10. kapitola

Neurónové siete

11.1. Historický úvod

Umelé neurónové siete [1-6] pôvodne vznikli ako modely mozgu resp. nervového systému a jeho činnosti. Postupne sa ale rozvinuli do nástrojov strojového učenia, používaných napr. na automatické rozpoznávanie priestorových objektov alebo časových postupností signálov, klastrovanie apod. Neurónová sieť je v podstate súborom jednoduchých prvkov (uzlov, neurónov), vykonávajúcich jednoduché operácie. Spojenia neurónov sú zvyčajne ohodnotené reálnymi číslami - váhami. Vo svojej najrozšírenejšej podobe (viacvrstvové s dopredným šírením) neurónové siete dostanú príklady so správnou odpoveďou, ktoré sa "naučia", a potom dokážu odpovedať aj na "dotazy" alebo klasifikovať vzory, ktoré nie sú celkom totožné s naučenými, ale iba podobné. Takéto neurónové siete môžu fungovať ako univerzálny aproximátor, teda modelovať s požadovanou presnosťou akúkoľvek spojitú funkciu, pokiaľ dostanú správne vzory na učenie. V takom prípade sú vlastne univerzálnym prostriedkom regresnej analýzy.

Problém je ale zvyčajne s učením, teda nastavením parametrov siete (napr. hodnôt váh, alebo množstva neurónov [7,8]). Nastaviť parametre tak, aby sieť čo najlepšie fungovala odpovedá optimalizácii, kedy premenné sú parametre funkcie a minimalizujeme chybu predpovedi siete. Najčastejšie sú takto optimalizované parametre viacvrstvovej neurónovej siete zo sigmoidným prenosom a dopredným šírením - viď ďalej obr. 11.3-11.5).

Trocha bočnou aplikáciou je použitie optimalizačných algoritmov pre výber (= redukciu) trénovacích dát. Pre aplikácie neurónových sietí ako klasifikátorov a prediktorov je významné pre efektívnosť adaptačného procesu, ale aj pre interpretáciu výsledkov dobre vybrať deskriptory (teda vstupné dáta) opisujúce vzory, ktoré sa má sieť naučiť interpretovať a/alebo predikovať. Objekt (vzor) môže byť opísaný vektorom rôznorodých dát, z ktorých iba časť môže byť významná pre klasifikáciu. KNN (ang. K Nearest Neighbor) klasifikátor [1] ohodnocuje testovaný objekt na základe najbližších K objektov (vzdialenosť objektu od testovaného objektu sa určuje ako suma absolútnych hodnôt rozdielov jednotlivých prvkov vektorov opisujúcich tieto dva objekty). Otázka je, ktoré prvky z opisných vektorov sa môžu pri výpočte tejto vzdialenosti vynechať. To sa dá opísať vektorom núl (vynechať) a jedničiek (započítať). Pri takomto výbere trénovacích dát, vlastne optimalizácii binárneho vektora, sa ukázalo výhodné použiť genetického algoritmu [9,10].

Podobne pomocou genetických algoritmov môžeme tiež hľadať oblasti (väčšinou obdĺžniky alebo hyperkvádre) v prehľadávanom priestore, v ktorých boli namerané želané výstupné hodnoty. Z takto vytvorených pravidiel (keď sú vstupné dáta v tom a v tom rozmedzí, a výsledok bude tiež v určitom rozmedzí) tak môžeme potom vytvoriť expertný systém. Žiaľ namerané výstupné dáta pre nejaké body v prehľadávanom priestore často chýbajú, a tak sa nahradzujú neurónovou sieťou, ktorá je už naučená aproximovať tvar funkcie z ostatných vstupov a výstupov [11].

Ďalšou atypickou aplikáciou je hľadanie takých dát, ktoré odpovedajú zadaným hodnotám výstupov už naučenej neurónovej siete [12,13]. Genetický algoritmus bol takto využitý na hľadanie hraničných ťažko rozhodnuteľných prípadov (ohodnotenie 0.5), kedy neurónová sieť bola naučená diagnostikovať zo vstupných dát či pacient má alebo nemá zápal slepého čreva (má = ohodnotenie 1, nemá = 0). Takéto výsledky môžu byť použiteľné jednak na extrakciu pravidiel, podľa ktorých sa neurónová sieť rozhodla (pokiaľ s klasifikáciou 0.5 súhlasíme), alebo na rozhodnutie, či sa sieť naučila dobre (napr. pokiaľ s klasifikáciou nesúhlasíme).

Neurónové siete, alebo ich vstupné dáta sú teda väčšinou objektom optimalizácie.

Zásadne odlišnou možnosťou prepojenia neurónových sietí s optimalizáciou je použitie špeciálne prispôsobených neurónových sietí ako optimalizačných metód. Neurónové siete, keď už sú používané na optimalizáciu (čo je skôr výnimka, väčšinou sa používajú namiesto polynomiálnej regresie) sú používané predovšetkým pre riešenie špecializovaných problémov. Takýmito problémami sú napr. spracovanie obrazu Hopfieldovou neurónovou sieťou, čo je rekonštrukcia objektu zo zašumeného alebo rozmazaného obrazu. Na tento účel Hopfieldova sieť funguje vynikajúco [14-19], no ide o veľmi špecializovaný problém. Hopfieldova sieť ale funguje aj pre klasické úlohy kombinatorickej optimalizácie [20], kedy si musíme iba zvoliť vhodnú reprezentáciu dát. Ide konkrétne o problémy váhového priradenia (weight matching), problém obchodného

cestujúceho (Traveling Salesperson Problem) [21-23], alebo aj delenie vrcholov grafov na dve podmnožiny (graph bipartitioning) [24]. Problém obchodného cestujúceho sa dá riešiť aj pomocou Kohonenovej neurónovej siete [25-27]. Potiaž je v tom, že posledne menované problémy riešia neurónové siete nie veľmi efektívne a v praxi sa preto pre tieto úlohy neurónové siete nevyužívajú, aj keď sa tieto algoritmy rôznymi úpravami postupne vylepšujú. Riešenia týchto problémov sa väčšinou publikujú iba ako ilustračné príklady pre teoretikov, že neurónovou sieťou takéto druhy úloh tiež idú riešiť.

Ďalej sa teda budeme zaoberať tak optimalizáciou neurónových sietí, ako aj optimalizáciou neurónovými sieťami.

11.2. Viacvrstvové neurónové siete s dopredným šírením

11.2.1 Všeobecný klasifikačný problém

Zavedieme všeobecnú formuláciu klasifikačného problému pomocou pojmu zobrazenia—funkcie definovanej nad dvomi množinami A a B. Tento prístup bude užitočný pre interpretáciu neurónových sietí ako *klasifikátora* alebo *prediktora*. Nech $F(\mathbf{x})$ je funkcia definovaná nad množinou A, ktorá priradí každému elementu $\mathbf{x}\in A$ obraz—funkčnú hodnotu z množiny B, $\hat{\mathbf{x}}=F(\mathbf{x})\in B$,

$$F: A \to B \tag{11.1}$$

Nech $G(\mathbf{x},\mathbf{w})$ je funkcia, ktorej argumenty sú z konečnej podmnožiny $A_{\text{train}}=\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_r\} \subset A$ (nazývanej *tréningová množina*) a $\mathbf{w}$ je parameter (alebo parametre) zobrazenia G, potom $\hat{\mathbf{x}}=G(\mathbf{x},\mathbf{w})\in B_{\text{train}}\subset B$ (pozri obr. 11.1)

$$G(\mathbf{w}): A_{train} \to B_{train} \tag{11.2}$$

Formálne môžeme povedať, že zobrazenie $G(\mathbf{w})$ je reštrikcia zobrazenia $F(\mathbf{x})$ nad množinou $A_{\text{train}}\subset A$. Komplement A_{train} vzhľadom k množine A je označený A_{test} (nazývaný *testovacia množina*), $A_{\text{test}}=A\setminus A_{\text{train}}$. Predpokladajme, že pre každé $\mathbf{x}_i\in A_{\text{train}}$ poznáme požadovaný obraz—funkčnú hodnotu $\hat{\mathbf{x}}_i$,

$$\mathbf{x}_1 / \hat{\mathbf{x}}_1, \mathbf{x}_2 / \hat{\mathbf{x}}_2, ..., \mathbf{x}_r / \hat{\mathbf{x}}_r \tag{11.3}$$

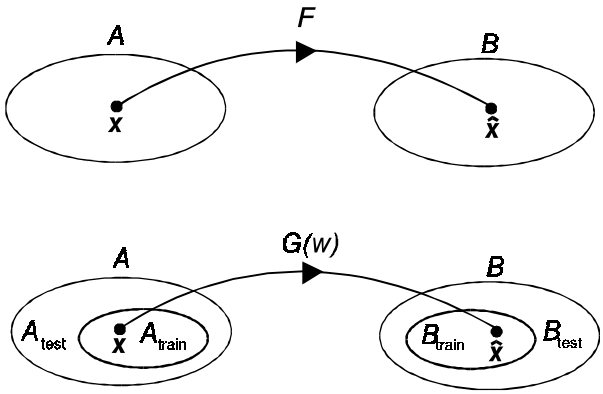
Požadované funkčné hodnoty $\hat{\mathbf{x}}_i$ sú interpretované ako obrazy funkcie F

$$\hat{\mathbf{x}}_i = F(\mathbf{x}_i) \qquad (i=1,2,...,r) \tag{11.4}$$

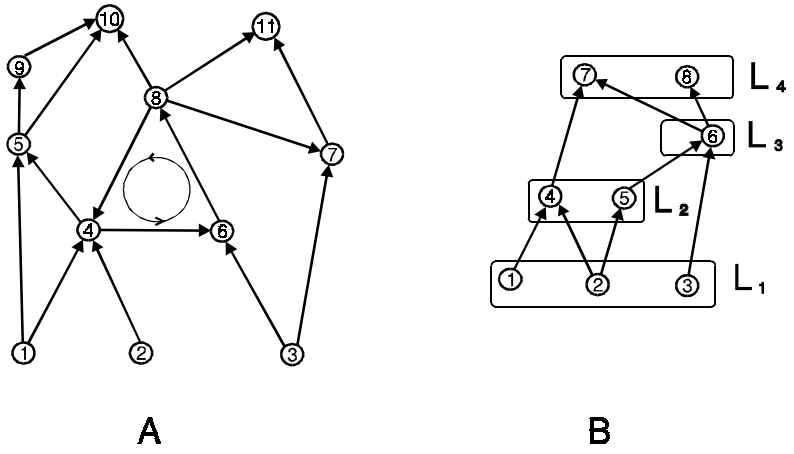
Cieľom našich úvah je nájsť taký parameter (alebo parametre) $\mathbf{w}$ funkcie $G(\mathbf{x},\mathbf{w})$, aby funkčné hodnoty argumentov z tréningovej množiny A_{train} boli čo najbližšie obrazom funkcie $F(\mathbf{x})$ (t.j. požadovaným hodnotám). Definujme *účelovú funkciu*

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{r}\left(G(\mathbf{x}_i,\mathbf{w}) - \hat{\mathbf{x}}_i\right)^2 \tag{11.5}$$

Táto funkcia vyjadruje sumu kvadrátov odchýlok funkcie $G(\mathbf{x},\mathbf{w})$ od požadovaných hodnôt $\hat{\mathbf{x}}$ braných z tréningovej množiny. Požiadavka, aby vypočítané hodnoty $G(\mathbf{x},\mathbf{w})$ boli "čo najbližšie" požadovaným hodnotám



Obrázok 11.1. Schematické znázornenie zobrazenia F: $A\to B$. Zúžením tohto zobrazenia na podmnožinu A_{train} dostaneme nové "modelové" zobrazenie $G(w)$, funkčný tvar tohto zobrazenia je určený parametrom (parametrami) w.



Obrázok 11.2. Neurónová sieť je definovaná ako orientovaný súvislý graf. Diagram A obsahuje orientovaný graf s jedným cyklom a teda nemôže byť použitý pre definíciu neurónovej siete s dopredným šírením. Diagram B ilustruje možnosť rozkladu vrcholov (neurónov) acyklického orientovaného grafu na vrstvy L_1,..,L_4.

$\hat{\boldsymbol{x}}$ je realizovaná pomocou požiadavky minimálnosti účelovej funkcie $E(\boldsymbol{w})$ vzhľadom k parametru $\boldsymbol{w}$. Hovoríme, že funkcia $G(\boldsymbol{x},\boldsymbol{w})$ je *adaptovaná*, ak jej parameter $\boldsymbol{w}$ je vybraný tak, aby sa rovnal svojej optimálnej hodnote (t.j. v ktorom má účelová funkcia globálne minimum). Nech $\overline{\boldsymbol{w}}$ je optimálna hodnota parametru $\boldsymbol{w}$ určená nasledujúcim minimalizačným problémom

$$\overline{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}\in W} E(\boldsymbol{w}) \tag{11.6}$$

kde W je množina (priestor) prípustných hodnôt parametra $\boldsymbol{w}$. Adaptovaná funkcia $G(\boldsymbol{x}, \overline{\boldsymbol{w}})$ simuluje pôvodnú funkciu $F(\boldsymbol{x})$ pre hodnoty argumentov z tréningovej množiny A_{train} na základe minimalizačného kritéria (11.6). Naviac, adaptovaná funkcia $G(\boldsymbol{x}, \overline{\boldsymbol{w}})$ sa používa aj pre predpoveď funkčných hodnôt odpovedajúcich argumentom z testovacej množiny A_{test}, t.j. predpokladá sa, že adaptovaná funkcia dobre aproximuje pôvodnú funkciu $F(\boldsymbol{x})$ tiež mimo tréningovej množiny. Naše úvahy môžu byť jednoducho chápané ako klasický regresný problém, v ktorom parametre modelovej funkcie G sú optimalizované (adaptované) tak, aby vypočítané funkčné hodnoty boli blízke požadovaným (experimentálnym) funkčným hodnotám.

