ИНТЕЛИГЕНТНИ ТЕХНОЛОШКИ СИСТЕМИ

АТ-3 Вештачке неуронске мреже

ВЕШТАЧКЕ НЕУРОНСКЕ МРЕЖЕ

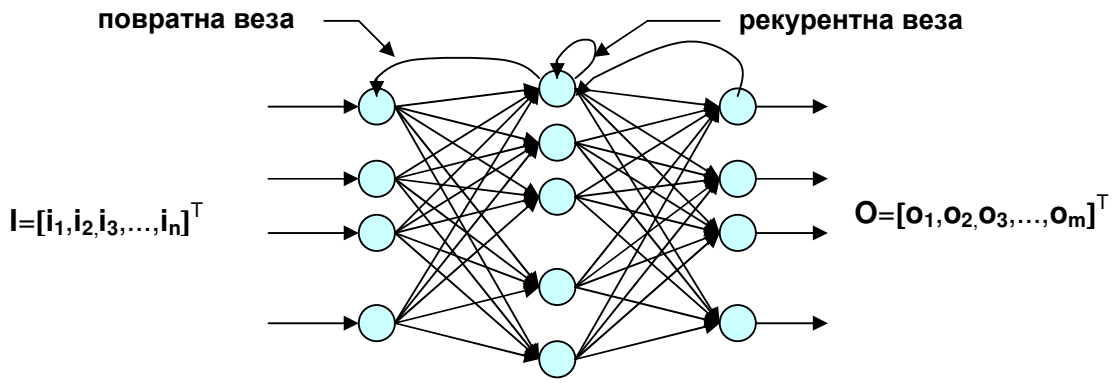
Дефиниција

Неуронска мрежа је парадигма[1] вештачке интелигенције која се дефинише као конективни[2] модел за резоновање заснован на аналогији са мозгом, уз наглашену когнитивну[3] способност да учи и врши генерализацију стеченог знања.

Вештачке неуронске мреже-основни концепти

При моделирању и управљању системима, углавном се претпоставља да општи, аналитички модел система може да се дефинише. Међутим, комплексни системи у машинском инжењерству (машине алатке, роботи, аутомобили, авиони, итд.), као и многи технолошки процеси, толико су компликовани да се општи модел ретко може успоставити. У тим случајевима **вештачке неуронске мреже** (ANN – Artificial Neural Networks) се успешно примењују за решавање неких од проблема (нпр. доношење правовремених одлука), тако да се рецимо на основу анализе технолошког процеса, без експлицитног модела, оне користе као *универзални апроксиматори*. Оне су примењиве и као „софистицирани сензори", са задатком да изврше естимацију[4] вредности сигнала или променљивих у процесу (нпр. одлучивања), уз пажљив избор скупа стабилних, квалитетних нумеричких података које верно описују сам процес. Као резултат студије под надзором **DARPA** (**D**efence **A**dvanced **R**esearch **P**rojects **A**gency) пројекта у САД, следећи закључак је понуђен и опште прихваћен:

> *Неуронске мреже обезбеђују значајне предности при решавању проблема процесирања који захтевају рад у реалном времену и интерпретацију односа између променљивих у вишедимензионалним просторима*.



Општа структура вештачке неуронске мреже

У општем смислу, неуронске мреже представљају скуп једноставних процесирајућих елемената-неурона, међусобно повезаних везама са одговарајућим тежинским односима (слика горе). Неуронске мреже имају способност адаптивног понашања према променама, кроз учење улазног узорка, што значи да могу да уче пресликавања између улазног и излазног простора и да при томе синтетизују асоцијативну меморију, која омугућава налажење одговарајућег излаза.

---

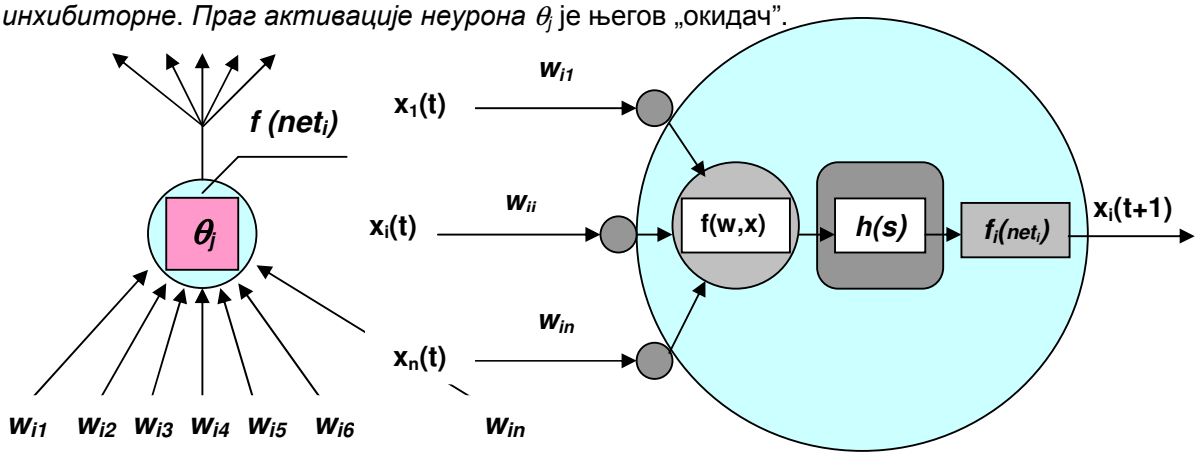
[1] Парадигма – образац, узор, пример за углед

[2] Конективан – повезан

[3] Когнитиван – који се тиче сазнања, сазнајни, спознајни

[4] Естимација – процена, процењивање, оцењивање

Вештачка неуронска мрежа или конективни модел има три основне компоненте: *неурон*, *топологију мреже* и *алгоритам учења*. Додатне компоненте су: *величина мреже* (број нивоа-слојева, број неурона у слоју), *функционалност неурона*, *обучавање/валидност* (величина обучавајућег узорка, формат података, итд.), *имплементација/реализација* (аналогна, дискретна, софтверска, итд.). Неурони или *процесирајући елементи* примају улазне сигнале/информације од окружења или од других неурона преко синапси или веза које могу бити *екситационе* (побуђивачке) или *инхибиторне*. *Праг активације неурона* θ_j је његов „окидач".



Приказ типичног процесирајућег елемента - неурона

Функционалност неурона обезбеђује мрежи способност репрезентовања знања и чине је три саставна елемента:

- **улазни оператор** $f(\mathrm{w},\mathrm{x})$, који обједињује улазе и тежинске односе међусобних веза w и формира јединствену вредност s (односно $s=f(\mathrm{w},\mathrm{x})=\mathbf{w}^{\mathrm{T}}\cdot\mathrm{x}$), која је на тај начин припремљена за функцију преноса;
- **функција преноса** $h(\mathrm{s})$, која обрађује излаз из неуронског улазног оператора (врши интеграцију), формирајући потребну вредност за активациону функцију;
- **активациона функција** $f_i(net_i)$, која обрађујући излаз функције преноса управља излазном вредношћу неурона.

Улазни оператори, који се најчешће користе код неуронских модела, су дати у табели у наставку.

Математичка формулација улазног оператора неурона

Тип улазног оператора	*Математички израз*	*Напомена*
Линеаран	$net_j = \sum w_{ij} x_j$	*Линеарна функција*
Тежинска сума	$net_j = \sum w_{ij} x_j + \theta_j$	*Најчешће коришћен; у неким случајевима праг* $\theta_j = 0$
Повратна веза	$net_j = \alpha\, net_{стара} + \beta \sum w_{ij} x_j$	α, β *су тежинске константе*

После детерминисања стимулације улаза, остварује се конвертовање улазне вредности у активациону вредност или једноставно активацију неурона, што се може приказати преко опште једначине:

$$a_i = F_i\left(a_i(t-1), net_i(t)\right)$$

Ова једнакост указује на то да је активација неурона увек експлицитна функција улаза мреже ка неурону у времену t, односно симболички написано ($net_i(t)$), при чему треба водити рачуна о чињеници да она може такође да зависи и од активације неурона у претходном временском тренутку, што је дато кроз израз $a_i(t-1)$. У многим случајевима активација неурона је еквивалентна текућем улазу, односно:

$$a_i(t) = net_i(t)$$

Када је активација неурона у датом тренутку позната, неурон даје **излазни сигнал** који је у релацији са његовом активацијом преко **активационе функције**, што се математички може изразити на следећи начин:

$$o_i = x_i(t) = f_i(a_i(t)) = f_i(net_i(t))$$

Активационе функције играју веома важну улогу у различитим моделима вештачких неуронских мрежа, тако да је потребно описати детаље који су везани за најпопуларније међу њима. Активационе функције се могу сврстати у неколико заједничких група као што су: *линеарне*, *бинарне*, *сигмоидне*, *компетитивне* и *Гаусове*.

