Научни трудове на Съюза на учените в България – Пловдив. Серия В. Техника и технологии. Том XVII, ISSN 1311 -9419 (Print); ISSN 2534-9384 (Online), 2019. Scientific Works of the Union of Scientists in Bulgaria - Plovdiv. Series C. Technics and Technologies. Vol. XVII., ISSN 1311 -9419 (Print); ISSN 2534-9384 (Online), 2019

АВТОМАТИЗИРАНО КОНСТРУИРАНЕ И ОБУЧЕНИЕ НА НЕВРОННИ МРЕЖИ В MATLAB

Костадин Йотов, Емил Хаджиколев, Станка Хаджиколева
Пловдивски университет „П. Хилендарски“

AUTOMATED BUILDING AND TRAINING OF NEURAL NETWORKS IN MATLAB

Kostadin Yotov, Emil Hadzhikolev, Stanka Hadzhikoleva
Plovdiv University „Paisii Hilendarski“

Abstract: Neural networks are successfully used in solving complex tasks that require processing of big data characterized with complex dependencies. They find application in different areas – for process automation, risk management, weather analysis and forecasting, consumption, financial markets etc. This widespread application is the reason for the development of various software tools that support neural networks functionalities. The functionalities of Neural Network Toolbox' application Neural Fitting in MATLAB environments are presented in this paper. An Algorithm for automated building and training of neural networks in MATLAB is presented.

Keywords: building of neural networks, training of neural networks

УВОД

С развитието на компютърните технологии и увеличаване на изчислителните мощности на компютърните системи, невронните мрежи се използват все по-активно, за решаване на различни задачи. Те успяват да се справят с проблеми, при които традиционните алгоритми се провалят. Невронните мрежи се използват успешно за: *апроксимиране*, тъй като чрез тях се „изглаждат" сложни нелинейни зависимости, дори когато като входяща променлива се използва само времевият фактор; при *изграждането на предикативни модели*, като в качеството на независими променливи могат да участват различни метрични и неметрични интервениращи променливи (разходи за промоция и реклама, цени и др.); за *отчитане влиянието на рекламната дейност на фирмите*, тъй като благодарение на своята архитектура този вид невронни мрежи позволява да се моделират ефекти, които се проявяват с известен лаг във времето. Други приложения са: *управление на роботизирани системи*; *автоматизация на различни процеси*; *управление на риска*; *прогнозиране* и др. Ето защо не е чудно, че голяма част от приложния софтуер, фокусиран върху математическите изчисления на тези процеси, тяхната обработка или статистика, предлагат готови модули за създаване, обучение и работа с невронни мрежи.

Приложението Neural Net Fitting на Deep Learning Toolbox в средата MATLAB предоставя възможности за конструиране на невронни мрежи. То автоматизира множество дейности, свързани с обучението и тестването на невронни мрежи, но има някои недостатъци – ограничен брой неврони в скрития слой, невъзможност за смяна на активиращите функции както на невроните от скрития слой, така и на тази на изходните неврони, малък брой на методите за обучение на невронната мрежа, фиксиран брой на

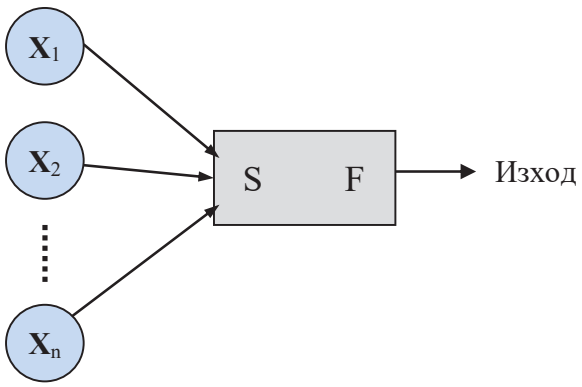
епохите и др. В статията е представена работата по създаване на приложение в средата MATLAB, което има за цел да преодолее тези ограничения, както и да улесни потребителите по отношение на провеждането на множество тестове с различни параметри, съпровождащи процеса на създаване на невронна мрежа.

ИЗКУСТВЕНИ НЕВРОННИ МРЕЖИ

Още от самото си възникване, системите с изкуствен интелект се развиват в две основни направления. *Първото направление* включва системи, имитиращи разумна дейност, характерна за човека, но използващи методи и средства, различни от човешките. Типичен представител на това направление са експертните системи. *Второто направление* се развива върху идеята за създаване на софтуерни системи, наподобяващи работата на вериги от мозъчни неврони – ***изкуствени невронни мрежи*** (Кирова, 1995; Нишева, 1995).