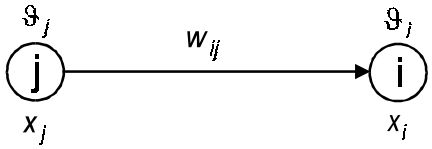
Jeden zo základných problémov v každej oblasti prírodných vied je *hľadanie vzťahu — funkcie medzi štruktúrou jej objektov a ich vlastnosťami*. Ideálom je konštrukcia tejto funkcie v analytickom tvare, ktorá vzťahuje vlastnosti objektov k ich štruktúre. V mnohých prípadoch je tento cieľ buď vôbec nerealizovateľný alebo len s veľkými obtiažami. Preto sa pomerne často používa prístup miešajúci čiastočné znalosti o problému s adaptáciou. Modelová funkcia G sa zostrojí na základe určitých úvah a jej voľne adjustovateľné parametre sa určia pomocou minimalizácie účelovej funkcie $E(\boldsymbol{w})$ (vzťah (11.6)). Takto adaptovaná modelová funkcia G sa potom berie ako analytický vzťah medzi štruktúrou prírodovedných objektov a ich vlastnosťami.

11.2.2 Definícia neurónovej siete

Formálne je neurónová sieť určená ako orientovaný graf G=(V,E), pozri obr. 11.2. Výrazy V={v_1,v_2,...,v_N} a E={e_1,e_2,...e_M} označujú neprázdnu vrcholovú množinu resp. hranovú množinu grafu G obsahujúceho N vrcholov (neurónov) a M hrán (spojov). Každý spoj $e\in$E sa interpretuje ako usporiadaná dvojica dvoch neurónov z množiny V, e=(v,v'). Hovoríme, že spoj e začína v neuróne v a končí v neuróne v'. Množina neurónov V je rozložená na disjunktné podmnožiny (pozri obr. 11. 2)

$$\mathrm{V} = \mathrm{V}_I \cup \mathrm{V}_H \cup \mathrm{V}_O \tag{11.7}$$

kde V_I obsahuje N_I vstupných neurónov, ktoré sú susedné len s vychádzajúcimi hranami, V_H obsahuje N_H skrytých (angl. *hidden*) neurónov, ktoré sú susedné súčasne s vychádzajúcimi ako aj s vchádzajúcimi hranami, a konečne V_O obsahuje N_O výstupných neurónov, ktoré sú susedné len s vchádzajúcimi hranami. V našich nasledujúcich úvahách budeme vždy predpokladať, že množiny V_I a V_O sú neprázdne, t.j. neurónová sieť obsahuje vždy aspoň jeden vstupný a jeden výstupný neurón.



Obrázok 11.4. Neuróny a spoje neurónovej siete sú ohodnotené reálnymi číslami: neuróny sú ohodnotené dvoma rôznymi parametrami, a to prahovými faktormi ϑ_I a aktivitami x_i. Spoje neurónovej siete sú ohodnotené váhovými koeficientmi w_{ij}.

Pre acyklické neurónové siete (ktoré neobsahujú orientované cykly (pozri graf A na obr. 11.2) neuróny môžu byť usporiadané do vrstiev (pozri graf B na obr. 11.2)

$$V = L_1 \cup L_2 \cup L_3 \cup \ldots \cup L_t \tag{11.8}$$

kde $L_1=V_I$ je *vstupná vrstva* (obsahuje len vstupné neuróny), L_2, L_3,..., L_{t-1} sú *skryté vrstvy* a L_t je *výstupná vrstva*. Vrstva L_i (pre $1 \le i \le t$) je určená nasledujúcim jednoduchým spôsobom

$$L_i = \{ v \in V; d(v) = i+1 \} \tag{11.9}$$

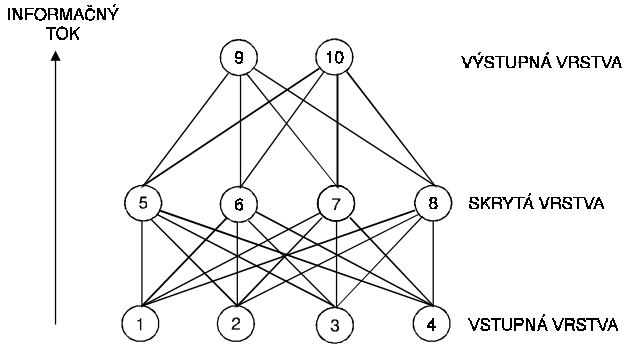
kde vzdialenosť $d(v)$ sa rovná dĺžke maximálnej cesty, ktorá spája daný neurón so vstupným neurónom, potom musí platiť $d(v)=0$, pre $v \in V_I$. Neurónová sieť určená acyklickým grafom je obvykle volená tak, že neuróny z dvoch susedných vrstiev sú poprepájané všetkými možnými spojmi (pozri obr. 11.3). Žiaľ, takýto rozklad množiny neurónov na vrstvy je možný len pre neurónové siete reprezentované acyklickými grafmi, pre cyklické grafy vzdialenosť $d(v)$ môže nadobúdať ľubovolnú kladnú celočíselnú hodnotu.

Neuróny a spoje sú ohodnotené reálnymi číslami, pozri obr. 11.4. Každý neurón v_i je ohodnotený *prahovým koeficientom* ϑ_i a *aktivitou* x_i. Podobne, každý spoj (v_j,v_i) je ohodnotený váhovým koeficientom (alebo jednoducho, váhou) w_{ij}. Postulujeme, že aktivity skrytých a výstupných neurónov sú určené vzťahom

$$x_i = t(\xi_i) \tag{11.10a}$$

$$\xi_i = \sum_{j \in \Gamma_i^{-1}} w_{ij} x_j + \vartheta_i \tag{11.10b}$$

kde sumácia beží cez neuróny, ktoré sú predchodcami neurónu v_i. Veličina ξ_i sa nazýva *potenciál* neurónu v_i. *Prechodová (aktivačná) funkcia* $t(\xi)$ z pravej strany (11.10a) je monotónne rastúca funkcia, ktorá vyhovuje nasledujúcim dvom asymptotickým podmienkam: $t(\xi) \to A$, pre $\xi \to -\infty$ a $t(\xi) \to B$, pre $\xi \to \infty$, kde $-\infty < A < B < \infty$. V teórii neurónových sietí sa často využíva nasledujúca "sigmoidálna" funkcia



Obrázok 11.3. Znázornenie trojvrstvovej neurónovej siete s dopredným šírením.

$$t(\xi) = \frac{B + Ae^{-\xi}}{1 + e^{-\xi}} \tag{11.11}$$

Táto prechodová funkcia zobrazuje celú množinu reálnych čísel R na otvorený interval (A,B), formálne $t:\mathrm{R} \to (A,B)$. Najčastejšie sa prechodová funkcia (11.11) využíva pre hodnoty parametrov A=0, B=1 alebo A=-1, B=1 (pozri obr. 11.5). Prvý graf odpovedá klasickej sigmoidálnej prechodovej funkcii, zatiaľ čo druhý graf je analógiou hyperbolického tangentu.

Sieť dostane ako vstup aktivačný vzor pre svoje vstupné jednotky (ináč povedané, vstupné aktivity, ako napr. hodnoty pixelov v obrázku ručne písaného písmena A, sú pridelené vstupným neurónom). Aktivácie sa šíria dopredu zo vstupných jednotiek cez jednu alebo viac prostredných vrstiev "skrytých" neurónov cez spojenia ohodnotené váhami, až skončia vo výstupných jednotkách. Aktivácie sú v priebehu šírenia z jednotky na jednotku násobené váhami spojenia, cez ktoré idú, a potom sú sčítané dohromady s ostatnými vstupujúcimi aktiváciami. Tento súčet je potom spracovaný aktivačnou funkciou. Takto vytvorená aktivácia sa potom rovnakým spôsobom šíri ďalej až do výstupného neurónu. Aktivácie výstupných neurónov sú zakódovanou "odpoveďou" siete na vstup, napríklad ktoré písmeno bolo vstupom siete.

V poslednej dobe sa používajú genetické algoritmy aj pre rekurentné neurónové siete, ktoré obsahujú aj spätné spojenia medzi vrstvami [28]. Topológia a váhy siete sa pritom optimalizujú súčasne.

Aktivity neurónov tvoria vektor $\boldsymbol{x}=(x_1,x_2,...x_N)$. Tento vektor možno formálne rozložiť na tri podvektory obsahujúce vstupné, skryté a výstupné aktivity

$$\boldsymbol{x} = \boldsymbol{x}_I \oplus \boldsymbol{x}_H \oplus \boldsymbol{x}_O \tag{11.12}$$

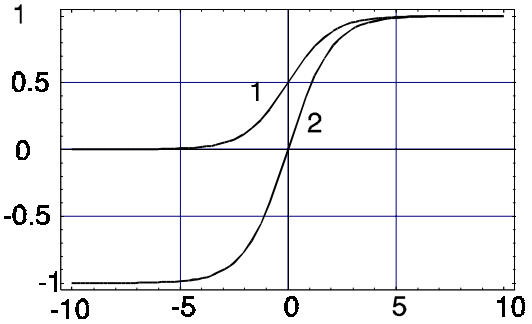
Neurónovú sieť s fixovanými váhami a prahovými koeficientmi možno formálne chápať ako funkciu

$$G:\mathrm{R}^{N_I} \to (A,B)^{N_O} \tag{11.13}$$

ktorá priradí vstupnej aktivite $\boldsymbol{x}_I$ (deskriptor) výstupný vektor $\boldsymbol{x}_O$ (klasifikátor) s hodnotami svojich zložiek z otvoreného intervalu (A,B)

$$G(\boldsymbol{x}_I) = \boldsymbol{x}_O \tag{11.14}$$

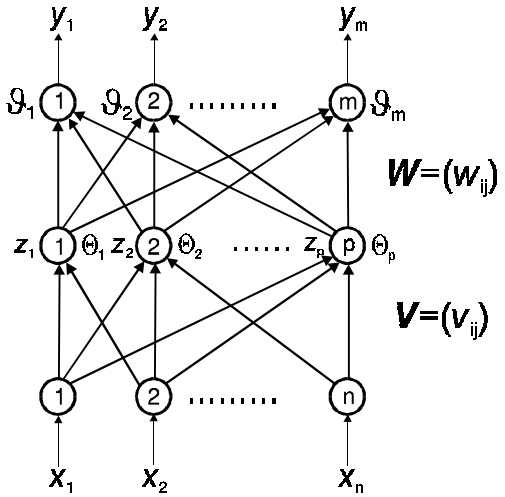
Skryté aktivity nie sú explicitne uvedené, hrajú len úlohu medzivýsledkov. Ešte niekoľko poznámok k výpočtu aktivít podľa (11.10a-b). Vstupné aktivity sú určené deskriptorom, preto ich pokladáme za fixované. Aktivity



Obrázok 11.5. Priebeh aktivačnej funkcie definovanej (11.11). Graf 1 odpovedá štandardnej sigmoide (A=0, B=1), graf 2 je podobný funkcii hyperbolický tangent (A=−1,B=1).

skrytých neurónov z druhej vrstvy L_2 môžeme teraz spočítať len použitím vstupných aktivít z vrstvy L_1. Vo všeobecnosti pre výpočet aktivít z vrstvy L_i (kde i>1) musíme poznať len aktivity z nižších vrstiev L_1, L_2,..., L_{i-1} . Týmto rekurentným spôsobom môžeme postupne spočítať aktivity všetkých neurónov, ako posledné sa počítajú aktivity výstupných neurónov. Vďaka tomu sa pre neurónové siete reprezentované acyklickým grafom zaužíval názov *neurónové siete* s *dopredným šírením* (angl. *feed-forward neural networks*). Žiaľ, spomínaný jednoduchý postup výpočtu aktivít neurónov je aplikovateľný len na neurónové siete reprezentované acyklickým orientovaným grafom. V prípade, že graf obsahuje orientované cykly (ide rekurentné siete), tento postup nie je použiteľný. Rovnice (11.10a-b) sú v tomto prípade spriahnuté a nelineárne. Preto ich riešenie (t.j. skryté a výstupné aktivity) môžeme dosiahnuť len použitím iteračného postupu, a to tak, že štartujeme z počiatočných aktivít, pomocou týchto spočítame nové aktivity a tieto sa v nasledujúcom iteračnom kroku použijú ako vstup pre výpočet nových aktivít. Tento iteračný postup sa opakuje tak dlho, až rozdiel medzi starými a novými aktivitami je menší ako predpísaná presnosť.

Vo väčšine aplikácií sa sieť učí správne mapovanie medzi vstupnými a výstupnými vzormi pomocou učiaceho algoritmu. Nastavenie váhových koeficientov neurónových sietí bolo dlho veľkým problémom. Až Rumelhart so spolupracovníkmi [29] navrhli jednoduchý gradientový algoritmus (nazvaný metóda spätného šírenia - angl. backpropagation) adaptácie viacvrstvových neurónových sietí s dopredným šírením. Typicky sú



Obrázok 11.6. Trojvrstvová neurónová sieť s dopredným šírením. Vstupná vrstva obsahuje *n* vstupných neurónov s aktivitami $x_1, x_2, ..., x_n$. Skrytá vrstva obsahuje *p* skrytých neurónov, ich aktivity resp. prahové koeficienty sú označené $z_1, z_2, ..., z_p$ resp. $\Theta_1, \Theta_2, ..., \Theta_p$. Výstupná vrstva obsahuje *m* výstupných neurónov, ich aktivity resp. prahové koeficienty sú označené $y_1, y_2, ..., y_m$ resp. $\vartheta_1, \vartheta_2, ..., \vartheta_m$. Váhové koeficienty medzi vstupnou a skrytou vrstvou resp. skrytou a výstupnou vrstvou tvoria maticu $\boldsymbol{V}=(v_{ij})$ resp. maticu $\boldsymbol{W}=(w_{ij})$.

váhy spočiatku nastavené na malé náhodné hodnoty. Potom je množina trénovacích vzorov postupne zadávaná sieti. Potom, čo každý vstup (vzor) prešiel sieťou a pre každý vzor bol vyprodukovaný výstup, "učiteľ" porovná aktivácie jednotlivých výstupných jednotiek zo správnymi hodnotami. Váhy siete sú potom nastavené tak, aby sa čo najviac zmenšil rozdiel medzi výstupom siete a správnym dopredu známym výstupom. Každá iterácia tejto procedúry je trénovacím cyklom, prebehnutie trénovacích cyklov cez všetky vzory je trénovacou epochou. Takýchto epoch je na naučenie siete typicky potrebné rádovo cez desať tisíc.