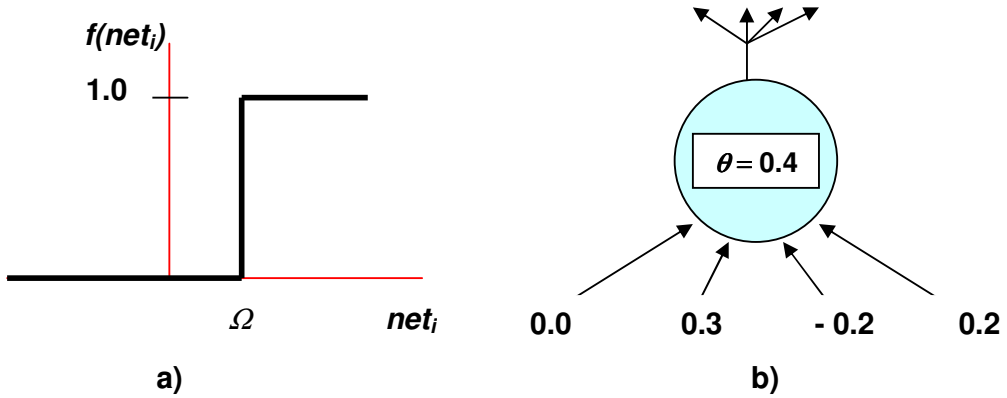
Линеарна активациона функција је најједноставнија и практично сумира све улазе у неурон пропуштајући их на излазу из неурона интегрисане у кохерентном облику, што се математички може изразити на следећи начин:

$$f_i^{\,l}(net_i(t)) = net_i(t)$$

Бинарна активациона функција, као што сам назив каже, обезбеђује неурону два стабилна стања - активно и неактивно, дато кроз следећу једначину:

$$f_i^{\,b}(net_i(t)) = \begin{cases} 1 & ako\ je\ net_i(t) > \boldsymbol{\Omega} \\ 0 & ostalo \end{cases}$$

где Ω представља тачку транзиције између два стања, што се види на слици доле под а). Код многих парадигми вештачких неуронских мрежа Ω је једнако нули, тако да је тада неурон у функцији детектора поларитета. Ову функцију можемо сматрати одскочном. Ако је улазна активација у неурон позитивна и већа од нуле, неурон „окида" и генерише активни излаз, док је у свим другим случајевима излаз инхибиторан, што значи да стимулација на улазу неурона није била довољна да пређе праг θ. За пример који је приказан на слици доле под б), уочава се да ће неурон имати активан излаз само када су улазни сигнали два и четири симултано активни, а сигнал три неактиван.



Бинарна активациона функција: а)График функције, б)Могућа стања улаза.

Сигмоидна или **S-активациона функција** функционише слично бинарној, тако што генерише излазни сигнал који има такође два стабилна стања. Разлика између ове две активационе функције је у томе што је сигмоидна математички континуална и диференцијабилна. Такође се може рећи да је она нелинеарна и неопадајућа. Има много функција које могу да се користе за моделирање, али најчешће су у употреби следеће две, дате следећим једначинама:

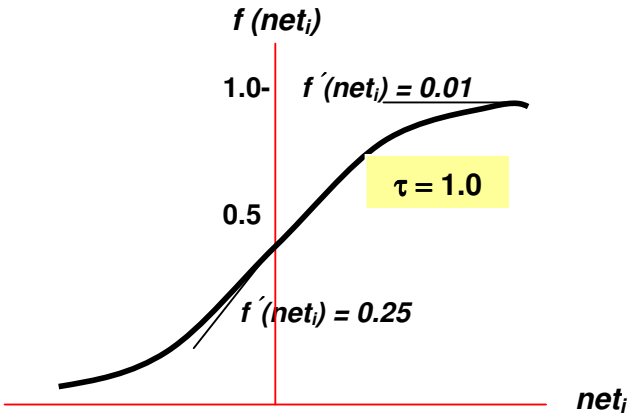
$$f_i^{\,s}(net_i(t)) = \frac{1}{1 + e^{-(net_i(t) - \boldsymbol{\Theta})/\boldsymbol{\tau}}}$$

Горња једначина представља најопштију формулацију сигмоидне функције, јер се може одредити и тачка транзиције и облик криве, варирањем вредности θ и τ, а у следећој једначини на облик криве утиче независни параметар λ и користи се код специјалних неуронских мрежа:

$$f_i^{\,s*}(net_i(t)) = \frac{1}{2}(1 + \tanh(\boldsymbol{\lambda}\, net_i(t)))$$

Сличност између сигмоидне и бинарне активационе функције се уочава почевши од графика (слика доле), и то:

- Обе активационе функције имају два домена, тако да значајна варијација улаза не утиче драстично на промену излаза који неурон остварује.
- Ширина региона транзиције код обе функције је занемарљива, ако се пореди ширина стабилних региона.
- Код обе функције регион транзиције тежи да егзистира око нулте активације, иако он може бити померен лево или десно, модификацијом величине θ, дате у једначини.



Сигмоидна активациона функција

Активни излаз из неурона који користи сигмоидну функцију, заправо представља индикацију да је присутна специфична комбинација карактеристика улазног вектора, док неактивни излаз из неурона имплицира одсуство тих карактеристика. Међутим, не би требало заборавити да континуалност сигмоидне функције понекад условљава ситуацију која може да створи конфузију о томе шта неурон стварно класификује. Да би се то илустровало, неопходно је констатовати да у примеру датом на слици горе под б), поред два могућа излазна стања неурона: активно и неактивно, коришћењем сигмоидне активационе функције понекад се на излазу појављује треће стање, а то је недефинисано стање између активног и неактивног.

Такође треба нагласити и то да сигмоидна функција условљава шта ће неурон на излазу генерисати. Могуће вредности се налазе између минималне (0.0) и максималне границе (1.0), тако да се, због претходно наговештене недефинисане ситуације, узима да су све вредности преко 0.8 активни излази, а све вредности испод 0.2 су неактивни излази. Излазне вредности између 0.2 и 0.8 ће бити разматране у транзицији, а могућа је и промена ефекта региона транзиције на понашање сигмоидног неурона преко промене вредности τ (слика горе). Што се тиче неактивног улаза (вредност 0.0), он условљава то да таква нулта стимулација, модулирана са било којом вредношћу тежинског односа, производи нулту активацију. Тако, нулта активација сигмоидног неурона проузрокује излаз од 0.5, што значи половину између валидних излазних стања. О овој значајној активационој функцији биће речи у каснијим разматрањима која су везана за одговарајући модел вештачке неуронске мреже.

Алгоритми учења вештачких неуронских мрежа

Модели неуронских мрежа (неурона) су најчешће **динамички**, јер се развијају у функцији од времена, што се и види по претходним изразима. Општа диференцијална једначина,

$$\dot{x} = g_i\left(x_i, net_i\right)$$

представља излаз i-тог неурона, при чему је очигледно да се, због **динамичких неуронских модела**, ради о диференцирању по времену. Како улаз i-тог неурона net_i, зависи од излаза многих неурона са којима је у вези, практично се развија систем спрегнутих нелинеарних диференцијалних једначина.

Потребно је и тежинске односе између неурона посматрати као **динамички систем**. То значи да имамо систем диференцијалних једначина за тежинске односе,

$$\dot{w}_{ij} = A_i\left(w_{ij}, x_i, x_j, ...\right)$$

где A_i представља **алгоритам учења**.

Процес учења, базиран на алгоритмима учења, одређује тежинске односе кроз итеративни поступак њихове модификације. Систем вештачке неуронске мреже на тај начин учи, тако да је неопходно издвојити најчешће коришћене алгоритме учења (табела доле). **Избор** одговарајућег **алгоритма учења**, сходно апликацији за коју се мрежа користи, представља значајан проблем при пројектовању мреже.

