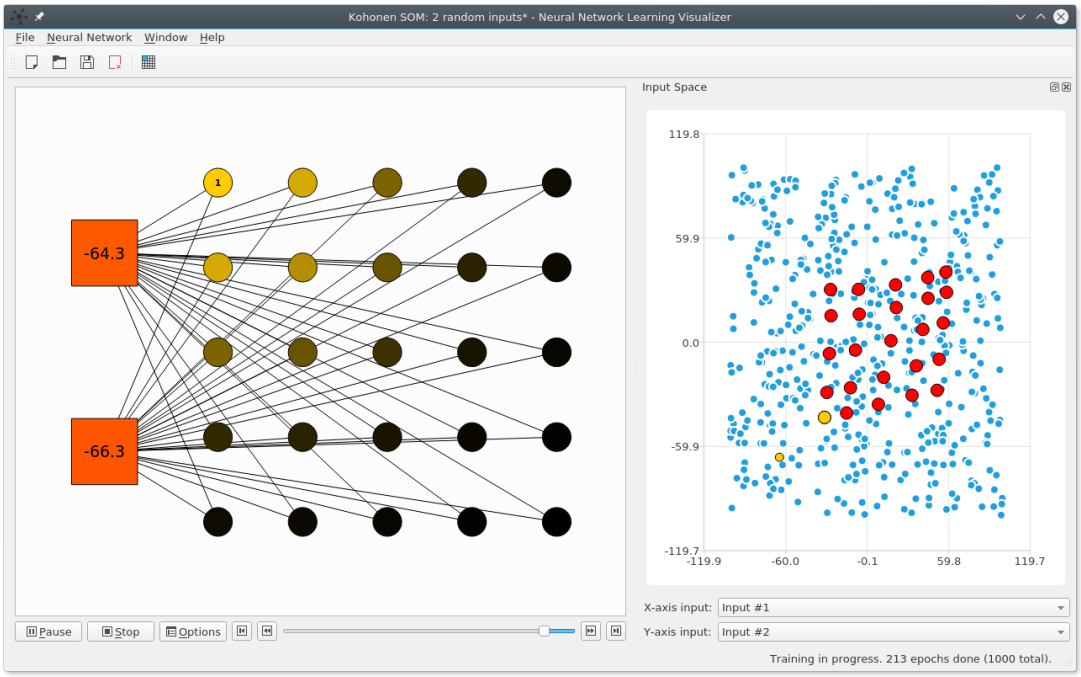


Obrázek 11: Vizualizace vícevrstvého perceptronu pro logickou funkci XOR

4.6 Kohonenova samoorganizační mapa

Vizualizace Kohonenovy mapy je znázorněna na obrázku 12. Výstupní vrstva je v diagramu zobrazena ve formě dvourozměrné mřížky velikosti $n \times n$ neuronů. Při vytváření nové Kohonenovy sítě je volen počet iterací učícího algoritmu. Tuto volbu je nutno provést před zahájením učení, jelikož je hodnota využívána pro vyčíslení Gaussovy funkce sousedství a má tedy vliv na postupný pokles míry vlivu trénovacích vzorů na okolní neurony. V diagramu Kohonenovy mapy jsou výstupní neurony implicitně podbarveny černě. Při výpočtu funkce sousedství jsou hodnoty této funkce znázorněny žlutou barvou, která s klesající hodnotou postupně přechází v černou. Neuron s vepsanou číslicí 1 je aktuální vítězný neuron. Aktuální hodnoty funkce sousedství výstupních neuronů jsou také zobrazovány v číselné podobě v panelu *Network Status* (tento panel je znázorněn na obrázcích 10 a 11). Podbarvení neuronů vstupní vrstvy na levé straně diagramu je totožné jako v případě perceptronových sítí.

Charakteristickou vlastností tohoto typu sítě je adaptace mřížky výstupních neuronů na vstupní prostor v závislosti na jeho pokrytí trénovacími vzory. Pro účely vizualizace této adaptace je v panelu *Input Space* znázorněna poloha všech trénovacích vzorů a na základě svých synaptických vah také poloha výstupních neuronů. V každé učící iteraci je v grafu žlutou barvou vyznačen právě vybraný trénovací vzor a vítězný výstupní neuron. Na tomto grafu lze také pozorovat typický přechod mezi dvěma učícími fázemi, kdy po relativně rychlém rozmístění jednotek ve vstupním prostoru dochází ke konci učení již pouze k méně významným lokálním změnám [12, str. 110].