Okrem najjednoduchšej gradientovej metódy najväčšieho spádu sa často používajú aj iné metódy, ako metóda spriahnutých gradientov, Levenberg-Marquardtova metóda apod. [30]. No najjednoduchšia metóda najväčšieho spádu sa stále ešte používa najčastejšie. Ostatné metódy sú síce rýchlejšie, no môžu sa jednoduchšie "preučiť" alebo skĺznuť do lokálneho minima. "Preučenie u neurónových sietí znamená niečo podobné ako použitie príliš veľkého stupňa polynómu u polynomiálnej regresie. Neurónová sieť sa v takom prípade naučí takmer perfektne zo zadaných vstupov tréningovej množiny produkovať požadované výstupy, no pre testovaciu množinu zlyháva. Podobný problém často nastáva aj pri použití evolučných optimalizačných algoritmov na nastavenie parametrov neurónové siete.

11.2.3 Algoritmizácia neurónovej siete s dopredným šírením

Ďalej naznačíme základné princípy algoritmizácie neurónových sietí s dopredným šírením, ktoré obsahujú skryté neuróny. Pre jednoduchosť budeme uvažovať 3-vrstvovú neurónovú sieť, ktorá obsahuje jednu vrstvu skrytých neurónov, pričom neuróny zo susedných vrstiev sú prepojené všetkými možnými spôsobmi, pozri obr. 11.6.

Aktivity skrytých a výstupných neurónov sú určené vzťahmi (pozri vzťahy (11.10a-b))

$$z_i = t\left(\sum_{j=1}^{n} v_{ij} x_j + \Theta_i\right) \qquad (\text{pre } i = 1,2,\ldots,p) \tag{11.15a}$$

$$y_i = t\left(\sum_{j=1}^{p} w_{ij} z_j + \vartheta_i\right) \qquad (\text{pre } i = 1,2,\ldots,m) \tag{11.15b}$$

kde $t(\xi)$ je sigmoida určená pomocou (11.11a) (s parametrami A=0 a B=1)

$$t(\xi) = \frac{1}{1+e^{-\xi}} \tag{11.16}$$

Grafický priebeh tejto funkcie je znázornený na obr. 11.5, graf 1.

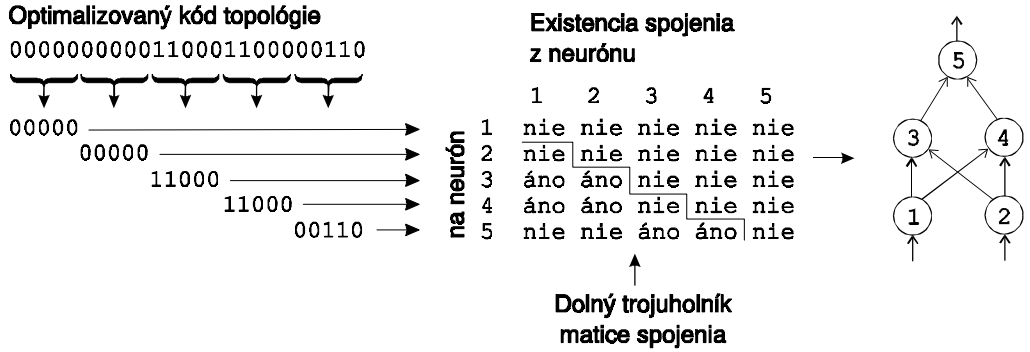
Výpočet aktivít neurónov pre dané váhové a prahové koeficienty sa nazýva aktívna fáza neurónovej siete. Tieto aktivity pre danú neurónovú sieť sa vypočítajú jednoduchým rekurentným postupom: Predpokladajme, že vstupné aktivity x_1, x_2, ..., x_n (deskriptory klasifikovaného objektu) sú známe, potom pomocou (11.15a) zostrojíme aktivity skrytých neurónov z_1, z_2, ..., z_p. Následne, pomocou (11.15b) zostrojíme aktivity výstupných neurónov y_1, y_2, ..., y_m. Uvedený rekurentný spôsob výpočtu aktivít postupuje zdola nahor neurónovou sieťou. Táto skutočnosť sa odráža v názve týchto sietí, ako neurónových sietí s dopredným šírením signálu. Algoritmizácia tohto postupu je uvedená formou pascalovského pseudokódu v algoritmu 11.1.

11.2.4 Optimalizácia neurónových sietí s dopredným šírením

Evolučné algoritmy sa používajú pri stochastickej optimalizácii neurónových sietí. Tieto algoritmy využívajú informáciu z predchádzajúceho riešenia pri návrhu novej, lepšej siete. Snažíme sa pritom, aby suma štvorcov odchyliek výstupov z neurónovej siete od vopred zadaných hodnôt bola čo najmenšia. Tam, kde sa budeme zaoberať optimalizáciou topológie dopredných viacvrstvových neurónových sietí, bude učiacu stratégiu tvoriť ten najznámejší algoritmus - "*backpropagation*", teda. metóda adaptácie pomocou spätného šírenia chyb. Toto zjednodušenie je oprávnené z toho dôvodu, že optimalizácia pomocou stochastických evolučných algoritmov sa v naprostej väčšine používa práve pre tento základný typ neurónovej siete.

V optimalizácii neurónových sietí sa vyskytujú dva základné úkoly: optimalizácia topológie a optimalizácia váh.

■ *Optimalizácia topológie neurónovej siete* spočíva v určení počtu skrytých vrstiev, počtu neurónov v týchto vrstvách, existencie spojení medzi nimi (obr. 11.7) [31,32], poprípade aj parametrov prechodovej funkcie neurónov alebo parametrov pre učenie siete pomocou spätného šírenia.



Obrázok 11.7. Príklad zakódovania topológie spojenia piatich neurónov. Optimalizácia topológie neurónovej siete je pretransformovaná na optimalizáciu úvodného bitového reťazca, pričom prvé dva neuróny sú definované ako vstupné a posledný piaty neurón je výstupný (podľa [31]).

```
procedure activities(input :Θ,V,ϑ,W,x;
                     output:z,y);
begin for i:=1 to p do
    begin ξ:=Θ[i];
        for j:=1 to n do ξ:=ξ+v[i,j]*x[j];
        z[i]:=t(ξ);
    end;
    for i:=1 to m do
    begin ξ:=ϑ[i];
        for j:=1 to p do ξ:=ξ+w[i,j]*z[j];
        y[i]:=t(ξ);
    end;
end;
```

Algoritmus 11.1. Algoritmizácia v pascalovskom pseudokóde aktívnej fáze neurónovej siete s dopredným šírením, ktorá obsahuje jednu vrstvu skrytých neurónov. Vstupnými parametrami procedúry activities sú vstupné aktivity a váhové a prahové koeficienty, výstupnými parametrami sú skryté a výstupné aktivity. Reálna funkcia t(ξ) je prechodová funkcia definovaná (11.16).

Uvedený príklad ukazuje iba zlomok možností, čo všetko sa dá optimalizovať. Aj tak je príklad príliš zjednodušený. Pokiaľ by sme nechceli zakaždým kontrolovať, či je sieť skutočne "feed forward", teda či náhodou niektoré spojenia netvoria cyklus, potom je najjednoduchší vyplniť iba dolní trojuholník matice spojení. V každom prípade je ale treba pred učením vyraďovať neprijateľné architektúry, napr. také, kde nevedie cesta zo vstupu do výstupu, a vyraďovať tiež vnútorné neuróny a ich spojenia, ktoré neležia na žiadnej ceste zo vstupu do výstupu.

Tento typ zakódovaní topológie sa nazýva "nízkoúrovňovým", pretože priamo kóduje topológiu. Pri väčších sieťach je možné nekódovať priamo spojenia, ale treba len percento spojená medzi dvoma skupinami neurónov [33]. U ešte zložitejších typov zakódovania, tzv. "vysokoúrovňových", môže byť architektúra siete špecifikovaná pravidlami rastu alebo vetami formálneho jazyka. Tento typ je viac vhodný pre optimalizáciu rozsiahlejších sietí (na ktorú v našich podmienkach ale väčšinou nemáme dostatočne rýchly hardware). Vysokoúrovňové zakódovania sú väčšinou predstavované tzv. gramatickým kódovaním, kedy evolučné algoritmy nevyvíjajú priamo architektúru siete, ale skôr pravidlá formálnych gramatík, ktoré sú potom používané pre generovanie vlastnej topológie siete.

Tak "nízkoúrovňové", ako aj "vysokoúrovňové" zakódovanie musí byť ohodnotené, teda zo zakódovania musí byť vyrobená sieť, ktorá je inicializovaná malými náhodnými hodnotami váh o potom optimalizovaná väčšinou štandardným "backpropagation" algoritmom. Správne by sme asi mali pre každé také zakódovanie urobiť niekoľko štartov s rozdielnymi štartovnými váhami, a potom zobrať alebo najlepší výsledok, alebo priemer pre ohodnotenie "chromozómu" opisujúceho zakódovanie siete. V praxi sa ale robí pre každú sieť iba jeden štart, a to ešte sa väčšinou snažíme čo najviac redukovať počet cyklov backpropagation algoritmu.

Gramatické zakódovanie môže byť predstavené prácou Hiroaki Kitana [34]. Ten argumentuje tým, že počet možných spojení neurónov rastie kvadraticky s počtom neurónov, čo za použitia priameho kódovania môže pre veľké siete predstavovať problém. Gramatické kódovanie je schopné efektívne zakódovať aj niektoré opakujúce sa alebo zahniezdené podštruktúry siete. Pravidlá formálnych gramatík sú optimalizované genetickými algoritmami, no po každej zmene musí nasledovať vytvorenie siete podľa nových pravidiel a jej ohodnotenie. Toto ohodnotenie potom slúži ako podklad pre fitnes funkciu daného súboru pravidiel gramatiky.

Gramatika tak môže byť nazvaná "genotypom", analógiou genetického kódu živočíchov alebo rastlín, zatiaľ čo samotná sieť môže byť stotožnená s fenotypom, teda príkladom živočícha alebo rastliny vytvorenej podľa genetického kódu.

Vytváranie "slov" podľa určitej gramatiky sa robí postupným prepisovaním reťazcov znakov podľa pravidiel gramatiky. Generatívna gramatika obsahuje dve disjunktné konečné abecedy, štartovný symbol S, a prepisovacie pravidlá. Prvá abeceda sa volá množina neterminálov (premenných), druhá abeceda sa volá množina terminálov a prepisovacie pravidlá pre bezkontextovú gramatiku prepisujú vždy jednu premennú na reťazec vytvorený z množiny premenných a terminálov. Výsledným slovom je potom podľa týchto pravidiel vytvorený reťazec, ktorý obsahuje už iba terminály. Najjednoduchším takým prípadom by bola abeceda premenných tvorená jediným symbolom S, abeceda terminálov tvorená dvoma symbolmi 0 a 1, a produkčné pravidlá $S \rightarrow 0S1$, $S \rightarrow 01$.

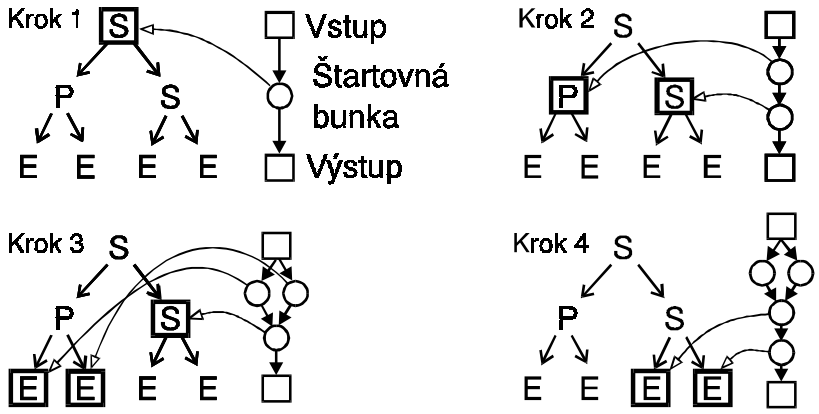
Zo štartovacieho znaku S tak ide vytvoriť každé "slovo" obsahujúce v prvej polovici samé nuly a v druhej samé jednotky, napr.

$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 0000S1111 \Rightarrow 00000S11111 \Rightarrow 000000S111111 \Rightarrow 00000001111111.$

Kitanov produkčný systém obsahoval abecedu premenných A-Z a abecedu šestnástich terminálov a-p. Grafy generujúca gramatika potom na pravej strane obsahuje namiesto jednorozmerného reťazca maticu 2×2. Šestnásť terminálov potom jednoznačne odpovedá všetkým možným maticám 2×2 zloženým z núl a jedničiek. Zo štartovného symbolu sa pomocou pravidieľ vygeneruje matica terminálov, z tej matica susednosti, a z nej potom orientovaný graf. Pravidlá sú pritom dané chromozómom (viď obr. 11.8). Tri neuróny odpovedajúce predposledným trom riadkom v matici nie sú s ničím spojené, preto nie sú zakreslené do grafu.

V zjednodušenej Kitanovej verzii je povinný iba prvý symbol S, ostatné sa môžu meniť (mutovať). Keď sa niektorá z premenných A-Z objaví v pravidlách viackrát, berie sa pri konštrukcii matice susednosti do úvahy iba prvé z pravidiel. Na pravej strane sa v tejto konkrétnej gramatike objavujú iba terminálne symboly, takže môžeme skonštruovať iba maticu 8×8. V prípade, že daný graf má aj rekurentné spojenia, tak sa tieto nebudú brať do úvahy, a budú sa brať iba grafy s dopredu daným počtami vstupných a výstupných vrcholov, odpovedajúcimi riešenému problému. Výsledné grafy sú potom učené pomocou "backpropagation" trénovaciou množinou a ohodnotené potom sumou chýb pre vzory z testovacej množiny. Kitano použil klasickú selekciou úmernú fitnes pomocou "rulety" (viď kapitola 4), kríženie s rozsekaním na niekoľko podreťazcov a následnou výmenou týchto podreťazcov, a mutáciu náhradou náhodne zvoleného symbolu symbolmi z A-Z a a-p abecied. Pravdepodobnosť mutácie potomkov bola priamo nepriamo úmerná Hammingovej vzdialenosti (počtu rozdielnych prvkov na odpovedajúcich si pozíciách) pre reťazce rodičov. Kitanove výsledky boli lepšie pre

gramatické kódovanie ako pre priame kódovanie, no výsledky nie sú príliš presvedčivé, pretože použité príklady boli príliš jednoduché. Ďalšie rozšírenie tohto prístupu, kde je integrovaný vývoj architektúry spolu s nastavením váh je možno nájsť v [36].