Преглед најзначајнијих алгоритама учења

Алгоритам учења	*Математичка формулација*	*Напомене*
***Hebb*-ово правило** (1949)	$\partial\Phi/\partial t = O(t)[I_i(t) - O(t)\Phi_i(t)]$	$O(t)$ – *излаз неурона* $I_i(t)$ – *улази у неурон* $\Phi_i(t)$ – *јачина синаптичких веза*
***Widrow-Hoff*-ово правило** (1960) (LMS-„least mean square" алгоритам, тј. алгоритам најмање квадратне разлике)	$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + 2\mu\,\varepsilon_k\,\boldsymbol{x}_k$	$\boldsymbol{x_k}$ – *k-ти улазни вектор* $\boldsymbol{w}$ – *тежински вектор* μ – *позитивна константа* ε_k – *тренутна разлика (грешка) између захтеване вредности излаза* (d_k) *и актуелне вредности* (y_k)
Генералисано делта правило (1986) (простирање грешке уназад)	$\boldsymbol{w_{ji}}(t+1) = \boldsymbol{w}(t)_{ji} + \eta\,\delta_{pj}\,\boldsymbol{x}_i$	$\boldsymbol{x_i}$ – *i-ти улазни вектор* $\boldsymbol{w_{ji}}$ – *вектор тежинских односа* η – *параметар учења* δ_{pj} – *грешка везана за излаз неурона* $\delta_{pj} = (y_{pj} - o_{pj}) - y_{pj}$ *је захтевана вредност излаза, а* o_{pj} *је актуелна вредност* p-*тог обучавајућег вектора за* j-*ти неурон*

Питања на која треба одговорити могу бити:

- Које перформансе треба да има систем неуронске мреже у погледу обучавања?
- Да ли је потребно развити чисто динамички систем неуронске мреже?
- Хоће ли фаза учења бити реализована у „on-line" или „off-line" режиму?
- Да ли је брзина конвергенције важна за учење неуронске мреже?

Одговори на ова питања су директно везани за алгоритме учења који се данас користе, што ће се у овом курсу увидети. Наиме, постоје варијације и усавршавања основних алгоритама учења, у погледу конвергентне способности неуронске мреже, па се у оквиру овог курса посебно разматра конвергенција приликом примене *генералисаног делта правила* за мрежу са простирањем грешке уназад („back-propagation").

Поред ових најзначајнијих алгоритама учења, који се користе за супервизорско и несупервизорско учење, постоје и неуронске мреже са обучавањем без надгледања. Оне уче кроз процес довођења обучавајућег вектора на њихове улазе, а да при томе вредности излазног вектора нису познате. У мрежу је уграђена способност класификације улаза у различите категорије, на бази самоорганизовања и коришћењем интерних правила. Једна од парадигми учења без надгледања је везана за мрежу са **компетитивним обучавањем** (ART-1). Основна снага ове врсте мреже је у оспособљавању мреже да врши једноставну идентификацију улазних варијанти, што је искоришћено и примењено у спроведеним истраживањима код нас (З.Миљковић-књига) кроз ефикасну процедуру идентификације објеката снимљених камером.

Модели вештачких неуронских мрежа

Примена система вештачких неуронских мрежа је веома широка. Током деведесетих година 20. века почеле су да се користе и у различитим областима производног инжењерства, при чему су се издвојиле три основне категорије употребе: *класификација*, *предикција* и *функционална апроксимација*.

Функционална апроксимација је представљена скупом независних променљивих $\mathbf{x}=[x_1,x_2,...,x_m]^T$ и излазним сигналом $y=f(x_1,x_2,...,x_m)$, који су груписани у парове $\{y,f(\mathbf{x})\}$.

Предикција је представљена претходним вредностима стања система, $\mathbf{X}=[x(k), x(k-1),...,x(k-n)]^T$, као и тренутним и претходним вредностима улаза, $\mathbf{U}=[u(k), u(k-1),...,u(k-m)]^T$, које са будућим стањима система, $x(k+1)=f(\mathbf{X},\mathbf{U})$ формирају парове $\{x(k+1),(\mathbf{X},\mathbf{U})\}$.

Класификација је представљена скупом улаза $\{{}^{c}x_i,\ _{i=1,...,Nj}\}$, који са специфицираном класом C_j формира парове $\{C_j,{}^{c}\mathbf{x}_j\}$.

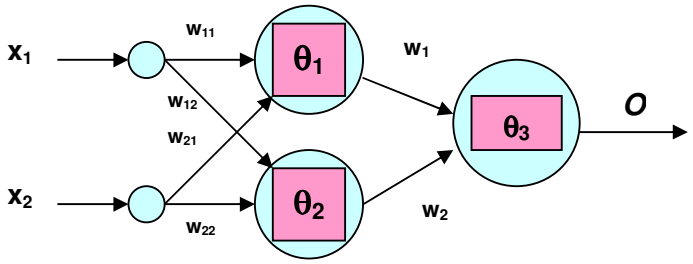
Најраспрострањенији *модели вештачких неуронских мрежа* су:

- Перцептрон;
- „**Backpropagation**" **(BP) неуронска мрежа**;
- Асоцијативне неуронске мреже;
- *Hopfield*-ове неуронске мреже;
- ART неуронске мреже (ART-1, ART-2, ART-3);
- *Fuzzy* асоцијативне неуронске мреже;
- Самоорганизујуће неуронске мреже.

Од свих набројаних модела вештачких неуронских мрежа, у најзаступљеније се убраја пре свих „backpropagation" (BP) мрежа, тако да ће се дати објашњења за ову неуронску мрежу (пројектни задатак), а како перцептрон представља претечу BP неуронске мреже објашњења ће почети од овог модела.

Перцептрон

Перцептрон представља најранији модел вештачке неуронске мреже, а први пионирски радови датирају од *Warren McCulloch*-а и *Walter Pitts*-а из 1943. године, па преко *Rosenblatt*-а (1958 и 1962) до *Minsky*-ог и *Papert*-а (1969). Архитектура перцептрона је једноставна, користи простирање сигнала у једном смеру („feedforward"), има један или више слојева неурона између улазних и излазних неурона (најчешће цео перцептрон има два слоја) и функционише на бази супервизорског учења. Шематски приказ перцептрона је дат на слици у наставку.



Елементарна структура перцептрона

Код ове неуронске мреже, која користи линеарни тип неурона са прагом θ, излазна вредност је дата следећим изразом:

$$o = f(net) = \begin{cases} 1 & net \geq \Theta \\ 0 & net < \Theta \end{cases}$$

при чему је $net = \sum w_{ij}\, x_j$.

Најрепрезентативније објашњење архитектуре перцептрона, као система вештачке неуронске мреже, дао је *Rosenblatt*. Структуру перцептрона чини скуп улазних неурона ({s}–неурони), скуп скривених неурона ({a}–неурони) и један излазни неурон ({r}–неурон). *Rosenblatt*-ова основна идеја је била везана за то, да би јачину веза између улазног слоја неурона и скривеног слоја неурона требало случајно изабрати по неком закону вероватноће и фиксирати њихове вредности током целог процеса учења. Алгоритам учења би затим могао да реализује подешавање тежинских односа између скривеног слоја неурона и јединог излазног неурона. Наиме, он је сматрао да ако има довољно различитих типова неурона у скривеном слоју који репрезентују различите логичке исказе из улазног слоја неурона, онда је процес учења од неурона из скривеног слоја до излазног неурона у стању да реализује комплексно логичко закључивање.

Алгоритам за учење перцептрона се може представити кроз шест корака. Констатовано је да процес учења увек конвергира у коначном броју итерација приликом коректне класификације свих чланова обучавајућег скупа и да перцептронски алгоритам учења тражи минималну вредност активационе функције.

Алгоритам учења перцептрона

Корак 1: Иницијализација. Нека је *t=0* и нека вектор иницијалних тежинских односа $\mathbf{w}(t)\in \Re^{a}$ дефинише иницијално пресликавање од скривених неурона до излазног неурона.

Корак 2: Изабрати стимулацију (подстреке) перцептрона примењујући узорке из обучавајућег скупа x^k на случајан начин, што се може исказати помоћу $x^k(t)=(s^k(t),\mathbf{o}^k(t))\in \{0,1\}$, где је $\mathbf{o}^k$ циљни вектор, који представља захтевани одговор на стимулацију.

