# РАЗПОЗНАВАНЕ НА СИМВОЛИ С ПОМОЩТА НА НЕВРОННА МРЕЖА

**Гочо Гочев, Иван Трифонов, Цветан Шалявски, Иван Вучков**

*Резюме: Извършена е предварителна обработка на изображението, съдържащо цифрова информация. Всеки символ се отделя и центрира в конкретно входно поле. Информацията от това поле се подава към входа на невронна мрежа с цел разпознаване на символа. Избрана е конкретна топология на невронната мрежа. Невроните имат **sigmoid**-предавателни функции. Използува се алгоритъм за обучение с обратно разпространение на грешката. Образците за обучение се вземат от реалното изображение. Тъй като операторът е затруднен при отделянето на образци от всяка цифра, за обучаващото множество е разработен алгоритъм за разпределяне на изображенията на дадена цифра в подмножества. При обучението се взема по един образец от всяко подмножество.*

# CHARACTER RECOGNITION WITH NEURAL NETWORK

**Gotcho Gotchev, Ivan Trifonov, Tzvetan Shaliavsky, Ivan Vutchkov**

***Abstract:** A preliminary processing has been done on the input image. Each character in the image is being extracted and centred within an input field. The information from this field is then feeded to a neural network for recognition of the character. A two layer network of sigmoid neurons has been chosen. Backpropagation learning rule is used. To train the network, characters from the real image are presented to the network. Because of the diversity of the characters an automation procedure for training set selection has been developed to help the operator.*

## 1. ВЪВЕДЕНИЕ

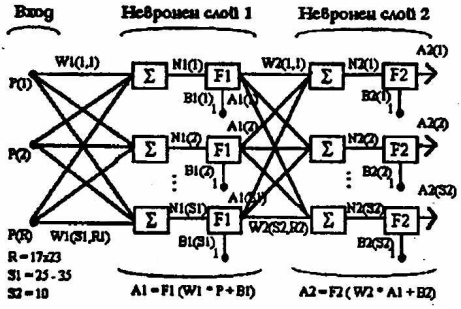
В работата се разглежда невронна мрежа за разпознаване на цифрова информация. Ако в изходния слой на дадена мрежа са организирани осем неврона, всеки например със *sigmoid*-предавателна функция, ограничена между нивата 0 и 1 и за целеви образи се използуват ASCII кодовете на символите, то при съответната топология и обучение на мрежата тя може да се обучи да разпознава пълно множество от символи. При разпознаването на неголям брой символи, например десетте цифри, е възможно за всеки символ да се оп-

Годишник на ТУ–София, т.48, кн.4/1995г.
Proceedings of the Technical University – Sofia, v.48, b.4/1995.

редели конкретен неврон. В този случай изходният слой ще съдържа десет неврона. Приема се, че предварителната обработка на визуалното изображение, съдържащо набори от цифри, е извършена [1]. Всяка цифра е центрирана в конкретно поле. Стойностите от пикселите от това поле се подават към входния слой на невронната мрежа.

## 2. ТОПОЛОГИЯ И ОБУЧЕНИЕ НА НЕВРОННАТА МРЕЖА

В реализираната невронна мрежа за разпознаване на цифрите от 0 до 9 се използува backpropagation алгоритъм за обучение с адаптивна стъпка за редуциране времето на обучение. Избрана е двуслойна невронна мрежа със *sigmoid*-предавателни функции (фиг. 1).




Фиг. 1

Входното поле е избрано с размери 17×23 пиксела. Следователно към всеки неврон от входния слой се подават 391 входа. Конкретната програмна реализация на невронната мрежа позволява броят на невроните в първия слой да е от 25 до 35. В изходния слой се използува унитарно кодиране на символите от 0 до 9, т.е. използуват се 10 неврона.