Obrázek 12: Vizualizace Kohonenovy samoorganizační mapy

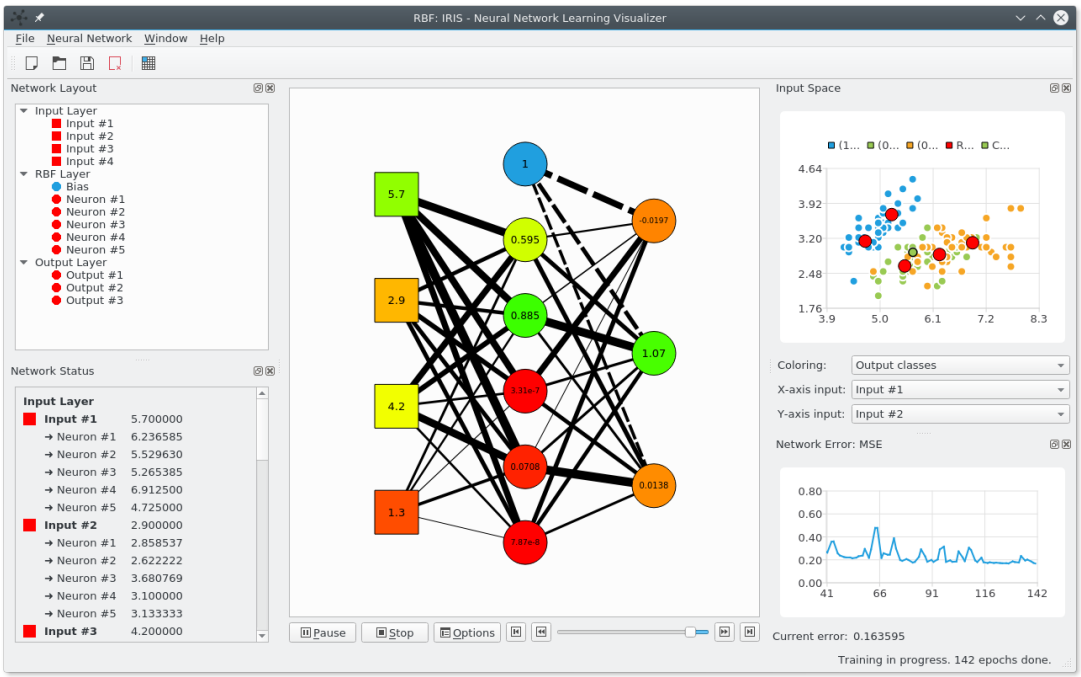
4.7 Síť typu radial basis function

Při vytváření sítě typu radial basis function je oproti jednovrstvé perceptronové síti nutno uvést počet neuronů (RBF jednotek) ve skryté vrstvě. Počáteční nastavení synaptických vah, zadané v průvodci, je aplikováno pouze na synaptická spojení mezi skrytou a výstupní vrstvou.

Váhy mezi vstupní a skrytou vrstvou jsou nově inicializovány vždy při zahájení učení sítě. K tomu je využíván algoritmus *k-means* s parametrem k, který odpovídá počtu RBF jednotek. RBF jednotky využívají Gaussovu aktivační funkci, kde pro hodnotu parametru σ je volena vzdálenost nejbližší vedlejší jednotky.

Pro RBF sítě je k dispozici vizualizační panel *Input Space*, který vychází z obdobného panelu Kohonenovy sítě. V tomto panelu jsou zobrazeny všechny přípustné hodnoty vstupů z trénovací množiny a rozmístění RBF jednotek. Tento panel obsahuje volbu nastavení *Coloring* s následujícími možnostmi, měnícími obarvení vstupních hodnot v grafu:

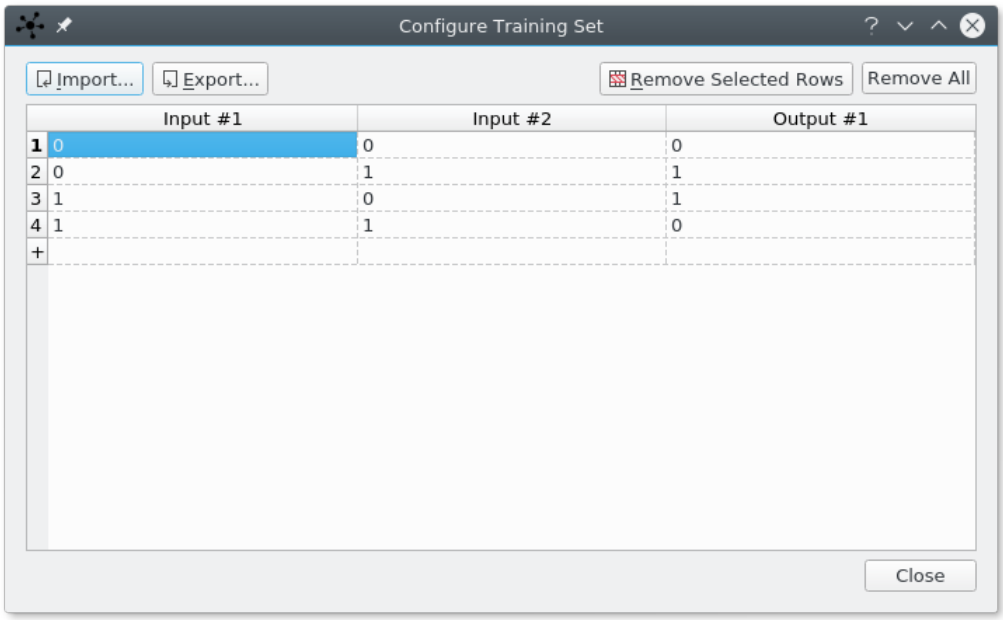
- *Output Classes*: barevně jsou rozlišeny vstupní hodnoty s různými kombinacemi požadovaných hodnot výstupních neuronů;
- *Uniform*: vstupní hodnoty nejsou nijak rozlišeny;
- *Values of Output #n*: barevně jsou rozlišeny vstupní hodnoty s různými požadovanými výstupy neuronu *Output #n*;