Obrázok 11.9. Vytváranie siete z celulárneho kódovania. Naľavo je vždy tzv. gramatický strom, napravo sieť s neurónmi (bunkami) označenými krúžkom. Šípky od neurónov na P, S, a E v strome určujú ďalšiu akciu neurónu. S znamená vertikálne rozdelenie neurónu na dva nové, P horizontálne rozdelenie a E ukončenie práce s neurónom (podľa Gruaua a Whitleyho [35]).

O zložitejšie zakódovanie sietí sa pokúsili Grunau, Bellew a Whitley [35,37,38], ktorí používajú k zakódovaniu siete tzv. celulárne kódovanie. Takto bolo vyvinuté napr. riadiace zariadenie pre šesťnohého robota, alebo balansovanie "tyče". Základom celulárneho kódovania je tzv. gramatický strom. Každá bunka ukazuje na určité miesto v tomto strome, a podľa toho, čo na tejto pozícii je, tak sa bunka rozdelí na dve bunky, zmení svoje vnútorné parametre (napr. prahový faktor neurónu), alebo premiestni svoj ukazovateľ na inú pozíciu v strome, čím je možné robiť rekurziu (treba si pamätať hĺbku rekurzie). Vývoj začína od jednej počiatočnej bunky ukazujúcej na vrchol stromu. Táto bunka je spojená so vstupom a výstupom. V Sekvenčnom rozdelení, označenom S, sa rodičovská bunka rozdelí na dve dcériné bunky tak, že prvá dcériná bunka podedí všetky vstupy rodičovskej bunky a druhá dcériná bunka podedí všetky výstupy. Pritom povedie spojenie z prvej dcérinej bunky na druhú. V obrázku 11.9 je druhá dcériná bunka vždy umiestnená pod prvou. Pri Paralelnom rozdelení, označenom P, sa rodičovská bunka rozdelí na dve nespojené dcériné bunky, každá z ktorých podedí aj vstupy, aj výstupy rodičovskej bunky. V obrázku 11.9 sú tieto bunky umiestnené vedľa seba. S aj P uzol

Príklad pravidiel generujúcej gramatiky

$$S \to \begin{matrix} A & B \\ C & D \end{matrix} \quad A \to \begin{matrix} c & p \\ a & c \end{matrix} \quad B \to \begin{matrix} a & a \\ a & e \end{matrix} \quad C \to \begin{matrix} a & a \\ a & a \end{matrix} \quad D \to \begin{matrix} a & a \\ a & b \end{matrix},$$

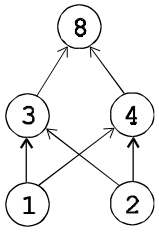
kde prepis terminálov (už nepatriacich do gramatiky) bude

$$a \to \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \quad b \to \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix} \quad c \to \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \quad e \to \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \quad p \to \begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix}$$

a s pomocou týchto pravidiel vznikne nakoniec matica susednosti

$$S \Rightarrow \begin{matrix} A & B \\ C & D \end{matrix} \Rightarrow \begin{matrix} c & p & a & a \\ a & c & a & e \\ a & a & a & a \\ a & a & a & b \end{matrix} \Rightarrow \begin{matrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

s odpovedajúcim grafom



Chromozóm odpovedajúci týmto pravidlám

S	A	B	C	D	A	c	p	a	c	B	a	a	a	e	C	a	a	a	a	...

Obrázok 11.8. Príklad pravidiel generujúcej gramatiky podľa Kitana [34].

zároveň znamenajú, že prvá dcériná bunka posunie svoj ukazovateľ do ľavého uzla naspodku, zatiaľ čo druhá dcériná bunka bude ukazovať do pravého uzla naspodku. Symbol E je potom symbol ukončenia. Je možné mať aj označenie uzla R, kedy sa ukazovateľ presunie na niektorý predchádzajúci uzol, čím je možno zaviesť rekurzivnosť. Podobne sa dá zadefinovať aj analógia Automaticky Definovaných Funkcií z Kozy [39], viď tiež kap. 6. Gramatické stromy sa podobne ako funkcie pri Kozovom genetickom programovaní [40] optimalizujú pomocou genetických algoritmov, kedy ohodnotenie chromozómu (gramatického stromu) je odvozené z chybovej funkcie odpovedajúcej neurónovej siete.

Stochastické optimalizačné algoritmy sú v podstate jediným systematickým prístupom k optimalizácii topológie siete. Žiaľ, tieto algoritmy potrebujú vygenerovať a ohodnotiť rádovo tisícovky (alebo viac) možných topológií neurónových sietí, aby dospeli k dobrému výsledku. Vzhľadom k tomu, že ohodnotenie každej topológie predstavuje vlastne naučenie jednej neurónovej siete pomocou tisícok iterácií spätného šírenia (alebo tisícok generácií genetických algoritmov), ide o výpočtovo veľmi náročný úkol. Preto sa na miesto systematického prístupu k návrhu topológie stále bežne používa skôr intuície, a príklady optimalizácie topológie neurónovej siete existujú väčšinou iba pre jednoduché problémy. Pri návrhoch sietí sa bežne používajú heuristiky ako "pre zložitejšie problémy je potrebné viac skrytých jednotiek" a metódy "pokusu a omylu". Návrh siete pre zložitejšie problémy je pritom často sprevádzaný neprípustným zjednodušovaním optimalizačných algoritmov. Ich hlavná sila, ktorou je prehľadávanie mnohorozmerného priestoru s viacerými minimami a schopnosť dostať sa z lokálnych miním a skončiť v globálnom minimu, sa potom znižuje. Zvyšuje sa tým pravdepodobnosť, že topológia skončí niekde v lokálnom minime, teda, že nebude ideálna. Žiaľ, výpočtové nároky týchto metód lepšie prehľadávanie možných topológií neumožňujú.

Okrem počtu vrstiev, neurónov a ich spojení môže byť sieť optimalizovaná na rýchlosť učenia, dosiahnutú presnosť a na veľa ďalších vecí. Ani optimalizovaná funkcia nemusí byť iba sumou odchyliek chyb predpovedí už známych faktov. Sieť môže byť optimalizovaná napr. na rýchlosť učenia, alebo na malý počet spojení medzi neurónmi, alebo na nízky počet vnútorných neurónov. Existuje aj špeciálne zakódovanie potenciálnej siete, tak aby sa ľahko menili počty neurónov a vrstiev, viď [33], ale tento postup má zasa iné nedostatky a nestojí na nejakej hlbokej teórii, preto ho tu nebudeme uvádzať.

Atypickým príkladom je práca Davida Chalmersa [41], ktorý optimalizoval kompletne prepojenú sieť iba so vstupnou a výstupnou vrstvou neurónov. Keď označíme aktivity vstupných neurónov x_i , výstupných neurónov y_j, ideálne výsledné aktivity pre daný vstup $\hat{y}_j$ a váhu spojenia neurónov i a j ako w_{ij}, potom existuje známe Widrow-Hoffovo "delta" pravidlo pre najvhodnejšiu zmenu váh siete

$$\Delta w_{ij} = \eta(\hat{y}_j - y_j)x_i, \tag{11.17}$$

kde η je rýchlosť učenia (číslo z intervalu (0,1)). Chalmers sa toto pravidlo pokúsil nahradiť lineárnou funkciou všetkých premenných a ich párových násobkov.

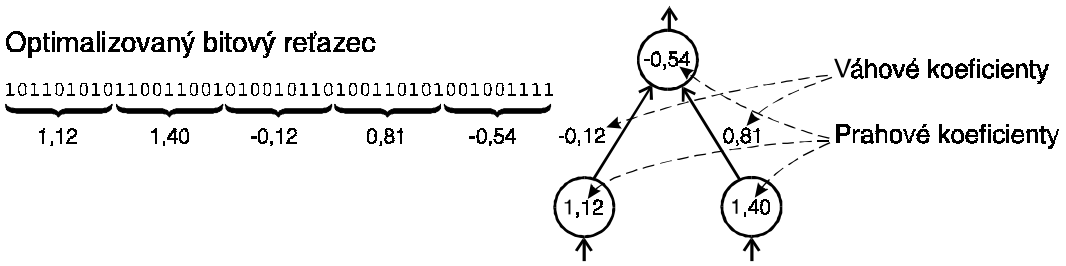
$$\Delta w_{ij} = k_0(k_1 w_{ij} + k_2 x_i + k_3 y_j + k_4 \hat{y}_j + k_5 w_{ij} x_i + k_6 w_{ij} y_j + k_7 w_{ij} \hat{y}_j + k_8 x_i y_j + k_9 x_i \hat{y}_j + k_{10} y_j \hat{y}_j) \tag{11.18}$$

Chalmers sa pokúsil optimalizovať konštanty k_0-k_{10}. Približne v jednom z desiatich pokusov skutočne dostal Widrow-Hoffovo "delta" pravidlo. Samozrejme, nie je pravdepodobné získanie účinnejšieho algoritmu ako je backpropagation, ale uvedený prístup by mohol slúžiť preučiace pravidlá pri učeniu bez učiteľa, alebo pre rekurentné siete.

Príklad 11.1. Naprogramujte sieť uvedenú na obr. 11.7, a optimalizujte pomocou genetického algoritmu váhy a prahové parametre tejto siete tak, aby s čo najmenšou chybou produkovala tabuľku č. 11.1, teda XOR, kde z numerických dôvodov namiesto 0 a 1 budeme brať 0.1 a 0.9.

Tabuľka 11.1. Upravená trénovacia množina pre XOR

vstup do neurónu		výstup z neurónu
č. 1	č. 2	č. 5
0	0	0.1
0	1	0.9
1	0	0.9
1	1	0.1



Obrázok 11.10. Zobrazenie prevodu bitového reťazca na váhové a prahové koeficienty, teda. reálne čísla. Na miesto vektoru tvoreného reálnymi číslami je potom možné optimalizovať bitový reťazec.

■ *Optimalizácia váhových a prahových koeficientov* spojení v neurónových sieťach (obr. 11.10), ktorá nahradzuje metódu spätného šírenia, je ďalšou úlohou často riešenou pomocou evolučných metód [42]. Evolučné metódy však pre túto optimalizáciu nie sú najtypickejšie, pretože klasická metóda spätného šírenia poskytuje dobré výsledky a je výpočtovo menej náročná. Klasické algoritmy optimalizujúce sumu kvadrátu odchyliek založené na derivácii "chybovej" (účelovej) funkcie automaticky končí v najbližšom lokálnom optimu. Našťastie hyperplocha optimalizovanej funkcie väčšinou nemá príliš mnoho lokálnych miním a výsledky metódy spätného šírenia sú väčšinou prijateľné. Pokiaľ sa optimalizácia nedarí ani po niekoľko "nástreloch" počiatočných váh, stačí často pridať niekoľko skrytých neurónov. Keď ale chceme vytvoriť efektívnu sieť s minimom vnútorných neurónov a spojov, sú evolučné algoritmy nenahraditeľné. Používajú sa tiež u problémov, kde nie sú k dispozícii korektné odpovedi jednotlivých výstupných jednotiek siete na daný výstup, čo sa vyskytuje väčšinou u problémov riadenia [43,44] alebo navigácie robotov v neznámom prostredí. (V týchto prípadoch je ale potrebné používať zakaždým iné štartovacie stavy, aby boli uniformne rozložené. Treba si tiež dávať pozor nie len na rýchlosť učenia, no aj na schopnosť zovšeobecňovania, aby sieť nefungovala iba na zopár naučených príkladoch.) Evolučné metódy dokážu relatívne rýchlo vyhľadať váhové a prahové koeficienty blízke optimálnym, trvá im však dlho prechod od takmer optimálnych váh k optimálnym váham [45,46]. Tento nedostatok sa často nahradzuje spojením evolučného algoritmu s metódou spätného šírenia — evoluční algoritmus nájde približné hodnoty optima, a spätné šírenie dokončí optimalizáciu do globálneho optima [47,48].

Jednou z hlavných podmienok úspešného použitia evolučných algoritmov je rýchlosť výpočtu hodnôt účelové funkcie v danom bode. Špeciálne táto posledná podmienka podstatne limituje úspešné použitie stochastických optimalizačných metód pre optimalizáciu neurónových sietí, ich relatívna jednoduchosť v porovnaní s gradientovými metódami je "kompenzovaná" náročnosťí na výpočtový čas.

Pre optimalizáciu topológie neurónových sietí sa zatiaľ prakticky výhradne používajú genetické algoritmy, pre optimalizáciu váhových a prahových koeficientov sa používajú genetické algoritmy a simulované žíhanie.