Поради голямото разнообразие от входни изображения за избирането на обучаващо множество от образи за невронната мрежа е разработен алгоритъм, разпределящ изображенията на дадена цифра в подмножества. В обучаващото множество се включва по един екземпляр от всяко подмножество. Клъстеризирането се извършва с вероятностен класификатор на Байес. Като вход на класификатора се подава след-

ният вектор от стойности, изчислени за всеки един образец:

$$\mathbf{B} = [m_{30} \quad m_{40} \quad m_{50} \quad m_{03} \quad m_{04} \quad m_{05}],$$

$$m_{ij} = \frac{1}{N} \sum_{x=1}^{X} \sum_{y=1}^{Y} \left( \frac{x - \bar{x}}{\sigma_x} \right)^i \left( \frac{y - \bar{y}}{\sigma_y} \right)^j .$$

За обучение на слоевете на невронната мрежа се използува алгоритъмът с обратно разпространение на грешката (backpropagation). Този алгоритъм е създаден, обобщавайки Widrow-Hoff алгоритъма за многослойни мрежи с нелинейни диференцируеми предавателни функции. Входните вектори, отговарящите изходни вектори и векторите-цели се използуват за обучение на мрежата, докато се постигне необходимата точност.

Този алгоритъм се използува за донагласяне на матрицата с теглата и вектора с праговете така, че да се минимизира сумата на квадратичната грешка на мрежата. Това се извършва чрез последователно променяне на стойностите на **W** и **B** в посока на най-стръмното намаление по отношение на грешката. Това се нарича градиентно намаляваща процедура.

В началото на теглата и на праговете трябва да се присвоят неголеми случайни числа между –1 и 1. Това гарантира, че в мрежата няма да има насищане от големи по стойност тегла и предотвратява и други неприятни случаи. Например, ако всички тегла са с еднакво значение, а за реализираната функция трябват различни значения, то мрежата никога няма да се обучи.

Алгоритъмът с обратно разпространение на грешката се използува за трениране на нелинейни, многослойни мрежи за реализирането на апроксимация на функции, класификация и асоциация на образи. Целта на обучението е да се получи такава матрица на теглата и вектор на праговете, така че прилагането на входно множество от вектори да дава като резултат необходимото множество от изходни вектори. За краткост ще наричаме тези множества вектори. При обучението се предполага, че за всеки входен вектор съществува съответен изходен вектор – цел (target). Заедно ще ги наричаме обучаваща двойка. Като правило мрежата се обучава на много двойки. Например, ако искаме да разпознаваме 26 букви от латинската азбука, ще трябва да обучаваме мрежата на минимум от 26 двойки. Реализираната програма за обучение на мрежата свива входното множество до

символите от 0 до 9, което е частен случай, но програмата може да се обучава и на всичките 26 букви плюс цифрите.

Обучението на мрежи с обратно разпространение на грешката изисква изпълнението на следните операции:

1) Избираме следващата двойка за обучение от обучаващото множество; подаваме входния вектор на входа на мрежата.

2) Изчисляваме изхода на мрежата.

3) Изчисляваме разликата между изхода на мрежата и вектора-цел.

4) Коригираме теглата и праговете така, че да минимизираме грешката.

5) Повтаряме стъпки от 1 до 4, докато грешката на цялото множество не падне под предварително зададен праг.

Операциите по изпълнение на стъпки 1 и 2 са подобни на тези, които се изпълняват от вече обучените мрежи, т.е. подава се входен вектор и се изчислява изходът. Изчисленията се правят по слоеве. Първоначално се изчисляват изходите на слой $j$, след това те се използуват за входове на слой $k$ ($k = j + 1$), изчисляват се изходите на слой $k$, които образуват изхода на мрежата (поради избрания модел на двуслойна мрежа).

На стъпка 3 всеки от изходите на мрежата се изважда от съответната компонента на вектора-цел, за да получим грешката. Тази грешка се използува на стъпка 4 на корекции на теглата и праговете на мрежата, като знакът и големината на изменение се определят по формулите, дадени по-долу.