Човешкият мозък съдържа около 25 милиарда нервни клетки – неврони, които чрез своите синапси общуват помежду си по различни начини – химически, електрически или смесени. Посредством дендритната си система, невронът получава информация от вътрешните органи, жлези, мускули, стави или от аксонните окончания на предхождащите го неврони. По този начин се образуват вериги, състоящи се понякога от само два или три, но достигащи до стотици хиляди нервни клетки. Приета чрез дендритите информацията постъпва в тялото на неврона, където се обработва, образувайки биоелектрически потенциал с определена стойност. Колкото е по-значим стимулът, т.е. колкото е по-важна постъпилата информация, толкова по-голям е и създаденият потенциал. Ако входните данни са особено важни, този потенциал надхвърля праговата стойност на неврона и сигналът се разпространява по аксона, като в крайна сметка се предава чрез синапсите към дендритната система на следващият неврон (Дудел, Рюегг и др., 1985).

Създадената върху естествения модел структура на изкуствен неврон е представена на фиг. 1. Входните данни X_i постъпват към неврона чрез дендритната му система, суматорната функция свързва тези данни с теглата по дендритите, създавайки един общ стимул S, който от своя страна се подлага на активираща функция F, характерна за съответния неврон. Така в крайна сметка се формира сигнал, който се предава по аксона към дендритите на невроните от следващия слой.



Фигура 1. Модел на неврон

Суматорната функция S се представя чрез скаларното произведение на входния вектор $X(x_1, x_2, \dots, x_n)$ и вектора от теглата $W(w_1, w_2, \dots, w_n)$ (Барский, 2004): $S = XW$, т.е.

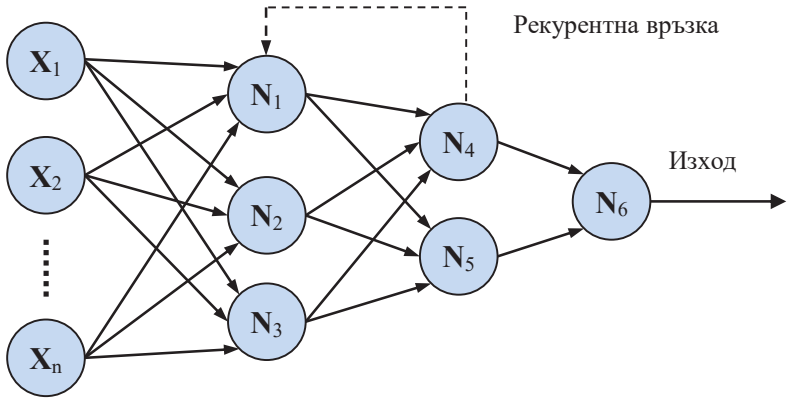
$$(1)\ S = \sum_{i=1}^{n} x_i w_i = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n$$

За да дефинираме прага на неврона към сумата (1) добавяме още един член b, което е еквивалентно на добавянето на допълнителна координата на векторите X и W, по следния начин: $X(x_0 = 1, x_1, x_2, \dots, x_n)$, и $W\ (w_0 = b, w_1, w_2, \dots, w_n)$.
Така сумата (1), с включен праг на неврона, придобива окончателното представяне:

$$(2)\ S = \sum_{i=0}^{n} x_i w_i = \mathrm{b} + x_1 w_1 + x_2 w_2 + \cdots + x_n w_n$$

В процеса на обработка на информацията от страна на неврона, тази сума се подлага на действието на ***активиращата функция F***, чрез която той изработва изходен сигнал, предаващ се към следващи елементи в мрежата. Функциите, активиращи неврона, могат да бъдат от различен тип, и изборът им на практика се определя от задачата на мрежата или желаната точност. Често използвани активиращи функции са: линейна, сигмоидална, стъпална, хиперболичен тангес и др. (Кирова, 1995; Нишева, 1995).

След построяването на отделните нервни клетки, следващата задача е свързването им една с друга в мрежа. Решенията за броя на слоевете на невронната мрежа, броя на невроните във всеки слой, и каква да е връзката между тях, се вземат от разработчиците на мрежата и зависят от типа на решаваната задача. В зависимост от вида на връзките между невроните различаваме многобройни ***мрежови архитектури***, които могат да се класифицират в две големи групи (Хайкин, 2016):



Фигура 2. Модел на изкуствена невронна мрежа: права и рекурентна

- ***Прави мрежи*** – данните се предават строго от входните към изходните неврони, без да съществуват никакви обратни връзки;
- ***Рекурентни мрежи*** – включват и обратни връзки, излизащи от невроните на един слой и завършващи на входа на неврони от същия или предходен слой (фиг. 2).