Obrázek 13: Vizualizace RBF sítě s datovou sadou Iris [6]

4.8 Definice trénovacích vzorů

Trénovací vzory lze definovat jejich vložením do tabulky, přístupné pomocí volby *Configure Training Set* z nabídky *Neural Network* a také z panelu nástrojů. Řádky tabulky tvoří jednotlivé trénovací vzory. Počet sloupců je dán počtem vstupních neuronů a v případě perceptronových, ADALINE a RBF sítí také počtem výstupních neuronů. Do sloupců výstupních neuronů je zadáván jejich požadovaný výstup. Požadované výstupy jednovrstvé perceptronové sítě mohou obsahovat pouze hodnoty 0 a 1, ADALINE sítě pouze hodnoty -1 a 1, v ostatních případech jsou přípustné celočíselné nebo reálné hodnoty. Import a export trénovací sady je možný ve formátech CSV a XML.



Obrázek 14: Tabulka trénovacích vzorů

4.9 Doplňkové funkce

Přestože se aplikace zaměřuje zejména na vizuální znázornění procesu učení neuronových sítí, byla doplněna několika užitečnými podpůrnými funkcemi, zejména:

- možnost uložení sítě do souboru ve formátu XML,
- přizpůsobení v podobě pojmenování sítě, vrstev a neuronů,
- jednotlivá i hromadná změna synaptických vah náhodným výběrem z určitého intervalu hodnot nebo nastavením konkrétní hodnoty,
- možnost ručního zadání vstupních dat a dopočítání průběžných i výstupních hodnot pro daný vstup.

5 Použité technologie a nástroje

5.1 Programovací jazyk C++

Programovací jazyk C++ vyvinul na počátku 80. let dánský informatik Bjarne Stroustrup. Jeho prvotní snahou bylo využít silné stránky programovacího jazyka C a rozšířit jej o objektové vlastnosti jazyka Simula. V roce 1998 vznikl ISO C++ standard a další významné rozšíření přinesl standard C++11 z roku 2011, který jazyk obohatil, mimo jiné, o funkcionální prvky a významně rozšířil standardní knihovnu [11].

Pro vývoj vizualizační aplikace byl zvolen standard C++11 a využívány byly zejména překladače Clang a GCC na platformě Linux. Pro sestavení aplikace na platformě Windows byl použit balík nástrojů MinGW, který je distribuován společně s komponentami knihovny Qt a který obsahuje port překladače GCC pro systém Windows. Mezi hlavní důvody volby tohoto jazyka patřilo:

- existence mnoha open-source knihoven, usnadňujících implementaci grafického uživatelského rozhraní, vizualizačních prvků a práci s XML daty;
- dostupnost kvalitních open-source překladačů napříč platformami;
- oproti dřívějšímu standardu přináší C++11 podporu *type inference* a *lambda výrazů* [11], které se při vývoji uplatnili zejména při použití mechanismu signálů a slotů knihovny Qt [13].