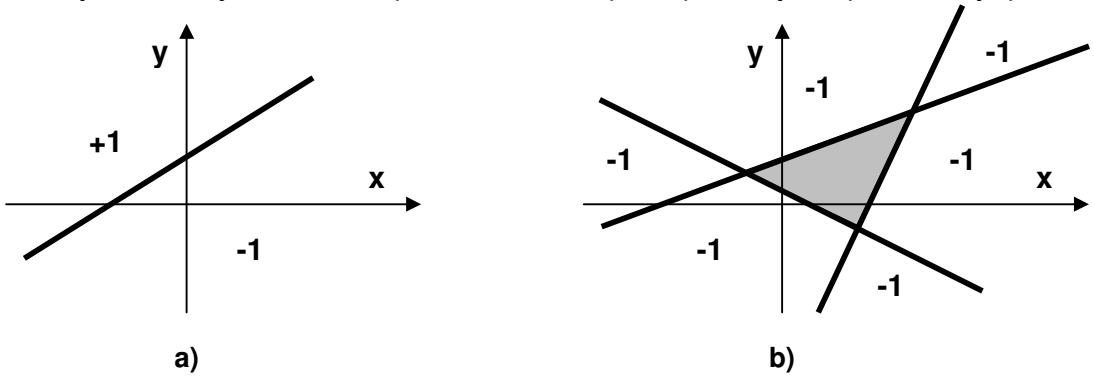
Корак 3: Израчунати активацију неурона у скривеном слоју h^k, на основу активације улазних неурона, $h^k(t)= q(s^k(t))$.

Корак 4: Израчунати активацију излазног неурона мреже r^k, на основу активације неурона из скривеног слоја, $r^k(t)= f(net)= f(\mathbf{w}(t)^T, h^k(t))$, при чему је *f(net)*=1 ако је $net \geq 0$ и *f(net)*=0 ако је $net < 0$.

Корак 5: Модификовати тежинске односе између скривених неурона и излазног неурона, $\mathbf{w}(t+1)= \mathbf{w}(t)+[\mathbf{o}^k(t)- r^k(t)]h^k(t)$.

Корак 6: Зауставити учење ако је $t > t_{max}$ или $\mathbf{o}^k(t)= r^k(t)$, за k=1,...,n , иначе нека је t=t+1 и тада је неопходно вратити се на корак 1.

Marvin Minsky и *Seymour Papert* су 1969. године дали детаљну анализу перцептрона у погледу могућности и ограничења примене. Они су констатовали да је његова главна карактеристика везана за линеарну дискриминациону функцију у простору узорака и то у коначном броју итерација. То значи да се коректна класификација може спровести само за линеарно раздвајајуће класе узорака. На слици доле су приказани региони које перцептрон може лако да идентификује. За проблем приказан на слици под а), само један процесирајући елемент перцептрона је потребан, с обзиром да је цела површина раздељена једном правом линијом на два одвојена региона, који су идентификовани са **+1** и **-1**. Међутим, регион који је осенчен (**+1**) на слици под б), формиран је од много правих линија, тако да је неопходно развити више перцептрона који ће решити тај проблем.



Идентификација региона

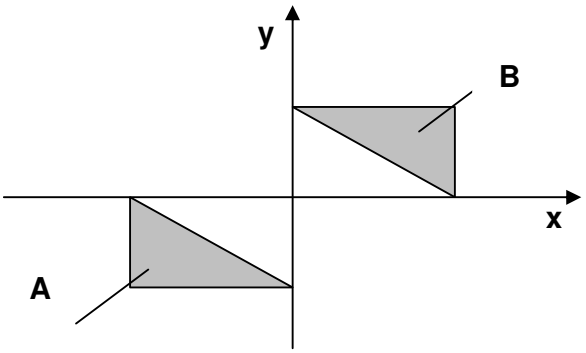
Да би се илустровало понашање перцептрона при реализацији логичког ***ILI***, полази се од табеле која садржи истините исказе за регионе приказане на слици у наставку. За реализацију резултата датих у табели у наставку довољна је једноставна структура трослојног перцептрона који је приказан на слици (почетак објашњења перцептрона), а који има следеће вредности тежинских односа w_{ij} и прага неурона θ_1, θ_2 и θ_3:

$w_{11}=1,\ w_{12}=1,\ w_{21}=1,\ w_{22}=1,$

$w_1=1,\ w_2=-1,$

$\theta_1=0.5,\ \theta_2=1.5,\ \theta_3=0.5.$

Према датом изразу (види горе), лако се може израчунати излаз из овог трослојног перцептрона за различите улазе x_1 и x_2 , нпр. ако је $x_1 = x_2 = 0$ (табела), а према датим вредностима за w и θ , очигледно да је излаз $o=f(net)=0$ јер је $net<\theta$. На сличан начин, помоћу трослојног перцептрона са три неурона у средњем слоју (модификована структура у односу на перцептрон приказан на слици горе-почетак објашњења перцептрона), могуће је извршити идентификацију региона **A** и **B** са слике у наставку:



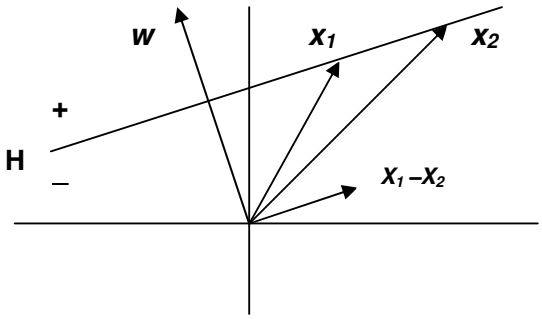
Пример за перцептрон (логичко ***ILI***)

Тачка 1(x_1)	Тачка 2(x_2)	Излаз
У рег. А (0)	**У рег. В** (0)	0
У рег. А (0)	**Ван рег. В** (1)	1
Ван рег. А (1)	**У рег. В** (0)	1
Ван рег. А (1)	**Ван рег. В** (1)	0

Вишеслојни перцептрони проширују могућности класификације на шире скупове могућих међуодноса класа, у зависности од броја слојева. Зато се уводе појмови *хиперраван* и *хиперпростор*, тако да су *хиперравни*, за *n*-димензионални простор (*хиперпростор*), одређене *n*-1 димензијом. У *n*-димензионалном улазном простору *X*, хиперраван *H* која врши сепарацију, дефинисана је на следећи начин:

$$w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \boldsymbol{\theta} = -w_0$$

при чему је вектор тежинских односа $\mathbf{w} = [w_1, w_2, \cdots, w_n]^T$ управан на ту хиперраван (слика доле). Да би се то појаснило треба посматрати једноставан пример са два *n*-димензионална улазна вектора, $\mathbf{x_1}$ и $\mathbf{x_2}$, који су лоцирани на хиперравни. Слика приказује дводимензионалан случај, где се уочава да су вектор **w** и хиперраван *H* међусобно управни. Хиперраван *H* дели *n*-димензионални улазни простор *X* на два полу-простора, регион X^+ где је $\sum w_i x_i > -w_0$ (излаз је 1), односно регион X^- где је $\sum w_i x_i \leq -w_0$ (излаз је 0). То значи да је у тродимензионалном простору раван дефинисана двема димензијама, тако да она може да изврши раздвајање таквог простора на два одвојена региона, а са две равни на три или четири региона, итд. На тај начин се може, егзактном анализом, доказати да је вишеслојни перцептрон са једним скривеним слојем, у стању да са произвољном тачношћу униформно апроксимира било коју реалну континуалну функцију на коначној реалној оси, док са два скривена слоја може да униформно апроксимира било коју реалну континуалну функцију више аргумената.