Nevýhodou genetických algoritmov pri aplikácii na topológiu neurónových sietí sú problémy so zmysluplnou výmenou informácií pri krížení. Konkrétne ide o váhy v neurónovej sieti. Povedzme, že máme štyri vstupné a štyri skryté neuróny. Podarilo sa nám vygenerovať sieť, v ktorej sú spojené vstupní neuróny nenulovými váhami iba s prvými dvoma skrytými neurónmi, teda. druhé dva skryté neuróny môžeme zanedbať. Vzhľadom k tomu, že sieť je v podstate symetrická, môže existovať rovnako dobrá typológia zanedbávajúci prvé dva skryté neuróny, a používajúci iba druhé dva skryté neuróny. Pri krížení potom môže nastať situácia, že prvý potomok bude používať všetky štyri skryté neuróny, a druhý potomok bude mať všetky váhy nulové. Existujú rôzne spôsoby, ako predchádzať tomuto problému krížením iba viac-menej podobných jedincov - sietí, ale žiaden z nich sa nedá považovať za uspokojivý. Pre n skrytých neurónov totiž existuje $n!$ permutácií ich postavení a teda $n!$ ekvivalentných sietí. Okrem tohoto druhu symetrie existuje i symetrie u váh skrytých neurónov. Keď je prechodová funkcia nepárna (čo väčšinou je), potom môžeme nahradiť všetky znamienka vstupných a výstupných váh skrytého neurónu opačnými znamienkami, a výstupy siete sa nezmenia. Pre n skrytých neurónov tak máme 2^n štruktúrne odlišných, ale funkčne identických sietí generovaných takýmto prehodením znamienok. Pri krížení týchto sietí potom dochádza k nelogičnostiam, pretože sa môže ľahko stať, že polovinu váh neurónu vezmeme z jednej siete, polovinu z druhej siete (kde boli opačná znamienka) a výsledkom je niečo, čo nebolo ani v jednej sieti. Dochádza tak skôr k mutácii, ako k výmene informácií. Celkovo je teda priestor váh 2^n $n!$ väčší, ako by v skutočnosti mal byť. Príkladom pokusu o elimináciu tejto redundancie je kríženie váh s rovnakým znamienkom u dvojíc neurónov s rovnakým počtom kladných váh a záporných váh. Počet odpovedajúcich dvojíc sa dá zvýšiť prípadným prehodením všetkých znamienok váh u neurónu.

Tieto problémy sa väčšinou zmenšujú zvýšením pravdepodobnosti mutácie a znížením pravdepodobnosti kríženia u genetického algoritmu. Dá sa tiež použiť evolučného programovania, čo je v podstate genetický algoritmus bez kríženia, prípadne s gaussovskými mutáciami (viď kap. 9) [49].

11.3. Optimalizácia pomocou neurónových sietí

Algoritmy inšpirované teóriou neurónových sietí môžeme použiť nielen na predikciu a klasifikáciu [1-6], no aj na **optimalizačné problémy**. Napríklad na neurónovú sieť Hopfieldovho typu alebo na Boltzmannov stroj sa môžeme pozerať ako na masívne paralelné algoritmy, ktorých cieľom je minimalizácia energie systému. Energia systému tu vlastne zastupuje účelovú (cenovú, objektívnu, ohodnocovaciu, kriteriálnu) funkciu, ktorej hodnota je optimalizovaná. U týchto sietí vlastne nejde o principiálne nové optimalizačné algoritmy, používaná optimalizácia je u Boltzmanovho stroja simulované žíhanie, u Hopfieldovej siete gradientová metóda. Podstatná je v týchto príkladoch formulácia problému, ktorá je zakomponovaná do topológie siete a do funkcie energie systému. Na každý typ problému je treba sformulovať ľahko odlišný predpis na zostrojenie siete a funkcie energie. Vo všeobecnosti je funkcia energie zostavená z dvoch členov:

E="hodnota vlastnej optimalizovanej funkcie"
+ "pokuta za prekročenie ohraničujúcich podmienok" (11.19)

Myšlienka Hopfieldových sietí a Boltzmannových strojov je založená na Isingovom modeli spinového skla [50,51], ktorý bol použitý na skúmanie kolektívnych vlastností fyzikálnych systémov zložených z veľkého množstva jednoduchých prvkov.

Netypickou možnosťou je využitie Kohonenovej SamoOrganizujúcej sa Mapy (SOM) pre riešenie problému obchodného cestujúceho. Táto sieť sa ale nevyužíva na optimalizáciu iných problémov, a aj jej použitie pre problém obchodného cestujúceho nie je typické. Preto sa jej budeme venovať až na konci tejto kapitoly.

Pomocou Hopfieldových sietí a Boltzmannových strojov sa riešia jednak kombinatorické optimalizačné problémy, ale aj spracovanie obrazu (angl. image processing), čiže rekonštrukcia objektu zo zašumeného alebo rozmazaného obrazu. Treba zdôrazniť, že v prípade spracovania obrazu nejde o vybavenie si podobného obrazu z pamäti (aj také problémy sa riešia typicky Hopfieldovou neurónovou sieťou), ale o interpoláciu povrchov objektov [18], detekciu hrán [1,19], rekonštrukciu trojrozmerného tvaru objektov z ich tieňovania alebo binokulárnej disparity [16,17,52], rekonštrukciu tvaru objektov na základe farby alebo pohybu, a pod. Pretože spracovanie obrazu nie je typickým optimalizačným problémom, aj keď sa k nim v súvislosti s neurónovými sieťami niekedy zaraďuje, a samotná problematika je dosť špecializovaná, nebudeme sa spracovaním obrazu ďalej zaoberať. Problematike spracovania obrazu je podrobnejšie venovaná kniha [53] (bez použitia neurónových sietí), alebo kniha [54] (viacvrstvové siete s dopredným šírením).

Neurónové siete sa dajú použiť aj pre minimalizáciu lineárnych alebo kvadratických funkcií s obmedzeniami premenných vyjadrenými pomocou lineárnych rovníc a nerovníc, alebo aj pre nelineárne programovanie, kedy optimalizované funkcie a ohraničujúce podmienky pre premenné sú nelineárne. V týchto prípadoch sa ale väčšinou vyžaduje, aby optimalizované funkcie boli hladké a diferencovateľné. Pokiaľ nie sú, musíme problémy na optimalizáciu hladkých diferencovateľných funkcií pretransformovať. Podrobne je táto tematika uvedená v knihe [55], tu sa týmito prístupmi nebudeme zaoberať, lebo sú pre svoju neefektívnosť zaujímavé (aspoň zatiaľ) iba z teoretického hľadiska. Efektívne by sa tieto prístupy mohli stať iba pri hardwarovej implementácii.

11.3.1 Kombinatorická optimalizácia pomocou Boltzmanovho stroja a Hopfieldovej siete

U optimalizačných problémov je základom zostaviť sieť a ohodnocovaciu funkciu tak, aby optimálny stav siete odpovedal optimálnemu riešeniu. Základným problémom je napríklad optimalizácia permutácie. Problémom tu je už dostať samotnú permutáciu, bez ohľadu na jej ďalšie ohodnotenie. Tento problém je riešený na obrázku 11.11, kde optimalizácia znamená zapínanie a vypínanie uzlov siete - pravoúhlej mriežky. Ide vlastne o generovanie permutačnej matice, kde zapnutý uzol predstavuje jednotku, vypnutý nulu a v každom riadku alebo stĺpci by mala byť práve jedna jednotka. Uzly sú ohodnotené hranami a slučkami, pričom hrany spájajú všetky uzly v rovnakom riadku aj všetky uzly v rovnakom stĺpci, iba šikmé hrany sa nevyskytujú. Hrany sú ohodnotené záporným číslom, slučky kladným číslom. Minimalizuje sa tu suma

$$E = \sum_{x,i}^{n} \sum_{y,j}^{n} u_{xi} u_{yj} w_{xi,yj} \tag{11.20}$$

kde u_{xi} =0, keď je prvok U_{xi} vypnutý, a u_{xi} =1, keď je prvok U_{xi} zapnutý, a $w_{xi,yj}$ znamená váhu medzi dvoma prvkami (alebo slučku, keď *xi=yj*). Počet prvkov v permutácii je *n*. Váhy sú nastavené tak, že medzi uzlami v riadku alebo stĺpci sú záporné *–p*, aby súčasne neboli "zapnuté" prvky v rovnakom riadku alebo stĺpci, a zároveň slučku majú kladné ohodnotenie *b*, aby vždy jeden prvok v každom riadku alebo stĺpci bol zapojený. Keďže *b* a *p* sú kladné čísla, musí platiť, že *p* > *b*, aby zapnutie druhého uzla v rovnakom riadku alebo stĺpci nezvýšilo hodnotu maximalizovanej funkcie *E*. Sumácie prebiehajú cez všetky uzle. Týmto sa ale dostávame iba k permutácii, ktorá môže vyjadrovať riešenie spĺňajúce základné obmedzenia. Ohodnotenie permutácie je ďalším problémom. Keď riešime napríklad problém obchodného cestujúceho, hľadajúceho najkratšiu dráhu, ktorá prechádza práve raz cez každé mesto a vracia sa do štartovného mesta, musíme minimalizovať sumu vzdialeností medzi mestami na dráhe. Pretože ale používame maximalizáciu, označíme vzdialenosti medzi mestami opačným znamienkom. Pre mestá *x* a *y* označíme spojnice uzlov U_{xi} a U_{yj} váhou $-d_{xy}$. Tieto nové spojnice budú teda existovať medzi všetkými uzlami v susedných stĺpcoch, a medzi všetkými uzlami medzi prvým a posledným stĺpcom. Konštanta *b* pritom musí byť väčšia ako dvojnásobok najdlhšej vzdialenosti medzi mestami, $b>2d_{MAX}$. Funkcia (11.20) sa teda bude počítať cez všetky váhy, *b*, *p* aj *d*.

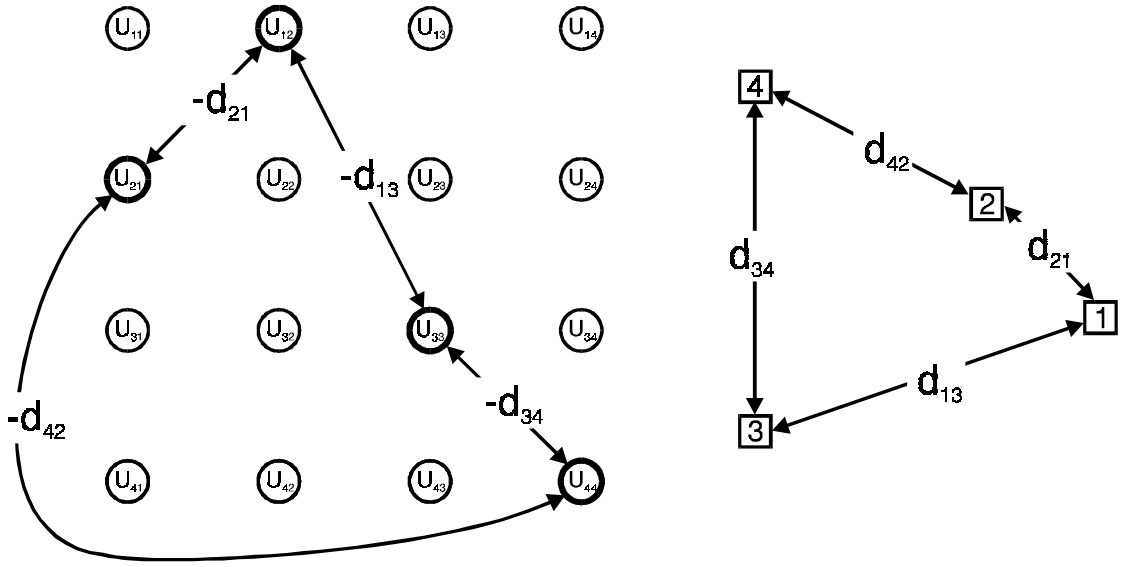
Týmto spôsobom sme ale previedli iba optimalizáciu permutácie na optimalizáciu siete, no nepovedali sme, ako sieť optimalizovať. Pokiaľ použijeme ako mutáciu alebo perturbáciu "zapnutie" alebo "vypnutie" uzla, a nový stav siete akceptujeme na základe simulovaného žíhania (viď kapitola 7), máme Boltzmannov stroj, zavedený pôvodne Hintonem a Sejnowskim [56].

Pokiaľ použijeme rýchle simulované žíhanie [57], založené na pridaní šumu k hodnotám funkcie navrhovaných riešení a rýchlejšom poklesu teploty, a pokiaľ namiesto Gaussovskej distribúcie použijeme Cauchyho distribúciu, dostaneme tak Cauchyho stroj [58].

Ďalším obdobným prístupom k optimalizácii je Hopfieldova sieť. Jej zostavenie je podobné ako na obr. 11.11. Nebudeme teraz ale brať do úvahy váhy *p* a *b*, budeme ale ohodnocovať jednotlivé uzly, viď obr. 11.12. Toto ohodnotenie ale nebude "zapnuté" alebo "vypnuté", no bude založené na sume zloženej z predchádzajúceho ohodnotenia, relaxačného členu spôsobujúceho stále znižovanie pravdepodobnosti zmeny stavu siete, váh odpovedajúcich vzdialenostiam medzi mestami a ďalšími členmi závislými na hodnotách ostatných uzlov. Výstupný signál z uzlu s aktivitou u_i je transformovaný väčšinou sigmoidálnou funkciou *g* s výstupom medzi nulou a jednotkou, podľa Hopfielda a Tanka [21,22],

$$v_i = g(u_i) = 0.5\,(1+\tanh(\alpha u_i)) \tag{11.21}$$

Keď vo všeobecnosti definujeme optimalizovanú funkciu (energiu) ako



Obrázok 11.11. Na ľavej strane je uvedená sieť bez váh *b* a *p*, no so zvýraznenými vybranými uzlami, ktoré sú spojené ďalšími váhami tvorenými zápornou vzdialenosťou medzi odpovedajúcimi mestami. Na pravej strane je potom reálne rozmiestnenie týchto miest, ktoré sú označené štvorčekmi. Indexy vo vnútri štvorčekov odpovedajú prvým indexom (teda riadkom) zvýraznených uzlov siete na ľavej strane.

$$E = \sum_{x,i=1}^{n,n} \sum_{y,j=1}^{n,n} v_{xi} v_{yj} w_{xi,yj} + \sum_{x,i=1}^{n,n} v_{xi} \theta_{xi} \tag{11.22}$$