5.2 Knihovna Qt

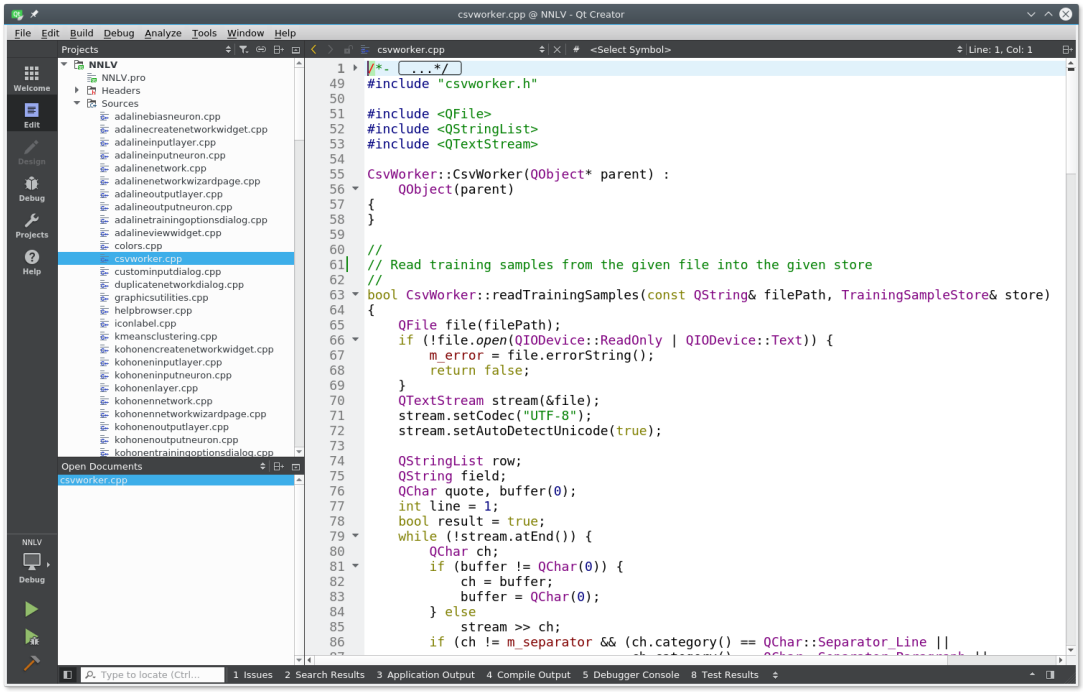
Vývoj knihovny Qt začal již na počátku 90. let, kdy se zakladatelé projektu Haarvard Nord a Eirik Chambe-Eng rozhodli vyvinout objektově-orientovaný, multiplatformní systém pro vývoj grafických aplikací. V současné době knihovna poskytuje rozsáhlý soubor modulů pro tvorbu grafických uživatelských rozhraní, multimediálních, síťových a databázových aplikací, včetně podpory vývoje pro mobilní platformy. Mezi hlavní důvody volby knihovny Qt patřilo:

- knihovna nabízí komplexní a prověřené aplikační rozhraní pro tvorbu aplikací s grafickým uživatelským rozhraním,
- umožňuje vytvářet multiplatformní aplikace, schopné přizpůsobit se vzhledu nativních aplikací dané platformy,
- poskytuje všechny požadované komponenty pro tvorbu vizualizací umělých neuronových sítí a pro práci s daty.

Knihovna je distribuována ve verzích pod komerční licencí a pod svobodnými licencemi GPL a LGPLv3. Použitý modul *Qt Charts* byl v době tvorby této práce dostupný pouze v rámci verze pod licencí GPL [15] a proto byla pro vývoj aplikace zvolena právě tato verze.

5.3 Vývojové prostředí Qt Creator

Toto open-source vývojové prostředí je distribuováno společně s komponentami knihovny Qt v rámci stažitelného open-source instalačního balíčku. Běžně se také nachází v softwarových repozitářích distribucí systému Linux. Prostředí je schopno automaticky konfigurovat sestavení projektu, je v něm integrována nápověda knihovny Qt a grafický designer uživatelského rozhraní. Současně také rozumí jazyku C++, pro který je schopno generovat kostry tříd a inteligentně doplňovat názvy a parametry metod.



Obrázek 15: Vývojové prostředí Qt Creator

5.4 Nástroje Qt Installer Framework a Inno Setup

Sada nástrojů *Qt Installer Framework* umožňuje snadnou tvorbu grafických instalátorů pro platformy Windows, Linux a macOS [14]. Instalátory přitom znají specifika dané platformy a jsou schopny odvodit správné umístění pro instalované soubory dle práv přihlášeného uživatele a vytvořit zástupce pro spuštění aplikace. Pro vizualizační aplikaci byl prostřednictvím této sady nástrojů vytvořen instalátor pro platformu Linux.

Instalátor pro systém Windows byl vytvořen pomocí nástroje *Inno Setup*, který je specifický pro tuto platfromu a díky mnohem pokročilejším možnostem nastavení se ukázal být vhodnější volbou.

6 Vývoj aplikace

6.1 Uživatelské rozhraní

6.1.1 Primární okno aplikace

Uživatelské rozhraní programu využívá okenní režim *Single-Document Interface* (SDI). Tento režim byl vybrán, jelikož je z důvodu použitelnosti a přehlednosti vhodné reprezentovat objekt neuronové sítě pomocí primárního okna aplikace [2]. Při návrhu uživatelského rozhraní bylo také zvažováno použití jediného primárního okna a reprezentace neuronových sítí pomocí záložkových karet v tomto okně. Tento způsob se však ukázal být problematický ve spojení s dokovacími prvky, které mohou být vztaženy pouze k celému primárnímu oknu a ne k jednotlivým kartám.

Primární okno aplikace tvoří instance třídy QMainWindow. Okno obsahuje vždy jeden centrální prvek, který musí být instancí třídy QWidget. Kromě centrálního prvku může okno volitelně obsahovat hlavní nabídku (QMenuBar), panely nástrojů (QToolBar) a stavový řádek (QStatusBar). Pravá strana stavového řádku je v aplikaci využívána k oznamování průběhu učení neuronové sítě, jelikož knihovna Qt umožňuje mít v této oblasti stálý zobrazovací prvek. Levá část stavového řádku je použitelná pouze pro dočasná oznámení, které knihovna za určitých okolností automaticky odstraňuje.