Пример раздвајања дводимензионалног простора

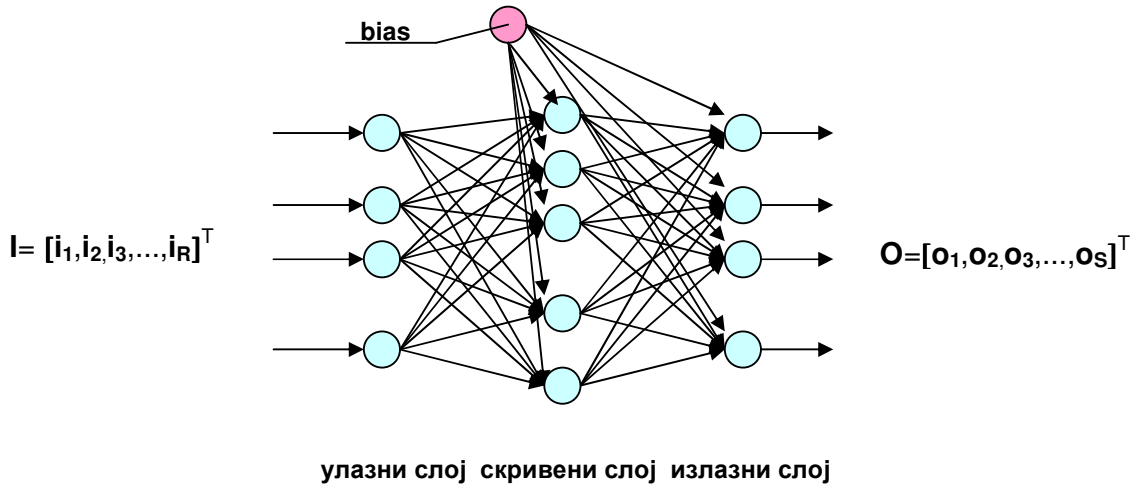
Перцептрон, као линеарни сепаратор, има велика ограничења у погледу нелинеарних пресликавања. То значи да је проблем његове примене везан за нелинеарно раздвајање класа, јер он користи бинарну активациону функцију, која условљава да неуронска мрежа има дисконтинуитете.

„Backpropagation" (BP) неуронска мрежа

„Backpropagation" (BP) неуронска мрежа је развијена са циљем да се проблем нелинеарног пресликавања из улазног простора у излазни простор успешно реши, при чему се остварује модификација тежинских односа и између улазног и скривеног слоја неурона. BP је, као и перцептрон, неуронска мрежа са простирањем сигнала унапред, која такође реализује супервизорски вид учења, са различитим активационим функцијама и алгоритмима учења. BP мрежа користи *градијентни поступак* при обучавању, који је аналоган процесу минимизације грешке. То значи да учи пресликавања из улазног простора узорка у излазни простор, кроз *процес минимизације грешке* између актуелног излаза који је остварила мрежа и захтеваног излаза, на основу скупа обучавајућих парова, односно примера. Процес учења почиње са презентацијом улазног облика узорка BP мрежи, који простирањем кроз мрежу остварује излазни облик. BP мрежа затим примењује *генерализовано делта правило* да би се утврдила грешка на излазу, коју простирањем уназад преко скривеног слоја, користи за „лагано" модификовање сваког тежинског односа између неурона, што се понавља за сваки нови узорак. Генерализовано делта правило обезбеђује конвергенцију процеса учења до задатог нивоа тачности кроз итеративни процес адаптације тежинских односа.

Систем BP неуронске мреже користи сигмоидну активациону функцију. Као што је речено раније, ова активациона функција је нелинеарна, непрекидна и диференцијабилна, што обезбеђује да се процес учења BP мреже успешно реализује. Применом *генерализованог делта правила* утврђена грешка се коригује, тако што се користи први извод сигмоидне функције $f'(net) = f'(S)$. Уколико сигмоидна функција $f(net)$ брже расте, онда треба извршити већу корекцију грешке и обрнуто.

Елементарна архитектура BP мреже има три слоја (слика доле), који су потпуно повезани, с тим што се, не тако ретко, користи и структура са више од једног скривеног слоја. Број неурона у сваком слоју се разликује, у зависности од области примене, као и број скривених слојева. Најчешће постоји само један скривени слој, јер је BP мрежа са таквом структуром у стању да обезбеди репродуковање скупа захтеваних излазних облика за све обучавајуће парове. На слици се види да мрежа може да поседује и екстра улаз - неурон који је увек активан и има излазно стање 1 (константна активација), познат као „*bias*" неурон. Он је повезан са свим неуронима у скривеном и излазном слоју. Понаша се у мрежи као и сваки други неурон, при чему има задатак да партиципира у процесу учења тако што тежински однос између „*bias*" неурона и сваког неурона у следећем слоју формира активацију која мора бити савладана остатком улаза у сваки неурон, услед чега је и њихова активација контролисана. „*Bias*" неурон обезбеђује константан члан у тежинским сумама неурона наредног слоја, што резултира побољшањем конвергентне карактеристике BP мреже.



Елементарна архитектура BP неуронске мреже

Дакле, BP мрежа користи ***генерализовано делта правило*** приликом модификације тежинских односа између неурона, чиме се остварују основне карактеристике мреже а то је **генерализација** и **нелинеарно раздвајање**. Да би се схватиле предности ових карактеристика, у наставку се описује генерализовано делта правило и поступак обучавања BP мреже.

Нека је *M* број обучавајућих парова који се презентирају мрежи, *R* број неурона у улазном слоју и *C* број неурона у излазном слоју. Обучавајући парови ($\mathbf{x_1},\mathbf{y_1}$)...($\mathbf{x_M},\mathbf{y_M}$), који су доведени на улаз мреже, обезбеђују мрежи могућност да реализује нелинеарно пресликавање између $\mathbf{x_i}$ и $\mathbf{y_i}$ вектора, за i=1,...,M.

Нека је *L* број слојева мреже и нека је са *l* означен сваки слој:

$l = 0$ улазни слој,

$l = 1,...,L-1$ скривени слој(еви),

$l = L$ излазни слој.

Неуронска мрежа има K_l неурона у сваком слоју, где је $l = 1,...,L$. Улази у k-ти неурон l-тог слоја су дати преко I_{ij}^{l}, за j=1,...,K_{l-1}. Ови улази су истовремено и излази неурона претходног слоја, који се могу означити са O_{ij}^{l-1}. Заједно са прагом $\boldsymbol{\theta}_k^l$, за k-ти неурон у слоју l, сумирана улазна вредност је дата следећим изразом:

$$S_{ik}^{l} = net_{ik}^{l} = \sum_{j=1}^{K_{l-1}} w_{kj}^{l} I_{ij}^{l} + \boldsymbol{\theta}_k^l$$

где су w_{kj}^{l} тежински односи између неурона j и k.

Излаз из k-тог неурона у l-*том* слоју је остварен као резултат активације неурона. Наиме, пропушта се улаз S_{ik}^{l} преко функције преноса $h_k^l(s)$, како би се коришћењем активационе функције $f_k^l(x)$ генерисао излаз, дат преко израза:

$$O_{ik}^{l} = f_{ik}^{l}\left(S_{ik}^{l}\right)$$

Грешка на излазном слоју *L*, ВР неуронске мреже, је представљена разликом између израчунатих и задатих излаза:

$$E_{pi} = \frac{1}{2}\sum_{k=1}^{K_L}\left(\, y_{ik} - O_{ik}^{l}\right)^2$$

Ова грешка се минимизира уз модификацију тежинских односа. Да би се то илустровало у наставку се даје израз за израчунавање грешке за мрежу са два скривена слоја (L=3):

$$E_{pi} = \frac{1}{2}\sum_{k=1}^{K_3}\left(\, y_{ik} - O_{ik}^{3}\right)^2 = \frac{1}{2}\sum_{k=1}^{K_3}\left(\, y_{ik} - f_k^3\left(S_{ik}^{3}\right)\right)^2 = ... =$$

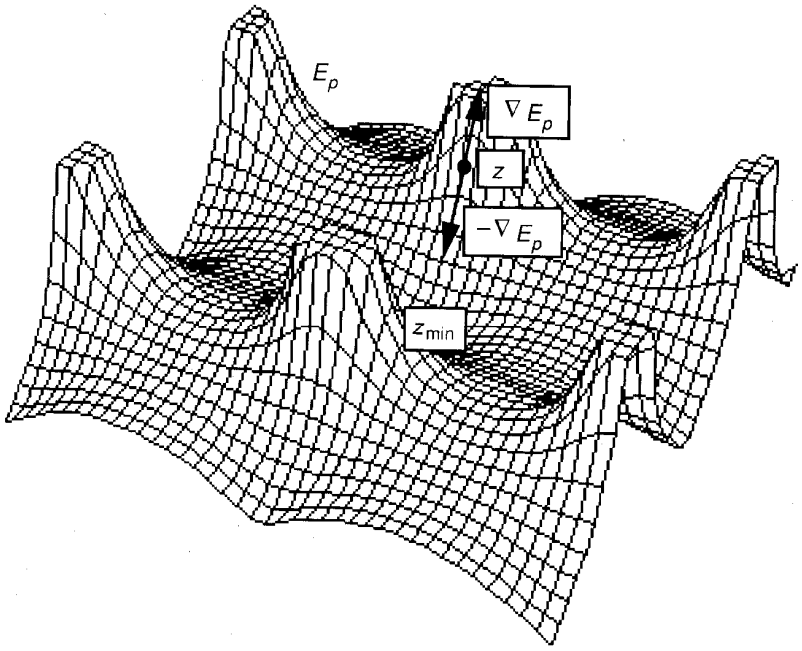
$$= \frac{1}{2}\sum_{k=1}^{K_3}\left(\, y_{ik} - f_k^3\left(\sum_{j=1}^{K_2} w_{kj}^3 f_j^2\left(\sum_{u=1}^{K} w_{ju}^2 f_u^1\left(\sum_{v=1}^{K_0} w_{jv}^1 O_{iv}^0 + \boldsymbol{\theta}_u^1\right) + \boldsymbol{\theta}_j^2\right) + \boldsymbol{\theta}_k^3\right)\right)^2$$

где је $O_{iv}^{0} = I_{iv}^{1} = x_{iv}$.