След достатъчен брой повторения на гореспоменатата процедура разликата между действителните изходи и векторите-цели трябва да бъде намалена до приемлива величина. След това мрежата се използува за разпознаване и теглата и праговете повече не се изменят.

На стъпки 1 и 2 можем да гледаме като на **"преход напред"**, понеже сигналът се разпространява от входа към изхода. Стъпки 3 и 4 образуват **"обратния преход"**, понеже изчисляваната грешка се разпространява от изхода към входа. Тези два прехода сега ще бъдат детайлизирани и описани формално.

**Преход напред**

Стъпки 1 и 2 могат да бъдат изразени и във векторна форма по следния начин: подава се входният вектор **X** и на изхода се получава

вектор **Y**. Векторната двойка вход-цел **X** и **T** се вземат от обучаващото множество. Както видяхме, изчисленията в многослойните мрежи се извършват слой след слой, започвайки с най-близкия до входа. Първо изчисляваме стойността *NET* за всеки от невроните, след това тя се подава на предавателната функция и получаваме стойността *OUT* за всеки неврон от текущия слой.

Този процес може да се изрази кратко в матрична форма. Теглата на невроните, както по-горе беше отбелязано, се задават с матрицата **W**. Тогава *NET* векторът може да се опише с матрично умножение: **N** = **XW**. При покомпонентно прилагане на предавателната функция към вектора **N** получаваме изходния вектор **A**. По този начин даденият изчислителен процес се описва със следната формула:

$$\mathbf{A} = F(\mathbf{XW}).$$

**Обратен преход**

Настройка на теглата на изходния слой. Понеже имаме целево значение за изхода на всеки неврон от изходния слой, настройката на теглата за изходния слой се прави лесно с използването на модифицираното делта правило. Вътрешните слоеве се наричат "скрити", за техните изходи няма целеви вектори. Затова и обучението се усложнява.

За процеса на обучение на едно тегло от неврон $p$ в скрития слой $j$ към неврон $q$ от изходния слой $k$ е нужно да изчислим изхода на неврона от слой $k$. След това го изваждаме от стойността на съответния вектор-цел (Target) и така получаваме стойността на грешката. Тя се умножава по стойността на производната и предавателната функция ($OUT(1 - OUT)$), изчислена за неврона от слой $k$, като по този начин получаваме $\delta$:

$$\delta = OUT(1 - OUT)(\text{Target} - OUT).$$

След това умножаваме $\delta$ по стойността *OUT* на неврона от слой $j$, от който излиза разглежданото тегло. Това произведение се умножава по една величина, наречена обучаваща норма (*learning rate*) $\eta$ (обикновено в интервала 0.01 до 1.0), и резултатът се прибавя към теглото. Тази процедура се изпълнява за всяко тегло на невроните от

скрития слой към невроните от изходния слой. Уравнение (1) илюстрира това изчисление:

(1) $$\Delta W_{pq,k} = \eta \delta_{q,k} OUT_{p,j},$$

(2) $$\Delta W_{pq,k}(n+1) = W_{pq,k}(n) + \Delta W_{pq,k},$$

където $W_{pq,k}(n)$ е стойността на теглото от неврон $p$ в скрития слой към неврон $q$ от изходния слой на стъпка $n$ (преди корекцията); нека отбележим, че индексът $k$ се отнася до слоя, в който завършва даденото тегло; $W_{pq,k}(n+1)$ е стойността на теглото за стъпка $n+1$, т.е. след корекциите; $\delta_{q,k}$ е $\delta$ стойността на неврона $q$ в изходния слой $k$; $OUT_{p,j}$ е изходната стойност за неврона $p$ в скрития слой $j$.