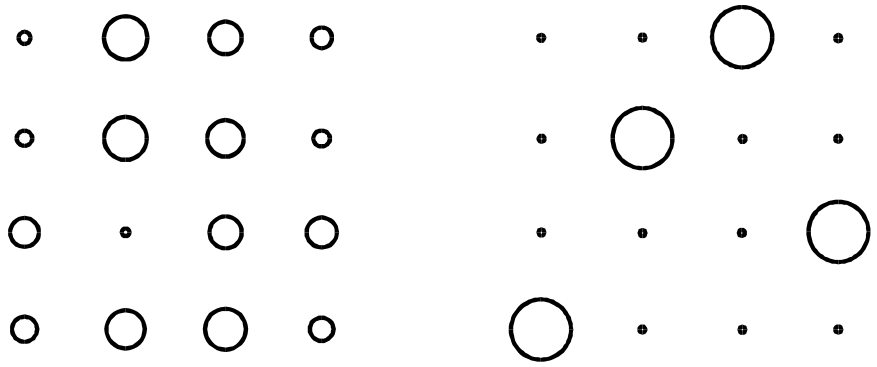
potom podľa ďalej uvedeného algoritmu bude sieť konvergovať, dokiaľ derivácia tejto funkcie podľa času bude záporná. Premenná θ_{xi} označuje externý vstup do siete, čo v prípade problému obchodného cestujúceho znamená zahrnutie vzdialeností miest a podmienky, že celkový počet zapnutých uzlov by mal byť rovný počtu miest. Aktivita jednotlivých neurónov sa v priebehu času bude teda meniť podľa diferenciálnej rovnice

$$\frac{d}{dt} u_{xi} = -\frac{u_{xi}}{\tau} + \sum_{y,j=1}^{n,n} w_{xi,yj} v_{yj} + \theta_{xi} \tag{11.23}$$

Vo väčšine obdobných rovníc sa vyskytuje iba jeden index, napr. *i* namiesto *xi*, tu sme urobili výnimku pre problém obchodného cestujúceho, kedy prvý index označuje mesto a druhý pozíciu v permutácii, teda koľké v poradí bude dané mesto navštívené. Hopfield-Tankova funkcia "energie" pre problém obchodného cestujúceho je vyjadrená pomocou štyroch častí. Prvá časť s konštantou *A* odpovedá príspevkom dvojíc uzlov v jednom rade, pokiaľ sú obidva uzly "zapnuté". Druhá časť s konštantou *B* odpovedá príspevkom dvojíc uzlov v jednom stĺpci, pokiaľ sú obidva uzly "zapnuté". Tretia časť s konštantou *C* závisí na uzloch samotných, malo by byť "zapnutých" nie viac ani nie menej ako N uzlov. A konečne posledný člen s konštantou *D* vyjadruje vlastný cieľ problému obchodného cestujúceho, teda že vzdialenosť cesty by mala byť čo najmenšia.

$$\begin{aligned} E = & \frac{A}{2} \sum_x \sum_i \sum_{j \neq i} v_{xi} v_{xj} + \frac{B}{2} \sum_i \sum_x \sum_{y \neq x} v_{xi} v_{yi} \\ & \frac{C}{2} \left[N - \sum_x \sum_i v_{xi} \right]^2 + \frac{D}{2} \sum_x \sum_{y \neq x} \sum_i d_{xy} v_{xi} (v_{y,i+1} + v_{y,i-1}) \end{aligned} \tag{11.24}$$

Parameter *N* v tejto rovnici je väčšinou zadefinovaný o niečo väčší ako počet miest *n*. S pozíciami miest transformovanými tak, aby sa vošli do štvorca 1.0×1.0 Hopfield a Tank použili pre desať miest nasledujúce hodnoty konštánt: $A=B=500$, $C=200$, $D=500$, $N=15$, $\alpha=50$. Relaxačný term spôsobujúci konvergenciu hodnôt energie bol pre vo Wilsonovej a Pawleyovej adaptácii [23] Hopfieldovho algoritmu (viď Algoritmus 11.2) stanovený na $\Delta t=10^{-5}$. Ukončovacie kritérium bolo, keď našli platnú cestu, kedy pri aktivácii *v* väčšej ako 0.9 bol uzol považovaný za zapnutý a pri aktivácii menšej ako 0.1 za vypnutý. Keď bola sieť "zmrznutá", teda žiadna aktivácia sa nezmenila viac ako o 10^{-35} alebo keď platná cesta nebola nájdená v priebehu 1000 epoch, algoritmus bol zastavený.



Obrázok 11.12. Schematické znázornenie aktivácií Hopfield-Tankovy siete pre riešenie problému obchodného cestujúceho pre 4 mestá. Podobne ako na obr. 11.11 číslo riadku predstavuje číslo mesta, číslo stĺpca predstavuje poradí mesta v permutácii. Veľkosť krúžku naznačuje veľkosť aktivácie daného uzla. Na ľavej strane sú štartovné náhodne vygenerované aktivácie, ktorých súčet má dať počet miest, teda 4. Na pravej strane je táto sieť už skonvergovaná, po "zaokrúhlení", kedy uzly s najväčšou aktiváciu berieme ako "zapnuté" a ostatné ako "vypnuté", dostávame ako výsledok permutáciu 4,2,1,3. Z tohto výsledku vyplýva, že ako prvé bude mesto č. 4, z neho pôjde cesta cez mestá č. 2, 1, 3, až sa zasa nakoniec z mesta č. 3 vráti do mesta č. 2.

Príklad 11.2. Zostrojte Hopfieldovu a Boltzmannovu sieť pre priradenie práce strojom vo fabrike. Nech je M úkolov, N strojov, cena priradenia úkolu *k* pre stroj *i* je c_{ik} . Každý stroj *i* má celkové množstvo zdrojov b_i, z ktorej sa úkolom *k* vyčerpá množstvo a_{ik}. Cieľom je minimalizovať cenu priradenia každého úkolu práve jednému stroju bez toho, že by sa prečerpal zdroj b_i tohto stroja. Tento problém sa dá formálne popísať ako minimalizácia energie *E*

Tabuľka 11.1. Vstupné hodnoty pre príklad 11.XX.

a_k	7	9	8	4	2	3	10	1

b_i	c_{ik}							
8	6	12	15	13	4	5	8	7
24	18	10	6	4	2	8	14	4
12	9	7	10	4	8	1	4	12

$$E = \sum_{i=1}^{N}\sum_{k=1}^{M} c_{ik}x_{ik} \text{ s obmedzujúcimi podmienkami } \sum_{k=1}^{M} a_{ik}x_{ik} \le b_i \quad (i = 1,\ldots,N), \tag{11.25}$$

Inicializuj aktivácie v jednotlivých uzlov náhodnými číslami tak, aby súčet všetkých aktivácií bol rovný počtu miest, $\sum_x \sum_i v_{xi} = n$.

WHILE podmienka ukončenia nie je splnená
BEGIN REPEAT 3-5 n^2 krát
BEGIN náhodne zvoľ uzol zmeň aktivitu na uzlu

$$u_{xi}(new) = u_{xi}(old) + \Delta t\Big[-u_{xi}(old) - A\sum_{j\neq i} v_{xj} - B\sum_{y\neq x} v_{yi} - C\Big\{N - \sum_x\sum_j v_{xj}\Big\} - D\sum_{y\neq x} d_{xy}(v_{y,i+1} + v_{y,i-1})\Big]$$

$$v_{xi} = 0.5[1 + \tanh(\alpha u_{xi})]$$

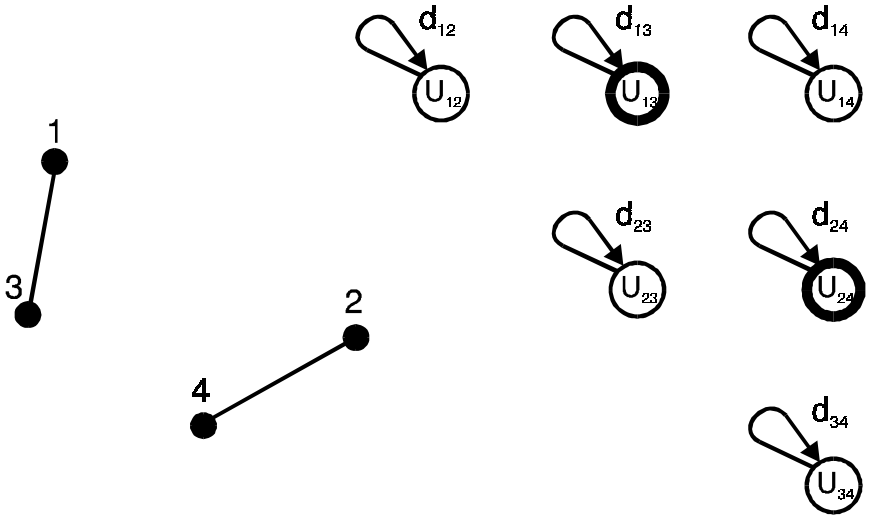
END
END

Algoritmus 11.2. Hopfield-Tankov algoritmus na riešenie problému obchodného cestujúceho.

$$\sum_{i=1}^{N} x_{ik} = 1 \quad (k = 1,\ldots,M)$$

kde x_{ik} = 0 alebo 1. Simulujte na počítači pre dáta v tabuľke 11.1, kde N=3, M=8, a $a_{ik} = a_k$ pre všetky *i*. (Každý uzol u_{ik} v sieti bude odpovedať priradeniu úkolu *k* pre stroj *i*, taký uzol môže byť alebo zapnutý alebo vypnutý). Ideálne optimum je vyznačené podčiarknutými hodnotami c_{ik}, suma týchto hodnôt je 57. V ideálnom prípade by teda mali byť aktivované uzly odpovedajúce podtrhnutým pozíciám.

Hopfieldov-Tankov prístup k riešeniu problému využívajúci gradientu k riešeniu kombinatorického problému je síce originálny, no v praxi zatiaľ nepoužiteľný, pretože pre *n* miest je potrebných n^2 uzlov. Rovnaký problém má Boltzmanov stroj. Daný algoritmus funguje s problémami pre rádovo desať až dvadsať vrcholov, zatiaľ čo čisto simulované žíhanie s mutáciou tvorenou prehodením pozícií v permutácii alebo ešte špecializovanejšie prístupy Lina [59] fungujú pre stovky miest, dávajúc pritom výsledky blízke optimálnym alebo optimálne. Riešenie kombinatorických optimalizačných problémov pomocou neurónových sietí začne byť zaujímavé až pri ich hardwarovej implementácii na VLSI.



Obrázok 11.13. Problém váhového priradenia. Na ľavej strane je graf zložený zo štyroch vrcholov, kde sú vybrané také dvojice vrcholov, aby súčet vzdialeností vrcholov v dvojiciach bol čo najmenší. Na pravej strane je odpovedajúca sieť, kde indexy u uzlov označujú čísla vrcholov hrany, ktorej uzol náleží, a zvýraznené sú tu tie dva uzly, ktoré majú ohodnotenie 1 a podľa ktorých sme priradili hrany grafu naľavo.

Podobne sú prakticky nepoužívané aj modifikácie sietí pre ďalšie kombinatorické optimalizačné problémy. Pri týchto problémoch hľadáme riešenie v množine veľkého množstva možných kombinácií základných prvkov systému. Celkový počet riešení pre veľkosť problému s počtom prvkov n je obvykle exponenciálnou funkciou n, a tomu je úmerný aj čas potrebný na vyriešenie problému. Konkrétne ide napr. o **problém váhovaného priradenia** (angl. ***weighted matching problem***) XX. Pri probléme váhovaného priradenia máme v priestore množinu n bodov, pričom poznáme vzdialenosti medzi jednotlivými dvojicami týchto bodov d_{ij}. Tieto body sa môžu nachádzať v euklidovskom priestore a d_{ij} môže reprezentovať Euklidovu vzdialenosť, alebo tieto body môžu byť abstraktné entity a hodnoty d_{ij} môžu reprezentovať ich vzťahy. Vo všeobecnosti sa môžu d_{ij} považovať za nezávislé náhodné premenné s pravdepodobnostnou distribúciou P(d_{ij}). Našou úlohou je pospájať dvojice bodov tak, aby každý bod bol spojený len s jedným iným bodom tak, aby celková dĺžka spojení bola minimálna, viď obr. 11.13. Praktickými príkladmi takéhoto problému môže byť spájanie prvkov v nejakom elektronickom zariadení, optimálne mapovanie procesov na dva ekvivalentné procesory, priraďovanie žiakov do škôl, a pod. Každému páru bodov i a j, priraďme prvok u_{ij} , $i < j$. Nech u_{ij} = 1, keď medzi i-tym a j-tym bodom existuje spojenie a u_{ij} = 0, keď medzi nimi spojenie nie je. Sám so sebou sa bod nemôže spojiť, takže $u_{ii} = 0$ a pre $j < i$ platí $u_{ij} = u_{ji}$. Funkcia, ktorej minimum hľadáme je celková dĺžka spojení

$$L = \sum_{i<j} d_{ij} u_{ij} \ . \tag{11.26}$$

Túto funkciu budeme musieť trochu modifikovať, lebo každé riešenie musí spĺňať jedno obmedzenie, a síce, že každý jeden bod môže byť spojený iba s jedným ďalším bodom, a teda $\sum_j u_{ij} = 1 \text{pre} \ \forall i$. Dodržiavanie tohto obmedzenia sa rieši penalizáciou v objektívnej funkcii. Ako základ pre objektívnu funkciu zoberieme (11.26) a pridáme k nej člen, ktorý bude rásť priamo úmerne porušeniu danej podmienky. Objektívna funkcia E bude rovná:

$$E = \sum_{i<j} d_{ij} u_{ij} + \frac{\gamma}{2} \sum_{i=1}^{n} \left(1 - \sum_{j=1}^{n} u_{ij} \right)^2 \ . \tag{11.27}$$

Veľkosť konštanty γ by mala byť asi taká ako priemerná hodnota d_{ij}. Pri použití Boltzmannovho stroja náhodne zapíname alebo vypíname uzly, pričom pravdepodobnosť, že hodnota ľubovoľného prvku u_{ij} sa zmení, je rovná

$$P(u_{ij} \to u'_{ij}) = \frac{1}{1 + \exp(\beta \Delta E)} \quad kde \quad \Delta E = E(u'_{ij}) - E(u_{ij}) \ . \tag{11.28}$$

Toto pravidlo hovorí, že zmeny, ktoré minimalizujú objektívnu funkciu E sú pravdepodobnejšie ako tie, ktoré ju zvyšujú. Hľadáme teda takú distribúciu u_{ij}, ktorá zodpovedá jednému z miním (11.27). Problém môžeme riešiť aj pomocou analógového systému, vtedy u_{ij} blízke 1 (0) budeme považovať za rovné 1 (0).

Posledným príkladom kombinatorického optimalizačného problému je **delenie grafov** (angl. ***graph bipartitioning***) [24], viď. obr. 11.14. Predstavme si, že navrhujeme čip s N prvkami, ale všetky sa nám naň nezmestia. Potom by sme chceli urobiť dva čipy, tak aby polovica prvkov bola na jednom a polovica prvkov na druhom a aby počet spojení medzi týmito dvoma čipmi bol minimálny. Uvažujme všeobecný graf, t.j. množinu N bodov, vrcholov grafu pospájaných hranami, ktoré spájajú dvojice vrcholov. Nech je N párne. Nech p je fixná pravdepodobnosť, že každý vrchol je spojený s iným. Priemerný počet vrcholov pN, ktoré sú navzájom spojené, sa nazýva valencia grafu. Našou úlohou je rozdeliť vrcholy do dvoch rovnako veľkých množín, medzi ktorými je minimálny počet hrán. Definujme si $c_{ij} = 1$, keď sú vrcholy i a j spojené hranou, a $c_{ij} = 0$, keď nie sú spojené. Ďalej si pre každý vrchol definujme premennú $S_i = +1$, keď sa vrchol nachádza v prvej množine a $S_i = -1$, keď sa vrchol nachádza v druhej množine. Chceme minimalizovať funkciu

$$L = -\sum_{\langle ij \rangle} c_{ij} S_i S_j \ , \tag{11.29}$$

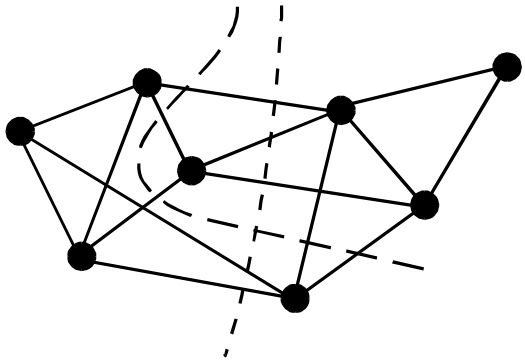
kde $\langle ij \rangle$ znamená indexovanie cez každý pár vrcholov zvlášť. Premenné S_i podliehajú obmedzeniu $\sum_i S_i = 0$. V termínoch teórie magnetických látok tieto rovnice zodpovedajú feromagnetu s nulovou celkovou magnetizáciou. Každú celkovú magnetizáciu, ktorá sa vzďaľuje od 0 budeme v objektívnej funkcii penalizovať, takže objektívna funkcia systému je

$$\mathrm{E} = -\sum_{\langle ij \rangle} c_{ij} S_i S_j + \mu (\sum_i S_i)^2 \ . \tag{11.30}$$

Hodnotu μ vyberáme v intervale 0 až 1/2. Rovnica (11.30) pripomína energiu systému analogického spinovému sklu. Systém naštartujeme z náhodnej nekorelovanej konfigurácie $\boldsymbol{S}$ a necháme relaxovať, pričom S_i sa menia podľa stochastického pravidla (11.28) s S_i namiesto u_{ij}, kde energia E je vyjadrená ako (11.30). Opäť bude užitočné použiť simulované žíhanie.

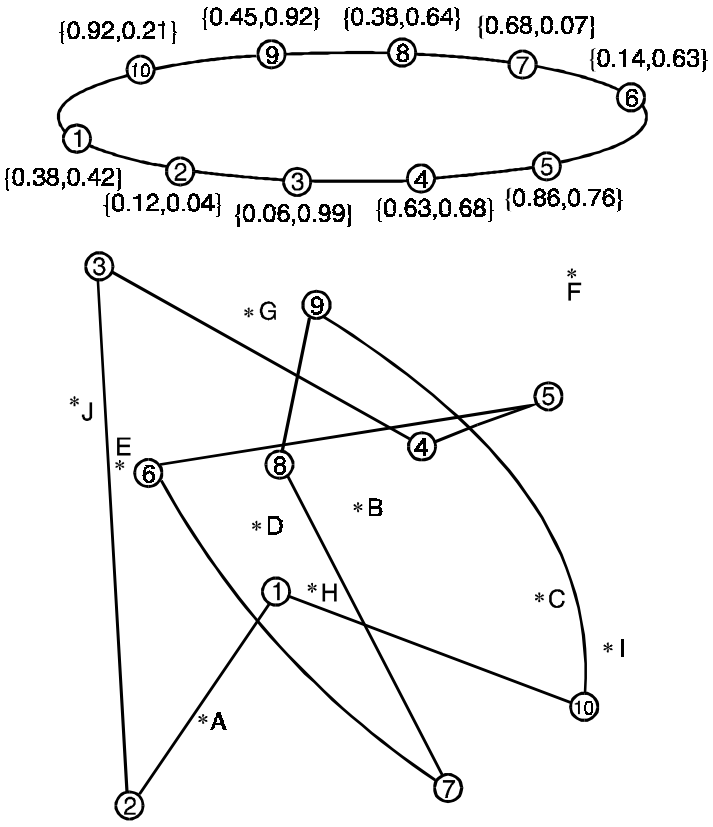
11.3.2 Riešenie problému obchodného cestujúceho pomocou Kohonenovej siete

Kohonenova SOM (SamoOrganizujúca sa Mapa) [60] patrí do kategórie sietí, ktoré sa učia bez učiteľa (angl. unsupervised learning). To znamená, že algoritmus učenia nemá informáciu, aké váhy u jednotlivých prvkov siete sú správne, na rozdiel od dopredných sieti v prvej časti kapitoly, ktoré pre trénovaciu množinu mali spolu



Obrázok 11.14. Dve možné delenia vrcholov grafu znázornené prerušovanou čiarou na dve podmnožiny vrcholov tak, aby medzi týmito podmnožinami existovalo čo najmenej hrán. V obidvoch rozdeleniach existuje päť hrán, spájajúcich podmnožiny na pravej a na ľavej strane prerušovanou čiary. V danom prípade vrcholy môžu priamo predstavovať uzly siete, ohodnotené S_i=0 keď sú priradené jednej množine a S_i=1 keď sú priradené druhej množine.

so vstupným vektorom vždy aj ideálny výstupný vektor. Základnou črtou Kohonenových sietí je schopnosť realizovať zobrazenie zachovávajúce topológiu trénovacej množiny dát. To znamená, že body (vstupné vektory), ktoré si boli podobné svojimi hodnotami, budú namapované na uzly, ktoré budú "ležať" blízko seba. Ležať blízko seba bude závislé na topológii siete, ktorá je typicky alebo lineárna - to znamená, že uzly sú formálne usporiadané do reťazca, alebo mriežková, kedy sú uzly usporiadané v dvojrozmernej pravouhlej mriežke. Susednosť v takýchto sieťach sa meria iba počtom uzlov, cez ktoré treba preskákať, aby sme sa dostali od jedného uzla k druhému, nejde teda o euklidovskú vzdialenosť. Uzly v reťazci majú dvoch najbližších susedov - naľavo a napravo, uzly v pravouhlej mriežke (vprostred pomyseľného štvorca) majú ôsmich najbližších susedov - hore, dole, naľavo, napravo (vprostred pomyseľných hrán štvorca), a naprieč na uhlopriečkach (na pomyseľných rohoch štvorca). Každý z týchto bodov je ohodnotený vektorom (spočiatku náhodne vygenerovaným) rovnakého typu ako sú vstupné vektory. Pri zadávaní sa pre každý vstupný vektor nájde "víťazný" uzol, ktorého vektor má od daného vstupu najmenšiu (väčšinou euklidovskú) vzdialenosť. Ohodnotenie tohto uzla sa potom zmení podľa tzv. Hebbovho pravidla učenia, kedy sa hodnoty vektoru uzla



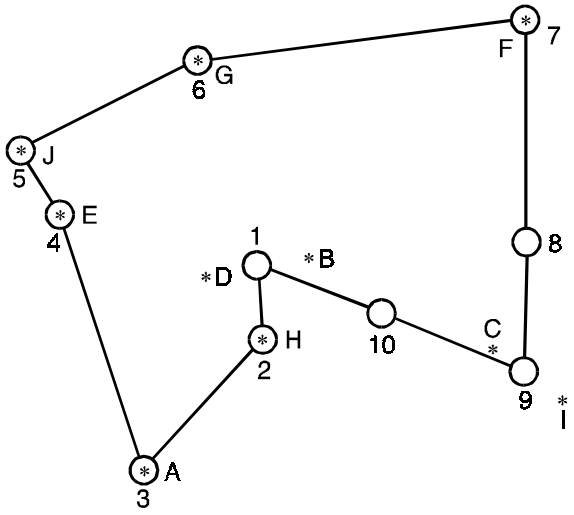
Obrázok 11.15. Riešenie problému obchodného cestujúceho pomocou Kohonenovej siete. V hornej časti je zobrazená topológia siete, uzly sú označené krúžkami a očíslované od 1 po 10, u každého uzlu je jemu prislúchajúci náhodne vygenerovaný štartovný vektor $\boldsymbol{w}$ súradníc {X,Y}. V dolnej časti sú jednotlivé uzly umiestnené v ploche so súradnicami odpovedajúcimi ich vektoru. Rovnako sú v ploche umiestnené aj mestá, znázornené hviezdičkami a označené A,B,C,D,E,F,G,H,I,J. K týmto mestám by sa mali v priebehu algoritmu približovať jednotlivé uzly.

zmenia tak, aby euklidovská vzdialenosť od daného vstupu bola menšia. Podobne sa rovnako danému vstupnému vektoru môžu priblížiť aj hodnoty vektorov uzlov, ktoré sú s "víťazným" uzlom susedné. U lineárnej topológie to budú susedné uzly vľavo a vpravo. Môžeme uvažovať aj širšie okolie zahŕňajúce viac uzlov vľavo a vpravo od víťazného uzla, no čím vzdialenejšie okolie, tým menej by sa hodnoty vektorov týchto uzlov mali priblížiť vstupnému vektoru. To je zohľadnené v algoritme 11.3 funkciou $h(I_{min},i)$, kde táto funkcia určuje, v akom okolí víťazného uzla budú adaptované ostatné uzly a do akej miery. Najjednoduchšia používaná funkcia je

$$h(I_{\min}, i) = \begin{cases} 1 & \text{ak } d_M(I_{\min}, i) \le 1 \\ 0 & \text{v ostatných prípadoch} \end{cases} \qquad (11.31)$$

kde $d_M(I_{min},i)$ je vzdialenosť typu Manhattan medzi neurónmi I_{min} a i, teda počet hrán, po ktorých treba v danej topológii siete prejsť, aby sme sa dostali od neurónu I_{min} k neurónu i. Niekedy sa pre vzdialenejšie okolie volí dokonca "záporné" učenie, teda že vektory uzlov vzdialených cez viac hrán od víťazného uzla sa "odďaľujú" od hodnôt vstupného vektora, zatiaľ čo ešte viac vzdialené uzly už ostávajú bezo zmeny. Tento proces učenia sa opakuje stále dookola. Veľkosť okolia a parameter učenia (teda miera priblíženia vektora víťazného uzla vstupnému vektoru) sa s časom v priebehu algoritmu znižujú.

Kohonenova sieť sa typicky používa na klastrovanie, kedy sa v "naučenej" sieti vždy niekoľko vstupov namapuje na jeden uzol. O týchto vstupoch a im odpovedajúcich objektoch potom hovoríme, že patria do jedného klastera, a obecne sa predpokladá, že objekty v jednom klastri by mali mať podobné vlastnosti.



Obrázok 11.16. Na obrázku je výsledok riešenia problému obchodného cestujúceho, kde počiatočný stav siete bol znázornený na obr. 11.15. Vidíme, že štyrom z desiatich miest nebolo priradené jednoznačné poradie, pretože uzly 8 a 10 nie sú víťazné pre žiadne mesto, zatiaľ čo uzol 9 je víťazný pre dve mestá C a I, a rovnako uzol 1 je víťazný pre dve mestá B a D. Pretože u týchto dvojíc miest sieť neurčila poradie, akékoľvek poradie v dvojiciach je prípustné a výsledkom sú teda štyri cesty: BDHAEJGFCI, DBHAEJGFCI, BDHAEJGFIC a DBHAEJGFIC.