На основу једначине за E_p може се израчунати градијент грешке E_p, с обзиром на тежинске односе w_{kj}^{l}, тако да је за излазни слој (l=3):

$$\frac{\partial E_{pi}}{\partial w_{kj}^{3}} = -\left(y_{ik} - O_{ik}^{3}\right)\frac{\partial O_{ik}^{3}}{\partial S_{ik}^{3}}\frac{\partial S_{Iik}^{3}}{\partial w_{kj}^{3}} = -\left(y_{ik} - O_{ik}^{3}\right) f_k^{3\prime}\left(S_{ik}^{3}\right) I_{ij}^{3}$$

Негативни предзнак указује на то да је правац промене тежинских односа са негативним градијентом ∇E_p, што је приказано на слици доле за тачку z хипотетичке комплексне површи грешке E_p у простору тежинских односа. Тежински односи се итеративно мењају док грешка E_p не оствари минумум у тачки $z_{\min}$.



Графички приказ правца простирања градијента грешке E_p

Инкремент промене тежинских односа, за задати параметар учења η, је:

$$\Delta_i w_{kj}^3 = \eta \left(y_{ik} - O_{ik}^3\right) f_k^{3\prime}\left(S_{ik}^3\right) I_{ij}^3 = \eta\, \delta_{ik}^3\, I_{ij}^3, \qquad \text{где је} \quad \delta_{ik}^3 = \left(y_{ik} - O_{ik}^3\right) f_k^{3\prime}\left(S_{ik}^3\right).$$

Параметар учења η се најчешће бира у опсегу од 0.25 до 0.75. Овај параметар одређује брзину обучавања, тако што је за веће вредности обучавање брже и обрнуто. Сувише велике вредности овог параметра утичу на стабилност мреже и узрокују незадовољавајући процес обучавања, док с друге стране, ако је вредност много мала, обучавање је веома споро. Понекад се користи и метода променљивог параметра, код које је у почетку обучавања параметар η већи, док се током обучавања он смањује, омогућавајући мрежи, на тај начин, боље карактеристике учења.

Слично претходном, за други скривени слој (l=2):

$$\frac{\partial E_{pi}}{\partial w_{ju}^2} = -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^3\right)\frac{\partial O_{ik}^3}{\partial S_{ik}^3}\frac{\partial S_{Ik}^3}{\partial O_{ij}^2}\frac{\partial O_{ij}^2}{\partial S_{ij}^2}\frac{\partial S_{Ij}^2}{\partial w_{ju}^2} = -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^3\right) f_k^{3\prime}\left(S_{ik}^3\right) w_{kj}^3 f_j^{2\prime}\left(S_{ij}^2\right) I_{iu}^2$$

$$\Delta_i w_{ju}^2 = \eta\, \delta_{ij}^2\, I_{ij}^2, \quad \text{где је} \quad \delta_{ij}^2 = f_j^{2\prime}\left(S_{ij}^2\right)\sum_{k=1}^{K_3}\delta_{ik}^3 w_{kj}^3$$

За први скривени слој (l=1):

$$\frac{\partial E_{pi}}{\partial w_{uv}} = -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^3\right)\frac{\partial O_{ik}^3}{\partial S_{ik}^3}\frac{\partial S_{Ik}^3}{\partial O_{ij}^2}\frac{\partial O_{ij}^2}{\partial S_{ij}^2}\frac{\partial S_{ij}^2}{\partial O_{iu}}\frac{\partial O_{iu}}{\partial S_{iu}}\frac{\partial S_{iu}}{\partial w_{uv}}$$

$$= -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^3\right) f_k^{3\prime}\left(S_{ik}^3\right) w_{kj}^3 \sum_{j=1}^{K_2} f_j^{2\prime}\left(S_{ij}^2\right) w_{ju}^2\, f_u^{1\prime}\left(S_{iu}^1\right) x_{iv}$$

$$\Delta_i w_{uv}^1 = \eta\, \delta_{iu}^1\, x_{iv}, \qquad \text{где је} \quad \delta_{iu}^1 = f_u^{1\prime}\left(S_{iu}^1\right)\sum_{j=1}^{K_2}\delta_{ij}^2 w_{ju}^2$$

Вредности $\Delta_i w_{kj}^3, \Delta_i w_{ju}^2, \Delta_i w_{uv}^1$ су коришћене за модификацију тежинских односа током *процеса обучавања*, односно *тренирања* ВР неуронске мреже, а обучавање се завршава када су ове вредности блиске нули. Алгоритам учења ВР мреже се, на основу описаног генералисаног делта правила, може представити преко седам корака.

Алгоритам учења ВР неуронске мреже

Корак 1: Иницијализација и филтрирање. Довести улазни вектор $\mathbf{I}_{ij} = (x_{i1}, x_{i2}, ..., x_{iR})$ до неурона у улазном слоју.

Корак 2: Израчунати излазне вредности за сваки неурон у мрежи коришћењем израза:

$$O_{ik}^l = f_{ik}^l\left(S_{ik}^l\right), \quad \text{где је} \quad S_{ik}^l = net_{ik}^l = \sum_{j=1}^{K_l} w_{kj}^l I_{ij}^l + \theta_k^l \ .$$

Корак 3: Израчунати грешку коју је мрежа генерисала на излазу $E_{pi} = \frac{1}{2}\sum_{k=1}^{K_L}\left(y_{ik} - O_{ik}^L\right)^2$.

Корак 4: За излазни слој, $l = L$, израчунати вредности промене тежинских односа помоћу израза: $\Delta_i w_{kj}^L = \eta\left(y_{ik} - O_{ik}^L\right) f_k^{L'}\left(S_{ik}^L\right) I_{ij}^L = \eta\, \delta_{ik}^L I_{ij}^L$

Корак 5: За скривене слојеве, $l = 1,...,L-1$, израчунати вредности промене тежинских односа

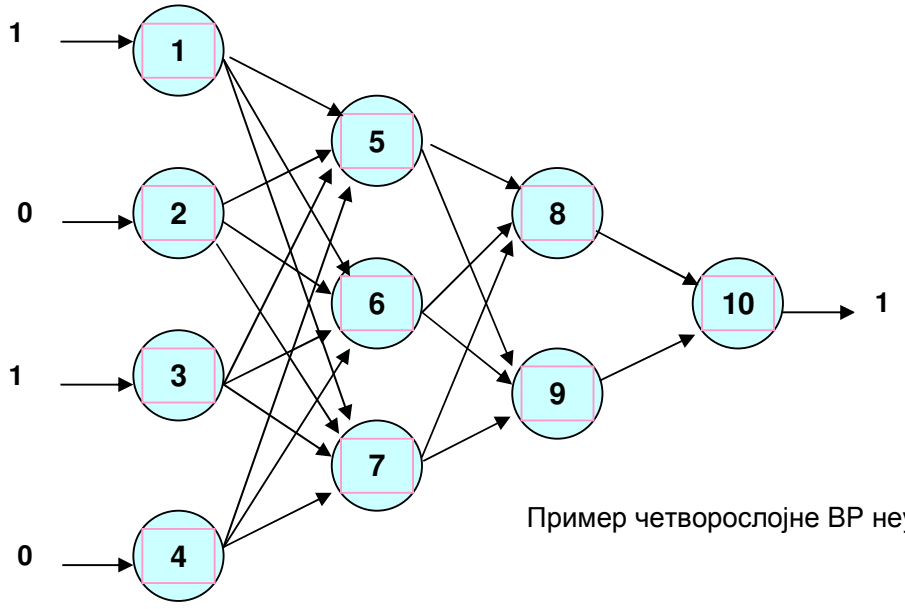
$$\Delta_i w_{ju}^l = \eta\, \delta_{ij}^l I_{ij}^l, \quad \text{где је} \quad \delta_{ij}^l = f_j^{l'}\left(S_{ij}^l\right)\sum_{k=1}^{K_{l+1}} \delta_{ik}^{l+1} w_{kj}^{l+1}$$

Корак 6: Ажурирати (модификовати) све вредности инкременталне промене тежинских односа $w_{ju}^l = w_{ju}^l + \Delta_i w_{ju}^l$, за $l = 1,...,L$.