Обучени с backpropagation алгоритъм мрежи се опитват да дадат разумни отговори дори и когато като входни вектори се подадат вектори, които мрежата никога не е виждала. Типично един нов вход дава за изход нещо подобно на правилното за входните вектори, на които се е обучавала мрежата и са близки до непознатия входен вектор. Тази генерализираща особеност ни дава възможност да обучаваме мрежата на една представителна извадка от входове/цели и след това мрежата да дава добри резултати за новите входни вектори и без обучението на всички възможни двойки. Обучаваме на една извадка от символи и след това се стремим да разпознаваме всички подобни.

За съжаление алгоритъмът може да води към локален вместо към глобалния минимум. Този локален минимум може да е задоволителен, но ако не е, мрежа с повече слоя и неврони може би ще свърши повече работа. Мрежите, използуващи backpropagation за обучаващ алгоритъм, обикновено имат един или повече скрити слоеве от неврони със sigmoid-предавателни функции, последвани от изходен слой с линейни предавателни функции, позволяващи изходът да не бъде само в интервала [–1, +1]. Скритите слоеве от нелинейни функции позволяват на мрежата да "научи" взаимовръзката между входа и изхода. Чистият backpropagation алгоритъм е бавен заради малката стойност на параметъра скорост на обучение и големия брой променливи в многослойните мрежи. Съществуват методи за подобря-

ване на алгоритъма. Един метод за подобрение е тъй наречения "моментно намаление на чувствителността" към малките детайли. Това помага да се избегне попадането в плитки локални минимуми, които не позволяват на мрежата да намери решение. Началната инициализация може да се направи по-селективно (за двуслойни *sigmoid*/линейни мрежи), отколкото чисто случайната инициализация. Времето за обучение може да бъде намалено също с използването на адаптивна скорост на обучение, която се стреми да държи стъпката на обучение толкова голяма, колкото е възможно за запазването на стабилно обучение. Моментното намаление на чувствителността може да се комбинира с адаптивна скорост на обучение. Backpropagation се използува в около 80-90% от практическите реализации на невронни мрежи.

## 3. АНАЛИЗ НА РЕЗУЛТАТИТЕ

Входното визуално изображение, подавано на невронната мрежа, се състои от сканирания образ на 10 реда броители. Всеки ред съдържа по 10 брояча, всеки с по петразрядна индикация. Тестовете са проведени върху два вида табла с броители. Подробният анализ на резултатите показва, че невронната мрежа постига висок процент на разпознаваемост от 97% до 99%, независимо от факта, че входните изображения са силно зашумени.

За отбелязване е фактът, че невронната мрежа на практика не греши. Тя или разпознава цифрата, или извежда служебния символ "@". След внимателния анализ на изображенията, за които невронната мрежа е извела символа "@", са открити следните причини за неразпознаване:

1) Цифрите са залепени за горната/долната част на броителя.
2) Цифрите попадат в технологична сянка.
3) Цифрите не са центрирани.
4) Причините за неразпознаване са неопределени.

Особено опасни са ситуациите, при които цифрите попадат в сянка, понеже тогава е възможно погрешно разпознаване. Основните грешки, които се допускат, са неправилното разпознаване на 1 и 7,0 и 1, когато някоя от цифрите попадне в сянка.

## ЛИТЕРАТУРА

1. Гочев, Г., Цв. Шалявски, Ив. Трифонов. Предварителна обработка на изображения със символна информация. Сборник трудове, 50 години ТУ, 1995.

2. Уоссермен, Ф. Нейрокомпютерная техника, Москва, Мир, 1992.

## АВТОРИ

**Гочо Вълев Гочев**, доц. д-р, катедра ПИИС, ТУ – София
**Иван Трифонов Търнавски**, експерт, Национална лаборатория по компютърна вирусология, Българска академия на науките
**Цветан Митков Шалявски**, аспирант инж., Национална лаборатория по компютърна вирусология, Българска академия на науките
**Иван Иванов Вучков**, инж., ФЕТТ, ТУ – София