Po stranách primárního okna se nacházejí dokovací oblasti, které aplikace využívá pro umístění vizualizačních panelů a pomocných nástrojů. Tyto prvky jsou zapouzdřeny v instancích třídy QDockWidget, které jsou následně umísťovány do dokovacích oblastí. Dokovací prvky lze mezi dokovacími oblastmi přesouvat, seskupovat, případně z nich i vytvářet samostatná okna. Tyto možnosti závisí na nastavení daných instancí QDockWidget.

Pro tvorbu nové sítě je využíván průvodce, který je založen na třídě QWizard. Vzhled a obecné fungování průvodce je plně v režii knihovny, jednotlivé obrazovky jsou tvořeny vlastními třídami, které jsou odvozeny z obecné třídy QWizardPage.

6.1.2 Grafy

Pro vykreslování grafů ve vizualizačních panelech byl využit modul *Qt Charts*, který je součástí distribuce knihovny Qt ve verzi pod licencí GNU GPL. Tento modul je dostupný ve verzích Qt 5.6 a novějších.

6.2 Objektový návrh

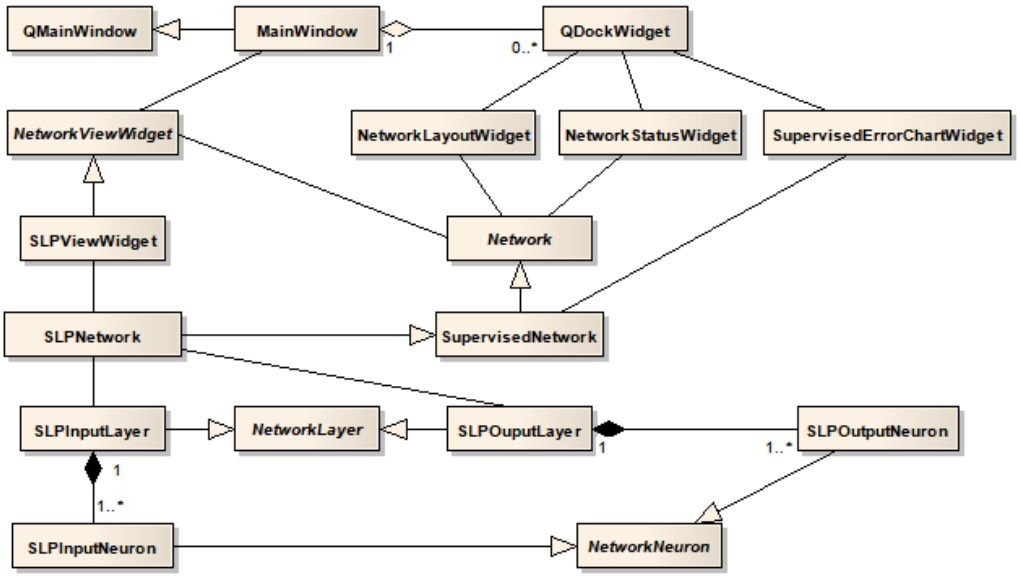
Při návrhu objektové architektury programu byl brán zřetel na množství obecných vlastností objektů neuronových sítí a na možná budoucí rozšíření v podobě implementace dalších typů neuronových sítí. Významná část aplikace tedy využívá pouze instance obecných abstraktních tříd.

Třída každého typu neuronové sítě je potomkem abstraktní třídy *Network*. Tato abstraktní třída především poskytuje informace o neuronové síti, obsahuje

veškerou obecnou trénovací logiku (např. realizuje výběr trénovacích vzorů v požadovaných intervalech a ověřuje podmínky pro zastavení tréninku) a také zprostředkovává přístup k instancím abstraktní třídy *NetworkLayer*, tedy objektům vrstev neuronové sítě. Instance *NetworkLayer* dále umožňuje přístup k instancím abstraktní třídy *NetworkNeuron*.

Vykreslení diagramu sítě v aplikačním okně využívá knihovní komponentu *Graphics View Framework*. Pro získání vykreslovacích schopností dědí všechny výše uvedené abstraktní třídy z příslušných tříd komponenty *Graphics View Framework*. Získané vykreslovací metody jsou však implementovány až ve třídách konkrétních sítí.

Centrální prvek primárního okna, obsahující diagram sítě a ovládací prvky učení, je potomkem abstraktní třídy *NetworkViewWidget*. Metody této abstraktní třídy realizují ovládání rychlosti učení a také řídí spuštění a ukončení učení použitím vhodných metod abstraktní třídy *Network*. Konkrétní implementace, dědící z abstraktní třídy *NetworkViewWidget* potom např. realizují konstrukci vizualizačních panelů, což je již činnost, kterou je nutno provádět pro každý typ sítě odlišně. Vybrané vztahy mezi třídami jsou znázorněny v diagramu tříd na obrázku 16.