Корак 7: Поновити све претходне кораке док грешка коју генерише мрежа не буде блиска нули или довољно мала.

Напомена: Треба нагласити да је грешка, везана за неуроне у скривеним слојевима, израчуната пре него што су ажурирани (модификовани) њихови тежински односи са неуронима у излазном слоју.

Да би процес обучавања био јасно презентиран, најбоље је илустровати цео поступак једноставним примером. Наиме, четворослојна ВР неуронска мрежа, приказана на слици у наставку, се тренира коришћењем једноставних обучавајућих парова. Обучавајући пар, (1010,1), је употребљен у овом примеру за демонстрацију описаног механизма ажурирања тежинских односа, уз параметар учења η=0.5. Различите сигмоидне активационе функције су коришћене за неуроне у скривеним слојевима и излазном слоју, а дате су у табели у наставку. Први изводи ових функција су такође дати у табели. Пре израчунавања, бинарни обучавајући парови морају бити промењени реалним бројевима, који ће се користити уместо 0 и 1. Тако ће 0.1 представљати 0, а 0.9 ће одговарати 1. Неурони улазног слоја прослеђују ка следећем слоју оне вредности које су примили од улазног вектора (1010).



Пример четворослојне ВР неуронске мреже

Преглед коришћених сигмоидних активационих функција

Неурон	*Сигмоидна активациона функција*	*Први извод активационе функције*
5	$f(net)=f(S)=f(x)=\left(1+e^{-3x}\right)^{-1}$	$f'(net)=f'(S)=f'(x)=\dfrac{3e^{-3x}}{\left(1+e^{-3x}\right)^2}$
6	$f(net)=\left(1+e^{-4x}\right)^{-1}$	$f'(net)=\dfrac{4e^{-4x}}{\left(1+e^{-4x}\right)^2}$
7	$f(net)=\left(1+e^{-5x}\right)^{-1}$	$f'(net)=\dfrac{5e^{-5x}}{\left(1+e^{-5x}\right)^2}$
8	$f(net)=\left(1+e^{-6x}\right)^{-1}$	$f'(net)=\dfrac{6e^{-6x}}{\left(1+e^{-6x}\right)^2}$
9	$f(net)=\left(1+e^{-7x}\right)^{-1}$	$f'(net)=\dfrac{7e^{-7x}}{\left(1+e^{-7x}\right)^2}$
10	$f(net)=\left(1+e^{-8x}\right)^{-1}$	$f'(net)=\dfrac{8e^{-8x}}{\left(1+e^{-8x}\right)^2}$

Процес обучавања се одвија сходно следећим корацима:

1. Одређују се тежински односи w^l и праг неурона $\boldsymbol{\theta}^l$, коришћењем вредности између – 0.5 и 0.5.

$$w^1=\begin{bmatrix}0.182 & -0.100 & -0.088\\ 0.376 & -0.391 & -0.321\\ 0.206 & 0.416 & 0.055\\ -0.233 & 0.442 & -0.192\end{bmatrix} \qquad \boldsymbol{\theta}^1=\begin{bmatrix}0.282 & 0.107 & -0.155\end{bmatrix}$$

$$w^2=\begin{bmatrix}0.058 & -0.249\\ -0.281 & -0.332\\ -0.352 & 0.471\end{bmatrix} \qquad \boldsymbol{\theta}^2=\begin{bmatrix}-0.187 & 0.478\end{bmatrix}$$

$$w^3=\begin{bmatrix}0.231\\ -0.399\end{bmatrix} \qquad \boldsymbol{\theta}^3=\begin{bmatrix}0.399\end{bmatrix}$$

2. Уводи се улазни вектор у мрежу.

$$I=\begin{bmatrix}0.9 & 0.1 & 0.9 & 0.1\end{bmatrix}^T$$

3. Израчунава се, према претходно датим изразима, излаз неурона у првом скривеном слоју (l=1),

$$O^1=\begin{bmatrix}0.874 & 0.831 & 0.234\end{bmatrix}$$

4. Израчунава се, према претходно датим изразима, излаз неурона у другом скривеном слоју (l=2),

$$O^2=\begin{bmatrix}0.062 & 0.661\end{bmatrix}$$

5. Израчунава се, према претходно датим изразима, излаз из излазног слоја мреже (l=3),

$$O^3=\begin{bmatrix}0.768\end{bmatrix}$$

6. Израчунава се δ^3, према претходно датим изразима, за грешку која се односи на разлику између захтеваног излаза и излаза који је мрежа генерисала, па с обзиром на први извод сигмоидне активационе функције, дат у табели, у слоју *l*=3 биће:

$$\delta^3 = [0.188]$$

7. Израчунавају се инкременталне промене тежинских односа, Δw^3, у слоју *l*=3,

$$\Delta\, w^3 = \begin{bmatrix} 0.0058 \\ 0.0621 \end{bmatrix}$$

8. Израчунава се δ^2, коришћењем w^3 и првих извода сигмоидних активационих функција у слоју *l*=2,

$$\boldsymbol{\delta}^2 = [0.0152 \quad -0.1179]$$

9. Израчунава се Δw^2, промене тежинских односа, у слоју *l*=2,

$$\boldsymbol{\Delta} w^2 = \begin{bmatrix} 0.0066 & -0.0515 \\ 0.0063 & -0.0489 \\ 0.0018 & -0.0139 \end{bmatrix}$$

10. Израчунава се δ^1, коришћењем w^2 и првих извода сигмоидних активационих функција у слоју *l*=1,

$$\boldsymbol{\delta}^1 = [0.0100 \quad 0.0197 \quad -0.0547]$$

11. Израчунава се Δw^1, промене тежинских односа, у слоју *l*=1,

$$\boldsymbol{\Delta} w^1 = \begin{bmatrix} 0.0045 & 0.0089 & -0.0246 \\ 0.0005 & 0.0010 & -0.0027 \\ 0.0045 & 0.0089 & -0.0246 \\ 0.0005 & 0.0010 & -0.0027 \end{bmatrix}$$

12. Ажурирају (модификују) се све матрице тежинских односа.

$$w^1 = \begin{bmatrix} 0.1865 & -0.0911 & -0.1126 \\ 0.3765 & -0.3900 & -0.3237 \\ 0.2105 & 0.4249 & 0.0796 \\ -0.235 & 0.4430 & -0.1947 \end{bmatrix}$$

$$w^2 = \begin{bmatrix} 0.0646 & -0.3005 \\ -0.2747 & -0.3809 \\ -0.3502 & 0.4571 \end{bmatrix}$$

$$w^3 = \begin{bmatrix} 0.2368 \\ -0.3367 \end{bmatrix}$$

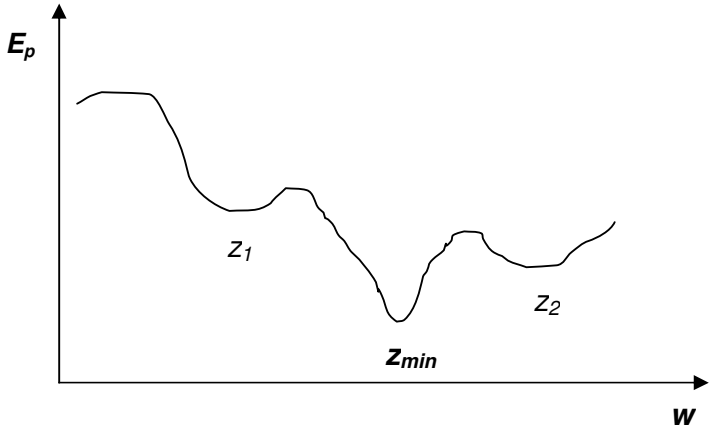
13. Доводи се нови узорак (улазни вектор) у мрежу и израчунава се нова излазна вредност.

Ова процедура се наставља све док се не дође до задовољавајућих резултата за све узорке из обучавајућег скупа. До задовољавајућих резултата ВР неуронска мрежа долази у тренутку када је грешка E_p довољно мала и када мрежа на излазу даје жељене вредности. Када се у том тренутку заврши обучавање мреже, тежински односи веза између неурона остају непромењени. Тада се мрежа може користити за оно за шта је обучавана!

Успешна примена ВР вештачке неуронске мреже зависи пре свега од њене структуре. Број неурона у улазном и излазном слоју је условљен репрезентативним улазним и излазним вектором, сходно примени за коју је структура ВР мреже пројектована. Што се броја скривених слојева тиче, тешко је проценити да ли ће мрежа са више скривених слојева „радити" боље. Такође је отворено

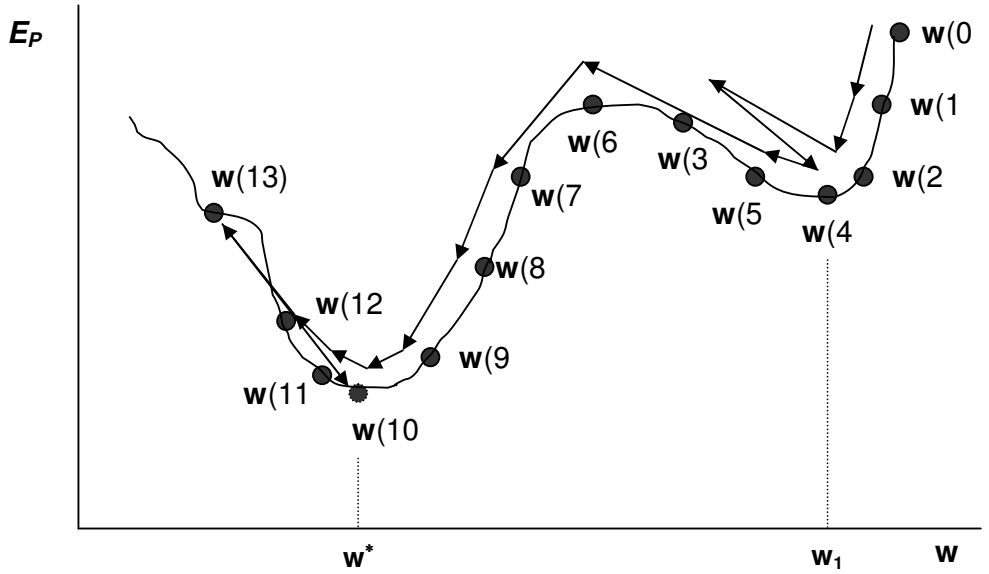
питање броја неурона у скривеном слоју(евима). Време обучавања ВР мреже директно зависи од параметра учења η. Додатни проблем се јавља и при избору параметра учења η, тако да се до његове оптималне вредности тешко долази. Овај параметар је изузетно битан за *детерминистичку конвергенцију* мреже ка оптималној вредности грешке E_p. Тако је могуће, да се у неким случајевима, процес обучавања заврши оног тренутка када је максимална грешка коју је мрежа остварила, ипак мања од најмање прихватљиве грешке за проблем који решава тренутна апликација мреже. Међутим, могуће је да ВР неуронска мрежа буде „ухваћена у замку", тако што ће се наћи у локалном минимуму приликом конвергенције ка жељеној минималној грешци.

На слици је дат график који представља попречни пресек хипотетичке комплексне површи грешке E_p у простору тежинских односа. Тачка z_{min} је заправо **глобални минимум**. Као што се види, на слици има и других тачака, z_1 и z_2, које представљају **локалне минимуме**, а градијентни поступак минимизације грешке E_p има за циљ да се уместо ових локалних минимума пронађе глобални минимум. Наиме и поред ових упозорења, конвергенција је онолико успешна, колико је ВР неуронска мрежа у стању да брзо дође до минимално прихватљиве грешке E_p, без обзира да ли се налази у локалном или глобалном минимуму. Понекад се обучавање ВР неуронске мреже зауставља у тачки која се налази негде на правцу простирања градијента ∇E_p, пре него што је стварни минимум остварен, јер је та вредност грешке E_p задовољавајућа. То управо говори о комплексности проблема конвергенције.



Локални и глобални минимум грешке E_p у простору тежинских односа w

Поред концепта детерминистичке конвергенције, примењује се и концепт *стохастичке конвергенције* (*пробабилистичка конвергенција*). На основама стохастичке конвергенције, у спроведеним истраживањима (З.Миљковић) је реализовано тражење коначне вредности грешке E_p преко локалних минимума, све до тренутка када се она нађе у глобалном минимуму, односно када достигне асимптотско стање $\mathbf{w}^*$. Стохастички алгоритам тражења глобалног минимума се може илустровати помоћу слике у наставку.



Стохастички алгоритам тражења глобалног минимума преко локалног минимума

Наиме, ради се о методи стохастичке нелинеарне оптимизације, која има веома важну предност везану за корисну хеуристичку технику напуштања тренутног локалног минимума и конвергирања ка „дубљим" локалним минимумима, све до глобалног минимума, и то за неконвексне функције, попут приказане на претходној слици. Интуитивна идеја, која прати стохастички нелинеарни оптимизациони алгоритам, се односи на евентуално тражење локалних минимума „дубље" у односу на критичне тачке, све док се конвергенција не заврши стриктно у глобалном минимуму, после кога више нема нових „праваца кретања трајекторије конвергенције". На датој слици се овај концепт може јасно и препознати. Практично, стохастички алгоритам тражења је иницран у тачки **w**(0), при чему се трајекторија конвергенције креће ка стриктном локалном минимуму који се налази у $\mathbf{w}_1$. После кратке дивергенције до тачке **w**(3), алгоритам враћа трајекторију конвергенције у **w**(4) уз тренутно задржавање у близини стриктног локалног минимума $\mathbf{w}_1$. Следећа итерација тражења, кретањем ка тачки **w**(6), обезбеђује секвенцијално конвергирање ка глобалном минимуму у **w**(10). Алгоритам претраживања, покушајем „бега" из стриктног глобалног минимума, преко тачака **w**(11) и **w**(12), уз евентуални одлазак и у тачку **w**(13), завршава конвергенцију у стриктном глобалном минимуму $\mathbf{w}^*$. Може се, на основу овог једноставног примера, закључити како стохастички нелинеарни оптимизациони алгоритам у основи ради (иначе математички-нумерички је веома сложен). У практичној примени, кроз спроведене експерименте, показало се да такав алгоритам понекад конвергира спорије ка глобалном минимуму, да би већ следећи пут процес био знатно бржи (Питање: Да ли је брзина конвергенције важна за учење неуронске мреже? → Одговор: Да, за већину проблема везаних за процес одлучивања, посебно када су роботи у питању!).

Конкретно, у оквиру развијеног експерименталног софтвера ***BPnet***, спроведена је описана стохастичка конвергенција, уз претходно обезбеђивање неопходних услова. Пре свега, реализовано је рескалирање вредности улазних и излазних неурона BP вештачке неуронске мреже, као и њихово нормирање, при чему су те вредности сведене на распон од 0.1 до 0.9, управо због успешне и брже стохастичке конвергенције. Када је конвергенција успешно завршена, вредности се конвертују, рескалирањем у супротном смеру, у стварне вредности нпр. углова ротације за зглобове робота (З.Миљковић-књига). Јасно је да су дефинисани и познати, распони могућих вредности улазних неурона (позиција и оријентација објеката је у дефинисаном опсегу) и излазних неурона (углови ротације у зглобовима робота), тако да је и процедура рескалирања једноставно остварена. Уз ове констатације, потребно је нагласити и то да се конвергенција грешке E_p прати и преко неколико контрола софтвера ***BPnet*** које имају задатак да ако, у току стохастичке конвергенције, дође до раста грешке (што је могуће и види се на претходној слици), онда трајекторија конвергенције пролази кроз претходно меморисану тзв. „*min. saved error*", обнављајући вредности тежинских односа између суседних неурона. Веома важну улогу у процесу конвергенције грешке E_p има и параметар учења η, који мора да буде довољно мали (не и сувише) како у току тражења глобалног минимума или „дубљег" локалног минимума не би дошло до тога да мрежа веома „тешко" и споро пронађе уопште било какав прихватљив минимум грешке E_p. За проблем који је у спроведеним истраживањима третиран, оптимална вредност параметра учења η је била 0.2.