За да започне работа невронната мрежа, също както биологичните невронни системи, които не се раждат програмирани със знание и способности, тя трябва да бъде предварително обучена.

Според ***вида на обучението*** са утвърдени две основни направления (Кирова, 1995):

- ***Асоциативно обучение***, известно още като обучение с учител. При този тип обучение мрежата се учи, като ѝ се предоставят входни данни и съответстващ им изходен образец. При обучението теглата на мрежата се уточняват, като се използва разликата между стойностите от изходните неврони при зададените входни образци.
- ***Самоорганизация***, или обучение без учител. При този метод изходните неврони се обучават да отговарят на класове от входни образци. Тази идея предполага, че невронната мрежа открива най-характерните черти на входната популация. За разлика от обучението с учител, в случая не съществува множество от категории, с които да се съпоставят входните данни и по-скоро мрежата сама трябва да изработи собствено представяне на входните стимули.

Самото обучение на невронни мрежи е изградено на принципи за „нагласяване“ на теглата на връзките между отделните неврони, съгласно някакво модифицирано правило,

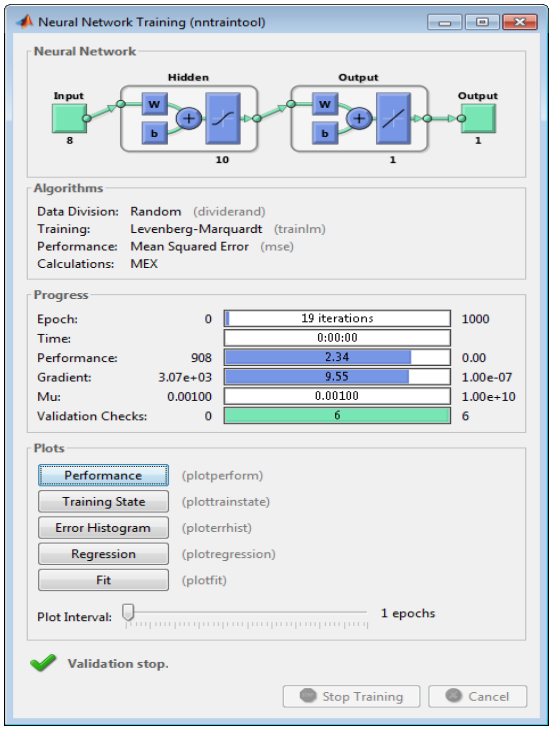
което следва модели, базирани на идеите на Доналд Хеб (Hebb, 1949). Съществуват много и най-различни по характер ***методи за обучение*** (Мръчев, Недева и др., 2007; Hagan, Demuth и др., 2014), които са подходящи за използване в различни ситуации.

ИЗПОЛЗВАНЕ НА NEURAL NET FITTING TOOL ЗА КОНСТРУИРАНЕ НА НЕВРОННИ МРЕЖИ В MATLAB

Приложението Neural Net Fitting tool на (фиг. 3) е чудесно изпълнение на идеята за конструиране на невронни мрежи и тяхното обучение посредством асоциативната парадигма (MATLAB Documentation).

В четири стъпки са определят ***основни характеристики*** за генериране на невронната мрежа:

1. ***Въвеждане на входни данни и съответните им изходни образци*** от потребителя. При това, потребителят посочва вече създадените входни и изходни променливи в средата на MATLAB.
2. ***Валидиране и тестване на данните***. Основен проблем, който възниква в процеса на обучение на невронните мрежи е свръхобучението. Колкото е по-продължително обучението им, толкова по-добре те ще апроксимират използваните за обучението примери. Но ако обучението не спре в подходящия момент, а именно когато функцията на грешката намери своя минимум, тя отново може да започне да нараства. Това налага да се обърне сериозно внимание на критерия за спиране на обучението. Ето защо, разполагаемите данни, след разбъркване по случаен начин чрез функцията на MATLAB dividerand, се разделят на три части, като всеки набор данни за валидиране и тестване е зададен на 15% от първоначалните данни. С тези настройки входните вектори и целевите вектори се разделят, както следва:



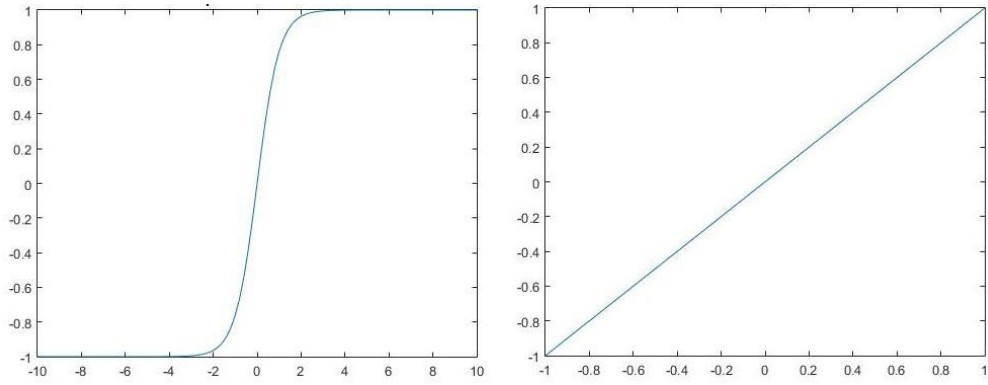
Фигура 3. Екран от процеса на обучение на неврона мрежа в Neural Net Fitting tool на MATLAB

   - 70% се използват за обучение;
   - 15% се използват, за да се установи, че мрежата е оформена и да спре обучението, преди да настъпи свръхобучение;
   - последните 15% се използват като напълно независим тест на мрежата.

   Тези настройки могат да се променят ръчно от потребителя, като се има предвид, че тяхната сума трябва да е равна на 100.
3. ***Избор на архитектура на невронната мрежа.*** Стандартният модул на MATLAB е пригоден за изграждане на стандартни двуслойни мрежи с право предаване на сигнала (прави двуслойни мрежи). По подразбиране, броят на невроните в скрития

слой е 10, но потребителят може да го променя, в зависимост от своите нужди. Активиращата функция, която се използва за невроните от скрития слой, по подразбиране е тангенс хиперболичен *tansig*, а за изходния неврон – линейната функция *purelin*.

4. ***Избор на метод за обучение на невронната мрежа.*** Модулът Neural Net Fitting tool предлага три алгоритъма за обучение (MATLAB Documentation):
    - ***Levenberg-Marquardt (trainlm).*** LMA е отличен алгоритъм, който се препоръчва за много проблеми. Има бърза скорост на мрежова конвергенция и е отлично средство за решаване на нелинейни проблеми с помощта на алгоритми, свързани с използване на метода на най-малките квадрати. LMA се използва в много софтуерни приложения за решаване на общи проблеми с построяването на търсени криви. Въпреки това, както при много алгоритми, LMA намира само локален минимум, който не е задължително глобалният минимум. LMA интерполира между алгоритъма на Гаус-Нютон (GNA) и метода на Спускане по градиента. Счита се, че LMA е по-стабилен от GNA, което означава, че в много случаи този подход намира решение, дори ако започва много далеч от глобалния минимум.
    - ***Bayesian Regularization (trainbr)***. Един от ключовите проблеми при изграждането на невронните мрежи е да се предотвратява свръхнастройването им. Bayesian Regularization е един от подходите за достигане на структурната им стабилизация, посредством ограничаване на скритите неврони и подходящо оптимизиране на теглата. Освен това, за някои от задачите, свързани с повече наличие на шум се оказва, че методът се справя много по-добре от LMA, макар и да отнема малко повече време.
    - ***Scaled Conjugate Gradient Algorithm (trainscg)***. Алгоритъм за настройване на теглата, с търсене на минимума на функцията на грешката, чрез спускане по градиента на тази функция.



Хиперболичен тангенс
a=tansig(n)=2/(1+exp(-2*n))-1

Линейна функция
a=purelin(n)=n

Фигура 4. Използвани в Neural Net Fitting tool активиращи функции по подразбиране

След задаване на настройките, потребителят стартира процеса на генериране на невронна мрежа. За да се определи момента на спиране, се следят грешките на примерите от обучаващото множество и на примерите, заделени за сравнение (валидация). В началото и двете грешки намаляват, но когато започне свръхобучението, грешката на примерите, които не са участвали в обучението, започва да нараства и този момент се избира за спиране на процеса.

По подразбиране обучението се прекратява след шест итерации, при които грешката не намалява. Накрая създадената невронна мрежа се тества с останалите 15% тестови данни и се дава оценка на възможността за практическото ѝ използване.