Obrázek 16: Diagram vybraných tříd aplikace

6.3 Práce s daty

Aplikace umožňuje ukládání vytvořených neuronových sítí do souborů ve formátu XML. Do souboru se ukládá rozložení neuronové sítě, aktuální hodnoty

synaptických vah, aktuální nastavení učení a také trénovací sada. Jedinou povinnou částí je přitom rozložení sítě, pro hodnoty vah a nastavení učení jsou při jejich neuvedení použity výchozí hodnoty. Ukázka XML struktury je uvedena v příloze A.

Dále je v aplikaci možné importovat a exportovat trénovací data ve formátech XML a CSV. Použití XML i v tomto případě je výhodné, jelikož je tím umožněn import trénovací sady z jiné uložené sítě se stejnou strukturou vstupní a výstupní vrstvy. Při exportu do XML formátu nejsou do souboru uložena žádná data o síti, ale pouze trénovací sada.

7 Instalace a sestavení aplikace

7.1 Instalace

Vytvořená aplikace je distribuována včetně grafických instalátorů pro platformy Windows a Linux x64. Instalátory obsahují kromě samotné aplikace také všechny potřebné knihovní závislosti a postarají se i o vytvoření zástupců pro spuštění aplikace.

7.2 Sestavení ze zdrojových kódů

Program může být snadno sestaven ze zdrojových kódů za předpokladu, že cílový systém obsahuje překladač jazyka C++ a instalaci knihovny Qt ve verzi 5.7 nebo novější se všemi vyžadovanými moduly.

Snadným způsobem získání knihovny Qt je pomocí instalátoru[2] verze pod licencí GNU GPL. Při instalaci knihovny pomocí tohoto instalátoru je nezbytné kromě základní instalace vybrat i nepovinný modul *Qt Charts*.

7.2.1 Sestavení v prostředí Qt Creator

Projektový soubor *NNLV.pro*, který se nachází v adresáři se zdrojovými soubory, lze přímo načíst ve vývojovém prostředí Qt Creator.

Při prvním načtení projektového souboru je zobrazen průvodce nastavením sestavení, ve kterém je nabídnuta možnost výběru pracovního adresáře pro sestavení a nastavení parametrů překladače. Po dokončení tohoto kroku lze projekt ihned sestavit a spustit, a to buď v produkčním, nebo v ladícím sestavení.

7.2.2 Sestavení pomocí příkazové řádky

Pro distribuce systému Linux je typická instalace softwarových balíčků ze softwarových repozitářů a také oddělení balíčků knihoven a vývojových s hlavičkovými soubory. Balíčky knihoven jsou dostačující pro spuštění již sestaveného programu, vývojové balíčky jsou nezbytné pro překlad ze zdrojového kódu.

[2] https://www.qt.io/download

Pro sestavení a provoz vizualizační aplikace na platformě Linux je nutné mít v systému nainstalovány balíčky modulů *core*, *gui*, *widgets*, *xml*, *xmlpatterns* a *charts*. V distribucích Debian, Ubuntu a z nich odvozených je vyžadovaný modul *Qt Charts* dostupný až od verze 17.10, která byla v době tvorby této práce ve vývoji. V těchto distribucích lze tedy doporučit instalaci knihovny Qt pomocí online instalátoru, který zmíněný modul obsahuje.

Posloupnost příkazů shellu pro sestavení a spuštění aplikace je uvedena ve zdrojovém kódu 1. V závislosti na použité platformě se postup může mírně lišit, např. v systému Windows při použití sady nástrojů MinGW je nutné příkaz *make* nahradit příkazem *mingw32-make*.

```
1  $ cd NNLV
2  $ qmake
3  $ make
4  $ ./NNLV
```

Zdrojový kód 1: Posloupnost příkazů pro sestavení programu

8 Možnosti rozšíření

Architektura aplikace (viz také kapitolu 6.2) byla zvolena s ohledem na možnost budoucího rozšíření. Nabízejí se zejména následující možná vylepšení:

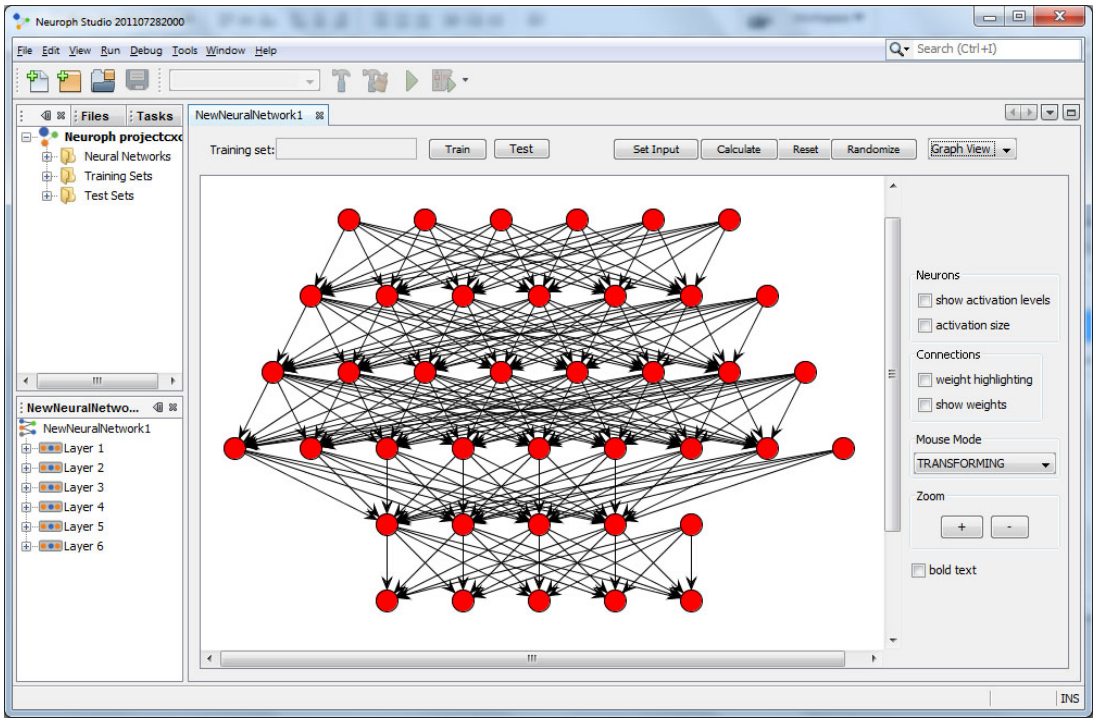
- doplnění o další modely neuronových sítí,
- rozšíření možností nastavení stávajících modelů o další učící parametry a pokročilejší učící algoritmy,
- vylepšení geometrického zobrazení pro možnost podbarvení různě klasifikovaných oblastí nebo vizualizaci vstupních dat ve skrytých vrstvách sítě,
- využití *Qt Data Visualization* pro 3D zobrazení vstupního prostoru,
- optimalizace pro možnost vizualizace rozsáhlejších neuronových sítí.

9 Jiné nástroje pro vizualizaci neuronových sítí

Aplikací, které se určitou formou zabývají vizuálním znázorněním neuronových sítí existuje značné množství.

Můžeme přitom mezi nimi nalézt nástroje pro návrh a vizualizaci komplexních neuronových sítí se schopností zhodnocení výsledného stavu sítě, např. ve formě grafu vývoje chyby, nebo znázorněním úspěšnosti klasifikace trénovacích vzorů. Tyto nástroje se uplatní např. při snaze o návrh neuronové sítě pro řešení praktických úloh, nebo pro zkoumání vlivu učících parametrů. Mezi tyto

nástroje lze zařadit např. *Neural Network Toolbox* [7] pro prostředí *MATLAB*, nebo aplikaci *Neuroph Studio* [8]. Způsob zobrazení sítě diagramem v aplikaci *Neuroph Studio* (viz obrázek 17) byl pro tuto práci významnou inspirací.



Obrázek 17: Diagram sítě v aplikaci *Neuroph Studio* [8]

Výukové nástroje, znázorňující stav sítě v jednotlivých krocích učícího algoritmu, lze často nalézt v podobě webových aplikací, appletů a jednoduchých grafických aplikací s předem definovanou množinou trénovacích vzorů. Zde lze zmínit např. aplikaci *xmlp* [1] pro vizualizaci učení vícevrstvého perceptronu s jednou skrytou vrstvou.

Závěr

Hlavní náplní této práce bylo vytvoření grafické aplikace pro vizualizaci učení vybraných typů umělých neuronových sítí pro výukové účely. Možné využití aplikace je tedy zejména uživateli z řad studentů a vyučujících univerzitních kurzů umělých neuronových sítí a strojového učení. Podporovanými modely jsou jednovrstvý a vícevrstvý perceptron, model ADALINE, Kohonenovy samoorganizační mapy a sítě typu radial basis function.

Aplikace dokáže znázornit proces adaptace uvnitř neuronové sítě, tedy změny synaptických vah a průběžných hodnot neuronů během jednotlivých kroků učení. Současně poskytuje, v závislosti na vybraném modelu, i geometrické znázornění aktuálního stavu a graf vývoje chyby sítě. Součástí aplikace je také sada několika hotových příkladů, které lze načíst při startu programu, nebo lze vytvořit novou síť vybraného typu dle zadaných parametrů. Trénovací sadu lze definovat pomocí tabulky nebo importem ze souboru. Aplikace byla vytvořena přenositelně, s využitím programovacího jazyka C++ a knihovny Qt.

Conclusions

The main goal of this thesis was to create a graphical application, capable of visualizing the learning processes in artificial neural networks. This application is intended to be used for educational purposes, mainly by students and teachers of university courses in artificial neural networks and machine learning. It supports single-layer and multi-layer perceptron networks, the ADALINE model, Kohonen self-organizing maps and radial basis function networks.