Pre riešenie problému obchodného cestujúceho sa používa topológie lineárneho reťazca so spojenými koncami (takže formálne kruhu), kde počet uzlov by mal byť rovný počtu miest [27]. Vstupné vektory budú dvojrozmerné, tvorené súradnicami (x,y) jednotlivých miest. Počiatočné hodnoty vektorov jednotlivých uzlov sú vygenerované náhodne v rovnakom rozmedzí hodnôt ako sú súradnice miest (niekedy sa tiež používa sústredenia týchto počiatočných hodnôt v strede štvorca vymedzujúceho súradnice miest), viď. obr. 11.15. Vektory jednotlivých uzlov sa postupne približujú hodnotám súradníc miest a v ideálnom prípade na konci učenia je každému miestu priradený jeden uzol (množina miest je "rozklastrovaná" až na jednotlivé mestá), a postupnosť hrán medzi uzlami odpovedá postupnosti miest s najkratšou vzdialenosťou cesty, viď obr. 11.16. Niekedy sa môže stať, že dve mestá sú namapované na jeden uzol, a iný uzol nie je "víťazný" ani pre jedno mesto. V takom prípade je poradie miest namapovaných na jeden uzol nejednoznačné, budú síce umiestnené vedľa seba, no nevieme, ktoré ako prvé. Ako výsledok môžeme zobrať dve cesty líšiace sa poradím týchto dvoch miest. Ako je vidno z obr. 11.16, pokiaľ nastane taký prípad viackrát, situácia sa komplikuje. Z uvedeného príkladu je vidno, že použitie Kohonenovej siete pre problém obchodného cestujúceho je zaujímavé iba z teoretického hľadiska. No Kohonenovej siete sa v praxi využíva [60], napr. pri rozpoznávaní vzorov (hlások reči), v robotike (transformácia súradníc) alebo pri kompresii obrazov.

```
Inicializuj váhy w_ij , kde vektor w_i=( w_i1 w_i2) je vektorom priradeným uzlu i.
Váhy sú generované náhodne v rozmedzí maximálnych a minimálnych hodnôt
súradníc (x,y) miest, pre i∈(1,...,n),  kde n je počet miest (napr. 10).
Priraď mestám topológiu cyklu, teda mesto i susedí naľavo s mestom i−1 a
napravo s mestom i+1. Mesto 1 susedí naľavo s mestom n, a opačne, mesto n
teda susedí napravo s mestom 1.
Nastav rýchlosť učenia α (bude nastavená na 0.5 a bude klesať na 0.4).
Nastav veľkosť okolia (zo začiatku 1, teda okrem víťazného uzla upravujeme
vždy váhy aj jeho ľavého a pravého suseda).
WHILE nie je splnená ukončovacia podmienka (napr. počet cyklov=200)
BEGIN FOR j=1 to n  { pre všetky vstupné vektory miest (x_j,y_j)}
      BEGIN FOR i = 1 TO n  DO  d( i ) = (w_i1−x_j)^2 + (w_i2−y_j)^2
            I_min= index i, kde d( i ) je minimálne
                  pre víťazný uzol a jeho okolie uprav váhy, teda pre
                  i={ I_min−1; I_min; I_min+1 }
            w_i1(nová):= w_i1(stará)+ α h(I_min,i) [x_j− w_i1(stará)]
            w_i2(nová):= w_i2(stará)+ α h(I_min,i) [y_j− w_i2(stará)]
      END
      uprav rýchlosť učenia (napr. α:=α (0.4/0.5)^0.005 )
      zmenši okolie uzla pri splnení zadanej podmienky (napr. pri počte
      cyklov väčšom ako    100 už budeme upravovať iba váhy víťazného uzla,
      bez susedov)
END
```

Algoritmus 11.3. Kohonenov algoritmus pre problém obchodného cestujúceho.

11.4. Záver

Neurónové siete ako predmet optimalizácie nie sú zvyčajne vhodným objektom evolučnej optimalizácie. Pri optimalizácii váh a prahových faktorov viacvrstvovej doprednej siete sa väčšinou uplatní gradientové metódy lepšie a evolučné metódy tu slúžia výhodne iba pre generáciu štartovných hodnôt pre gradientové metódy. Budúce uplatnenie pre evolučné metódy pri optimalizácii neurónových sietí je teda v prípadoch, kedy je gradient ťažko použiteľný - u rekurentných sietí a pri učení bez učiteľa (keď nepoznáme, aké majú byť presné hodnoty na výstupu), čo je väčšinou pri problémoch riadenia.

Optimalizácia topológie je zaťažená nutnosťou ďalej optimalizovať váhy pre každú navrhnutú topológiu. Preto sa pre tieto účely používajú rôzne heuristiky a navrhujú úzko špecializované algoritmy, ktoré sa svojím smerovaním k optimu podobajú skôr hillclimbingu ako evolučným algoritmom.

Neurónové siete ako optimalizujúce metódy sa okrem špecializovaných problémov pri spracovaní obrazov prakticky budú dať používať iba v hardwarovej implementácii, ináč sú na praktické použitie príliš pomalé.

Literatúra

[1] V. Kvasnička, Ľ. Beňušková, J. Pospíchal, I. Farkaš, P. Tiňo, A. Kráľ: *Úvod do teórie neurónových sietí*. IRIS, Bratislava, 1997.

[2] P. Sinčák, G. Andrejková: *Neurónové siete I, Neurónové siete II*. elfa, Košice, 1996.

[3] M. Šnorek, M. Jiøina: *Neuronové sítì a neuropoèítaèe*. Skripta vydavatelství ÈVUT Praha, 1996.

[4] M. Novák: *Neuronové sítì a neuropoèítaèe*. Edice Výbìr, SENZO a.s., Praha, 1992.

[5] M. Novák, J. Fáber, O. Kufudaki: *Neuronové sítì a informaèní systémy živých organizmù*. Grada, Praha, 1992.

[6] J. Šíma, R. Neruda: *Teoretické otázky neuronových sítí*. MATFYZPRESS, vydavatelství Mat.-fyz. fakulty Univerzity Karlovy, Praha, 1996.

[7] J.D. Schaffer, D. Whitley, and L. Eshelman: Combination of Genetic Algorithms and Neural Networks: The state of the art. In *Combination of Genetic Algorithms and Neural Networks*, D. Whitley and J.D. Schaffer, eds., IEEE Computer Society Press, 1992.

[8] D. Whitley: Genetic Algorithms and Neural Networks. In *Genetic Algorithms in Engineering and Computer Science*, G. Winter, J. Periaux, M. Galan, P. Cuesta, eds., J. Wiley, 1996.

[9] E.J. Chang, R.P. Lippmann: Using genetic algorithms to improve pattern classification performance. In *Advances in Neural Information Processing 3*, R.J. Lippman, J.E. Moody, and D.S. Touretsky, eds., Morgan Kaufmann, San Mateo, Ca, 1991, pp. 797-803.

[10] F.Z. Brill, D.E. Brown, W.N. Martin: Fast genetic selection of features for neural network classifiers. *IEEE Transactions on Neural Networks*, **3** (2) (1992) 324-328.

[11] R.J. Mitchell, J.M. Bishop, and W. Low: Using a genetic algorithm to find the rules of a neural network. In *Artificial Neural Nets and Genetic Algorithms*, R.F. Albrecht, C.R. Reeves and N.C. Steele, eds., Springer Verlag, Wien, 1993, pp. 664-669.

[12] R.C. Eberhart, R.W. Dobbins: Designing neural network explanation facilities using genetic algorithms. In *IEEE international joint conference on neural networks.* IEEE Computer Society Press, Singapore, 1991, pp. 1758-1763.

[13] R.C. Eberhart: The role of genetic algorithms in neural network query-based learning and explanation facilities. In *Combination of Genetic Algorithms and Neural Networks*, D. Whitley and J.D. Schaffer, eds., IEEE Computer Society Press, 1992.

[14] J.J. Hopfield: Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, **79** (1982) 2554-2558.

[15] V.S. Dotsenko: Neural networks: translation-, rotation- and scale-invariant pattern recognition. *Journal of Physics A: Mathematics and General*, **21** (1988) L783-L787.

[16] T. Poggio and C. Koch: Ill-Posed Problems in Early Vision: From Computational Theory to Analogue Networks. *Proc. Royal Soc. Lond. B* **226** (1985) 303-323.

[17] C. Koch, J. Marroquin, and A. Yuille: Analog "Neuronal" Networks in Early Vision. *Proc. Natl. Acad. Sci. USA* **83** (1986) 4263-4267.

[18] J.M. Hutchinson and C. Koch: Simple Analog and Hybrid Networks for Surface Interpolation. In *Neural Networks for Computing*, J.S. Denker, ed., American Inst. of Physics Conf. Proc. Vol. **151**, 1987, pp. 235-240.

[19] D. Geiger and A. Yuille: A Common Framework for Image Segmentation. *International Journal of Computer Vision*, **6**(3) (1991) 227-243.

[20] J. Hertz, A. Krogh a R.G. Palmer: *Introduction to the Theory of Neural Computation*, Addison-Wesley Publ. Comp., Redwood City, 1991.

[21] J.J. Hopfield and D.W. Tank: "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, **52** (1985) 141-152.

[22] J.J. Hopfield and D.W. Tank: Computing with neural circuits: a model. *Science*, **233** (1986) 625-633.

[23] G.V. Wilson and G.S. Pawley: On the Stability of the Traveling Salesman Problem Algorithm of Hopfield and Tank. *Biological Cybernetics*, **58** (1988) 63-70.

[24] Y. Fu and P.W. Anderson: Application of statistical mechanics to NP-complete problems in combinatorial optimization. *Journal of Physics A: Mathematics and General* **19** (1986) 1605-1620.

[25] R. Durbin and D. Willshaw: An Analogue Approach to the Traveling Salesman Problem Using an Elastic Net Method. *Nature*, **326** (1987) 689-691.

[26] D. J. Burr: An Improved Elastic Net for the Traveling Salesman Problem. *IEEE International Conference on Neural Networks (2nd)*, San Diego, July 1988, pp. 69-76.

[27] B. Angeniol, G. Vaubois, J-Y. Le Texier: Self-organizing Feature Maps and the Traveling Salesman Problem. *Neural Networks*, **1**(4) (1988) 289-293.

[28] P.J. Angeline, G.M. Saunders, J.B. Pollack: An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks* **5**(1) (1994) 54-64.

[29] D.E. Rumelhart, G.E. Hinton, and R.J. Wiliams: Learning internal representation by error propagation. In *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Vol 1: Foundation,* D.E. Rumelhart, J.L. McClelland, and PDP Research Group, The MIT Press, Cambridge, MA, 1987, pp. 318-362.

[30] T. Masters: *Advanced Algorithms for Neural Networks: A C++ Sourcebook*. J. Wiley, NY, 1995.

[31] G.F. Miller, P.M. Todd, and S.U. Hedge: Designing neural networks using genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms,* J.D. Schaffer, ed., Morgan Kaufmann, San Mateo, CA, 1989, pp. 379-384.

[32] J.D. Schaffer, R.A. Caruana, L.J. Eshelman: Using genetic search to exploit the emergent behavior of neural networks. In *Emergent Computation*, S. Forrest, ed., North Holland, Amsterdam, 1990, pp. 244-248.
[33] S.A. Harp and T. Samad: Genetic Synthesis of Neural Network Architecture. In Handbook of Genetic Algorithms, L. Davis, ed., Van Nostrand Reinhold, New York, 1991, pp. 202-221.
[34] H. Kitano: Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems* **4** (1990) 461-476.
[35] F. Gruau: Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. In *COGANN-92: International Workshop on Combination of Genetic Algorithms and Neural Networks*, D.L. Whitley and J.D. Schaffre, eds., IEEE Computer Society Press, 1992.
[36] H. Kitano: Neurogenetic learning: An integrated method of designing and training neural networks using genetic algorithms. *Physica D* **75** (1994) 225-238.
[37] R.K. Belew: Interposing an ontogenetic model between genetic algorithms and neural networks. In *Advances in Neural Information Processing* (NIPS 5), S.J. Hanson, J.D. Cowan, and C.L. Giles, eds., Morgan Kaufmann, 1993.
[38] D. Whitley, F. Grunau, and L. Pyeat: Cellular Encoding Applied to Neurocontrol. In *5th Intern. Conf. on Genetic Algorithms*, L. Eshelman, ed., Morgan Kaufmann, 1995.
[39] J. R. Koza: *Genetic Programming (II): Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.
[40] J. R. Koza: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.
[41] D.J. Chalmers: The evolution of learning: An experiment in genetic connectionism. In *Proceedings of the 1990 Connectionist Model Summer School*, D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G.E. Hinton, eds., *Morgan Kaufmann*, San Mateo, CA, 1990.
[42] D.J. Montana and L.D. Davis: Training Feedforward networks using genetic algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1989.
[43] A.P. Weiland: Evolving controls for unstable systems. In *Proceedings of the 1990 Connectionist Model Summer School. Morgan Kaufmann*, D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G.E. Hinton, eds., San Mateo, CA, 1990, pp. 91-102.
[44] A.P. Weiland: Evolving neural network controllers for unstable systems. In *IEEE international joint conference on neural networks*. IEEE Computer Society Press, Seattle, WA, 1991.
[45] J.D. Schaffer, ed.: Proceedings of the *Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Los Altos, Ca, 1989, pp. 360-397.
[46] R.F. Albrecht, C.R. Reeves and N.C. Steele, eds.: *Artificial Neural Nets and Genetic Algorithms*, Springer Verlag, Wien, 1993, pp. 628-730.
[47] A. Skinner, J.Q. Broughton: Neural Networks in Computational Material Science: Training Algorithms. *Modelling and Simulation in Material Science and Engineering*, **3** (1995) 371-390.
[48] M. McInerney and P. Dhawan: Use of genetic algorithms with backpropagation in training of feedforward neural networks. *Proc. IEEE Int. Conf. Neural Networks, vol. I*, IEEE Computer Society Press, San Francisco, 1993.
[49] D.B. Fogel, L.J. Fogel, and V. W. Porto: Evolving neural networks. *Biological Cybernetics* **63** (1990) 487-493.
[50] E. Ising: Bertrag zur Theorie des Ferromagnetism, *Zeitschrift Fur Physik*, **31** (1925) 253-287.
[51] S. Kirkpatrick and D. Sherrington: Infinite-ranged models of spin-glasses. *Physical Review B*, **17** (1978) 4384-4403.
[52] H.G.E. Hentschel and A. Fine: Statistical Mechanics of Stereoscopic Vision, *Phys. Rev. A* **40** (1989) 3983.
[53] J. Hlaváè, M. Šonka: *Poèítaèové vidìní*. Grada, Praha, 1992.
[54] T. Masters: *Signal and Image Processing with Neural Networks: A C++ Sourcebook*, J. Wiley, NY, 1994.
[55] A. Cichocki, R. Unbehauen: *Neural Networks for Optimization and Signal Processing*. J. Wiley, Chichester, 1994, chapters 3 and 7.
[56] G.E. Hinton and T.J. Sejnowski: Optimal Perceptual Inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, Washington DC, 1983, pp. 448-453.
[57] H.H. Szu and R Hartley: Fast Simulated Annealing. *Physics Letters A*, **122**(3,4) (1987) 157-162.
[58] H.H. Szu: Colored Noise Annealing Benchmark by Exhaustive Solutions of TSP. *International Joint Conference on Neural Networks*, Washington DC, I (1990) 317-320.

[59] S. Lin: Computer Solutions of the Traveling Salesman Problem. *Bell Systems Technical Journal* **44** (1965) 2245-2269.
[60] T. Kohonen. *Self-Organizing Maps*. Springer, 1995.