За потвърждаване на ефективността на мрежата потребителят може да използва предоставените възможности в панела „Plots“ (фиг. 3) като „Regression“ (фиг. 4) и „Error Histogram“, в които се наблюдават грешките и отклоненията при процесите на обучение, валидиране и тестване на невронната мрежа. Колкото по-близки са получените изходни резултати „Output“ с образците, зададени в изходната променлива „Target“, толкова конструираната невронната мрежа е по-добра. В идеалният случай е изпълнено равенството $\mathrm{Output} - \mathrm{Target} = 0$, или

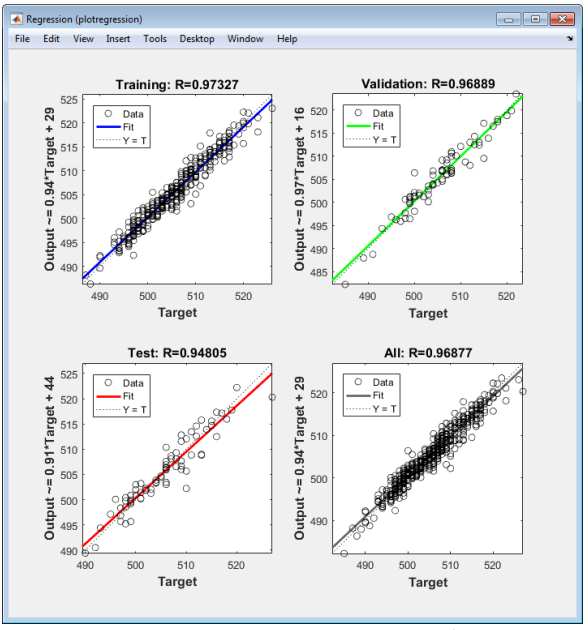
(3) $\mathrm{Output} = \mathrm{Target}$.

Така точките с координати (Target, Output) трябва да лежат колкото е възможно по-близо до линията с наклон 45°, определена от равенството (3).

При това, работата по създаване на невронната мрежа не приключват. На следващата стъпка потребителите на Neural Net Fitting tool могат да направят оценка на мрежата, като я тестват с нови данни. На този етап, ако потребителят не е доволен от ефективността на мрежата, може да направи някое от следните неща:

- да я обучи отново;
- да увеличи броя на невроните;
- да увеличи набора от данни за обучението.

Ако ефективността на обучението е добра (с малки грешки в неговият край), но потребителят не е доволен от независимото тестване на мрежата, това може да означава свръхобучение. В този случай, той може да се върне няколко стъпки назад и да намали броя на невроните. Ако пък ефективността на обучението не е добра (с незадоволителна грешка), то потребителят може да се увеличи броя на невроните и да повтори отново процеса на обучение на мрежата.



Фигура 5. Примерни резултати при обучението на невронна мрежа

При приключване на процеса по създаване на невронната мрежа, потребителят може да запише резултатите в променливи, да генерира скриптове и диаграми, както и да стартира симулация на невронната мрежа.

НЕДОСТАТЪЦИ НА NEURAL NET FITTING TOOL

В общия случай, за решението на конкретна задача потребителят търси ***най-ефективната невронна мрежа***, при която ***грешките и броят на невроните в мрежата да бъдат минимални.*** Neural Net Fitting tool предоставя отлични възможности за обучение и създаване на невронни мрежи: яснота при използването му; подробна визуализация на изпълняваните процеси; възможност да бъде симулирана работата на мрежата чрез MATLAB Simulink; запис на мрежата в средата на MATLAB, чрез което тя да бъде повторно използвана при необходимост и др. За намирането на добро решение обаче, често

на потребителя се налага многократно да повтаря процеса по генериране на мрежа. Някои от ***слабите страни*** на Neural Net Fitting tool са:

1. ***Множество настройки и продължителност на създаване на мрежата***. Потребителят задава входни и изходни данни, брой на невроните в скрития слой, тип на обучението. След това стартира и изчаква процеса по създаване на мрежата, и разглежда самостоятелно получените резултати. При недостатъчно приемливо решение, повтаря процедурата многократно, като променя характеристиките. За определени задачи, една добра невронна мрежа може да съдържа хиляди неврони. Потребителят би могъл да укаже още при първите експерименти голям брой неврони. При това обаче, може да възникне проблем с обучението – то ще става все по-трудоемко, ще се създават все по-обемни структури, съдържащи нови допълнителни тегла по дендритите, връзки по аксоните, допълнителни сложни пресмятания за оптимизирането на теглата и за търсене на минимума на функцията на грешката. Поради това, добър подход при експериментите е броят на невроните да се увеличава с някаква стъпка, започвайки от определен минимален брой, напр. едно. Стъпката може да зависи от субективния опит на потребителя, но търсейки оптимално решение, подходящо е тя също да е единица. При това е възможно провеждането на хиляди експерименти до намирането на добро решение.
2. ***Ограничен брой неврони в скрития слой***. В Neural Fitting е заложен максимален брой на невроните в скрития слой – 9999. Въпреки, че е голям, той може да се окаже решаващо препятствие при по-сложни задачи.
3. ***Невъзможност за смяна на активиращите функции както на невроните от скрития слой, така и на тази на изходните неврони***. По подразбиране са зададени функциите хиперболичен тангенс – за невроните от слоя, и линейна функция – за изходите. Тези функции са идеално средство при решаването на много проблеми, но в редица случай могат да се окажат и не толкова ефективни, колкото използването на алтернативни такива.
4. ***Малък брой на методите за обучение на невронната мрежа***. Изборът на потребителите е сведен единствено до методите: Levenberg-Marquardt, Bayesian Regularization и Scaled Conjugate Gradient.
5. ***Броят на епохите е твърдо установен на 1000*** и не може да бъде променян в Neural Fitting. Епохите определят броя на обучителните цикли, променящи теглата в тренировъчните вектори. Увеличеният им брой в определени случаи би могъл да помогне за откриване на мрежи с по-малък брой на невроните, които се справят с даден проблем, без при това те да са свръхобучени.

АВТОМАТИЗИРАНО КОНСТРУИРАНЕ И ОБУЧЕНИЕ НА НЕВРОННА МРЕЖА

За да се избегнат недостатъците на Neural Net Fitting tool, реализираме алгоритъм за автоматизирано конструиране на невронни мрежи. Използвани са възможностите на рамката Deep Learning Toolbox на MATLAB (в предишни версии Neural Network Toolbox). Освен графични средства за създаване на невронни мрежи – като разгледаното приложение Neural Net Fitting tool, рамката предоставя скриптови възможности за управление на невронните мрежи. Основни дейности с невронни мрежи се извършват с помощта на функции за: създаване на обект за невронна мрежа с един или повече скрити слоя – fitnet и feedforwardnet; обучение – train; тестване на обучението – net; оценка на грешката – perform; графична визуализация – view и др. Един обект за невронна мрежа притежава множество свойства, с които се контролират входните и целеви резултати, активиращи функции, методи за обучение, начините за представяне и запис на резултатите, и др. (MathWorks Documentation).

При реализацията на алгоритъма са поставени следните цели:

1. ***Намаляване на настройките и дейностите, извършвани от потребителя*** – задаване на броя на невроните, методите за обучение, проверка на ефективността на мрежата.
2. ***Автоматична промяна на броя на невроните*** при различните експерименти по създаване на невронни мрежи.
3. ***Неограничен брой на невроните в скрития слой***.
4. ***Използването на по-голям набор от методи за обучение и автоматичното им задаване***.
5. ***Автоматично изпробване на различни активиращите функции*** за невроните от скрития слой.
6. ***Автоматичната промяна на броя на епохите*** при обучението на невронната мрежа. Идеята е то да се случи преди увеличаването на броя на невроните или смяната на метода за обучение и активиращите функции, като по този начин увеличаваме шанса за откриване на мрежа с по-малък брой на невроните.

При това, ***дейностите, извършвани от потребителя*** ще са:

1. ***Подготовка на входно-изходните данни***, необходими за обучението на мрежата.
2. ***Задаване на стойността на критерия, определящ търсената ефективност***.
3. ***Стартиране на кода***.

При реализация на алгоритъма за автоматизирано конструиране и обучение на невронни мрежи използваме множеството от стандартни ***активиращи функции TransferFunctions*** (табл. 1), което в бъдеще може да се разшири, включително и с такива създадени от потребителя, както и множество с ***методи за обучение TrainingFunctions*** (табл. 2).

Таблица 1. Активиращи функции – TransferFunctions

Сигнатура	**Функция**
tansig	Hyperbolic tangent sigmoid transfer function
logsig	Log-sigmoid transfer function
hardlim	Hard-limit transfer function
hardlims	Symmetric hard-limit transfer function
compet	Competitive transfer function
elliotsig	Elliot symmetric sigmoid transfer function
elliot2sig	Elliot 2 symmetric sigmoid transfer function
netinv	Inverse transfer function
poslin	Positive linear transfer function
radbas	Radial basis transfer function
radbasn	Normalized radial basis transfer function
satlin	Saturating linear transfer function
satlins	Symmetric saturating linear transfer function
softmax	Soft max transfer function
tribas	Triangular basis transfer function

Таблица 2. Методи за обучение – TrainingFunctions

Сигнатура	**Алгоритъм**
trainlm	Levenberg-Marquardt
trainbr	Bayesian Regularization
trainbfg	BFGS Quasi-Newton
trainrp	Resilient Backpropagation
trainscg	Scaled Conjugate Gradient
traincgb	Conjugate Gradient with Powell/Beale Restarts
traincgf	Fletcher-Powell Conjugate Gradient
traincgp	Polak-Ribiére Conjugate Gradient
trainoss	One Step Secant
traingdx	Variable Learning Rate Gradient Descent
traingdm	Gradient Descent with Momentum
traingd	Gradient Descent

Основната ***идея на създадения алгоритъм*** е да се ***инициализират, конструират, обучават и тестват невронни мрежи с постепенно нарастващ брой на невроните*** в скрития ***слой за множество от наредени двойки (Активираща функция, Метод за обучение) до намиране на мрежа, удовлетворяваща*** зададен ***критерий за максимална грешка при тестване***. При всяка мрежа ***опционално*** може да се даде възможност за ***обучение с увеличаване на броя на епохите***. Предимства на разглеждания подход са:

- Постепенното нарастване на невроните ще ни гарантира, че ще намерим ***мрежа с минимален брой неврони, която решава удовлетворително зададена задача и не е свръхобучена.***

- ***Намиране на най-подходящата*** за конкретната задача ***комбинация от активираща функция и метод за обучение***. Различни комбинации (Активираща функция, Метод за обучение) са подходящи за различни задачи. При определени случаи една комбинация намира мрежа с по-малък брой неврони и по-малка грешка, а друга – с много неврони и по-голяма грешка.
- ***Обучението с различен максимален брой епохи*** също може да доведе до намиране на мрежа с по-малък брой неврони, но за сметка на неколкократни обучения на мрежа с един и същи брой неврони и комбинация (Активираща функция, Метод за обучение).
- ***Еднократно изпълнение на алгоритъма*** ще ни даде удовлетворително решение. Недостатък при сложни задачи е, че това би могло да изисква многократни обучения на мрежата и съответно – продължително време.
- Дори и ***потребители без достатъчно познания за невронните мрежи биха могли да намерят подходящи решения*** на проблеми върху подготвени от тях данни.

Алгоритъмът „AutoNN“, представен с псевдо код, е следния:

```
inData     = load('inDataFile');     % зареждане на входни данни
targetData = load('targetDataFile'); % зареждане на целеви данни
neurons = 1;              % начален брой неврони в скрития слой
input(maxNeurons);        % въвеждане на максимален брой неврони
input(criterion);         % въвеждане на критерий за край -
                          % напр. максимална грешка 1*10^-10 (1e-10)
% при всяка итерация на основния цикъл невроните се увеличават с един
while(neurons++ < maxNeurons){
  foreach(tf in TransferFunctions){   % за всяка активираща функция...
    foreach(tm in TrainingFunctions){ % ...и за всеки метод за обучение
      net = create(neurons, tf, tm, inData, targetData);
                                      % създаване на мрежа
      train(net);                     % обучение на мрежата
      performance = perform(net);     % резултат от представянето
      if(performance<criterion)){     % ако мрежата е "добра"...
        save(net);                    % ...запис на резултатите,
        view(net);                    % преглеждане на резултатите,
        exit();                       % приключване на алгоритъма
      }
    }
  }
}
```

Фигура 6. Алгоритъм „AutoNN“

В началото се въвеждат: данни, формиращи входните и целевите вектори; максимален брой неврони, с които да се тества мрежата; максимално отклонение от целевите данни, при тестване на мрежата. В три вложени цикъла се променят (по реда на вложеност) броя на невроните, активиращата функция и метода за обучение. При всяка итерация на вътрешния цикъл, последователно: създава се мрежа с указаните параметри; обучава се; оценява се представянето, с автоматично заделената част от данни за тестване; ако представянето е по-добро от зададения от потребителя критерий, то значи е открита мрежа, която може да бъде предложена на потребителя като краен резултат – при това мрежата се записва, представя се визуална информация на потребителя и алгоритъма приключва работа.

Алгоритъмът може да бъде модифициран в следните насоки:

- в четвърти, вложен цикъл, броят на епохите се увеличава неколкократно от 1000 с определена стъпка (напр. също 1000), с което при определени задачи може да се намери мрежа с по-малко неврони;
- избор на по-малки множества от активиращи функции и методи за обучение, с цел по-малък брой итерации и които по субективна преценка на потребителя може да са подходящи за конкретна задача;
- намиране на няколко ефективни мрежи, с които потребителят би могъл да работи с набор от нови данни и др.

ПОЛУЧЕНИ РЕЗУЛТАТИ

Реализираният в MATLAB алгоритъм AutoNN е изпробван за различни двойки входни и целеви вектори. В извършения експеримент участват данни за аргументите и съответните им функционални стойности на няколко конкретни математически функции. Входният вектор е множество от случайни стойности на променливи $x_1 \in [n_1, n_2]$ и $x_2 \in [n_3, n_4]$, а целевият вектор съдържа стойности, получени от изчисленията на избрани за апроксимиране функции $f = \mathrm{x}_1 + \mathrm{x}_2$, $f = \frac{1}{x}$ и др.

В табл. 3 са дадени част от резултатите за невронни мрежи, получени с Neural Net Fitting Tool и алгоритъма AutoNN за автоматизирано конструиране на невронни мрежи. Докато при автоматизираното конструиране може да се намерят мрежи с един неврон и много малка грешка, то с приложението Neural Net Fitting Tool грешката при един неврон е относително голяма. С цел създаването на еднакви начални условия, апроксимацията на всяка отделна функция се извърши чрез използването на едни и същи входни вектори от съответната дефиниционна област.

Таблица 3. Резултати от проведени експерименти

Апроксимирана функция	**Резултати: (неврони, точност активираща функция, метод за обучение)**	
	Neural Net Fitting Tool	**Алгоритъм AutoNN**
$f = \frac{1}{x}$	(1, 3.6e – 4, tansig, Levenberg-Marquardt) (2, 1.3e – 4, tansig, Levenberg-Marquardt)	(1, 9.18e – 19, Inverse transfer function, Bayesian Regulatization) (2, 6.8e – 32, Inverse transfer function, Bayesian Regulatization)
$f = \mathrm{x}^2$	(1, 9129851, tansig, Levenberg-Marquardt) (2, 7.2e-2, tansig, Levenberg-Marquardt)	(1, 0.097, Radial basis transfer function, Levenberg-Marquardt) (2, 7.2e-4, Log-sigmoid transfer function, Levenberg-Marquardt)
$f = \mathrm{x}_1 + \mathrm{x}_2$	(1, 5.23e-5, tansig, Levenberg-Marquardt)	(1, 4.98e-28, Positive linear transfer function, Levenberg-Marquardt)

ЗАКЛЮЧЕНИЕ

Невронните мрежи все по-често се използват за решаването на проблеми с неизвестни зависимости между входни и целеви величини. Създават се все повече библиотеки и инструменти за работа с тях, които могат да бъдат ползвани от потребители с познания в математиката и информатиката. Представеният в статията алгоритъм AutoNN за автоматизирано обучение на невронни мрежи е изграден с възможностите на MATLAB и има за цел да улесни процеса на конструиране и обучение на невронни мрежи, както от

запознати потребители, така и от потребители без познания в областите на математиката и информатиката. Постигнатите при експериментите резултати показват, че алгоритъмът AutoNN открива лесно и бързо по-добри невронни мрежи от такива, създадени с помощта на стандартни инструменти като Neural Net Fitting Tool.

Благодарности: Работата е частично финансирана от проект СП17-ФМИ-005 „Студентска школа за ИКТ иновации в бизнеса и обучението“ към Фонд „Научни изследвания“ при Пловдивския университет „Паисий Хилендарски“.

ЛИТЕРАТУРА

MATLAB Documentation. Достъпно на: https://se.mathworks.com/help/ (последен достъп: 22.11.2018)

Puppo, F., George, V., Silva, G. (2018) An Optimized Structure-Function Design Principle Underlies Efficient Signaling Dynamics in Neurons. Scientific Report. Available at: https://www.nature.com/articles/s41598-018-28527-2 (last access: 22.11.2018).

Барский, А. (2004). *Нейронные сети: распознавание, управление, принятие решений*. Изд. Финансы и статистика – Москва, ISBN 5-279-02757-X.

Дудел, Дж., Рюегг, И., Шмидт, Р., Яниг В. (1985). *Физиология человека*. Изд. Мир – Москва.

Кирова, Т. (1995). *Невронни мрежи – основни архитектури и обучаващи алгоритми*. Изд. Софтех, 1995.

Нишева, М., Шишков, Д. (1995). *Изкуствен интелект*, Изд. „Интеграл” – гр. Добрич, 1995, ISBN: 954-8643-11.

Хайкин, С. (2016). *Нейронные сети. Полный курс. Второе издание*. Изд. Вильямс, 2016, ISBN: 978-5-8459-2069-0.

Hebb, D. (1949). *The Organization of Behavior*. New York: Wiley & Sons.

Мръчев, С., Недева, В., Георгиев, Т., Гинчев, Д. (2007). *Методи за обучението на невронни мрежи*. Международна научна конференция „Наука, техника, технологии и образование“, Ямбол, 2007, стр. 84-96.

Hagan, M., Demuth, H., Beale, M., Jesús, O. (2014). *Neural Network Design* (2nd Edition), Publ. Martin Hagan, ISBN: 097-17321-16.

MathWorks Documentation, *Neural Network Object Properties*. Достъпно на: https://www.mathworks.com/help/deeplearning/ug/neural-network-object-properties.html, (последен достъп: 22.11.2018)