The application visualizes the process of adaptation inside a neural network, that is the changes of synaptic weights and values computed by individual neurons in each training iteration. Depending on the network type, it is also able to graph the geometrical representation of the network state and the current error for the loaded training set. Users can either load one of the included example networks, or they can define their own custom networks by selecting the network type and options. Training data can be added by filling the training table or loaded using the import function. The application was written portably in the C++ programming language and uses the Qt framework.

A Ukázka XML struktury uložené sítě

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network>
  <network-details>
    <name>SLP Network</name>
    <type>slp</type>
    <training-options>
      <learning-rate>0.1</learning-rate>
    </training-options>
    <layers>
      <layer>
        <type>input</type>
        <neurons>
          <neuron>
            <name>Input #1</name>
          </neuron>
          <neuron>
            <name>Input #2</name>
          </neuron>
        </neurons>
      </layer>
      <layer>
        <type>output</type>
        <neurons>
          <neuron>
            <name>Output #1</name>
          </neuron>
        </neurons>
      </layer>
    </layers>
  </network-details>
  <training-samples>
    <sample>
      <value>0</value>
      <value>0</value>
      <value>0</value>
    </sample>
  </training-samples>
</network>
```

Zdrojový kód 2: Ukázka XML struktury uložené jednovrstvé sítě perceptronů

B Obsah přiloženého DVD

Přiložené DVD obsahuje:

`bin/`
: Instalační soubory aplikace *Neural Network Learning Visualizer* pro operační systémy Windows a Linux x64. Součástí instalace jsou i všechny aplikací vyžadované runtime knihovny.

`doc/`
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce. Obsahem složky jsou také všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu.

`src/`
: Kompletní zdrojové texty aplikace.

`readme.txt`
: Instrukce pro sestavení a instalaci aplikace.

U veškerých cizích převzatých materiálů obsažených na DVD jejich zahrnutí dovolují podmínky pro jejich šíření.

Literatura

[1] BORGELT, Christian. Multilayer Perceptron Training Visualization [online]. Dostupné z: http://www.borgelt.net/doc/mlpd/mlpd.html

[2] DOSTÁL, Martin. Základy tvorby uživatelského rozhraní. Olomouc: Univerzita Palackého v Olomouci, 2008. 68 s. Dostupné z: https://phoenix.inf.upol.cz/esf/ucebni/gui-dostal.pdf

[3] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep Learning. 2016. Dostupné z: http://www.deeplearningbook.org

[4] HAYKIN, Simon. Neural Networks: A Comprehensive Foundation. 2nd Edition, Prentice Hall, Englewood Cliffs, 1999. 823 s. ISBN 81-7808-300-0.

[5] HERCULANO-HOUZEL, Suzana. The Human Brain in Numbers: A Linearly Scaled-up Primate Brain. Frontiers in Human Neuroscience. 2009, vol. 3, no 31. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2776484

[6] LICHMAN, M. UCI Machine Learning Repository [online]. University of California, Irvine, School of Information and Computer Sciences. Dostupné z: http://archive.ics.uci.edu/ml

[7] MATHWORKS. Neural Network Toolbox [online]. Dostupné z: https://www.mathworks.com/products/neural-network.html

[8] NEUROPH. Neuroph: Java Neural Network Framework [online]. Dostupné z: http://neuroph.sourceforge.net

[9] QUASAR, Jarosz. Neuron. Wikimedia Commons, [2009-08-11]. CC-BY-SA-3.0. Dostupné z: https://commons.wikimedia.org/wiki/File:Neuron_Hand-tuned.svg

[10] ROJAS, Raúl. Neural Networks - A Systematic Introduction. Springer-Verlag, Berlin, New-York, 1996. 502 s. Dostupné z: https://page.mi.fu-berlin.de/rojas/neural

[11] STROUSTRUP, Bjarne. The C++ Programming Language, Fourth Edition. Addison-Wesley Professional. 2013. 1 361 s. ISBN 978-0-321-56384-2.

[12] ŠÍMA, Jiri; NERUDA, Roman. Teoretické otázky neuronových sítí. Praha: MATFYZPRESS, 1996. 390 s. ISBN 80-85863-18-9. Dostupné z: http://www2.cs.cas.cz/~sima/kniha.html

[13] THE QT COMPANY. Qt Documentation: Differences between String-Based and Functor-Based Connections [online]. Dostupné z: http://doc.qt.io/qt-5/signalsandslots-syntaxes.html

[14] THE QT COMPANY. Qt Documentation: Qt Installer Framework Manual [online]. Dostupné z: http://doc.qt.io/qtinstallerframework

[15] THE QT COMPANY. Qt Licensing Comparison [online]. Dostupné z: https://www.qt.io/licensing-comparison

[16] VOLNÁ, Eva. Neuronové sítě 1, druhé vydání. Ostrava: Ostravská univerzita v Ostravě, 2008. 86 s. Dostupné z: http://www1.osu.cz/~volna/Neuronove_site_skripta.pdf

[17] VONDRÁK, Ivo. Neuronové sítě. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2001. 56 s. Dostupné z: http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf