Тема 2/Занятие 5/Лекция

Възможности на програмната среда MATLAB за разработване на програмни решения на типични задачи от областта на идентификацията и разпознаване на образи с използване на невронни мрежи..

лекция

Подходи на обучение на изкуствени невронни мрежи

В постепенно обучение (***incremental training***), теглата и отклоненията на мрежата се актуализират всеки път, когато входът е представен на мрежата.

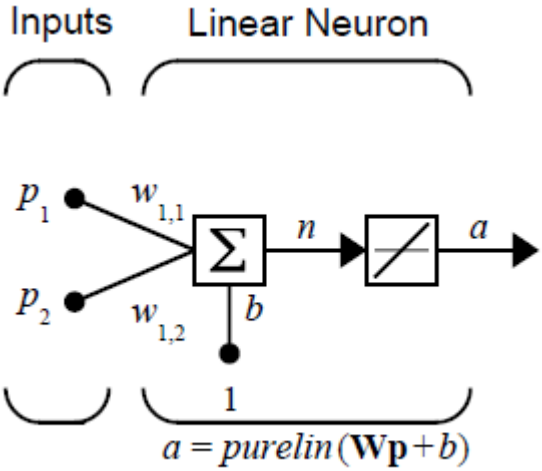
При групово обучение (***batch training***) теглата и отклоненията се актуализират само след представяне на всички входове.

Постепенно обучение (на адаптивни и друг вид мрежи)

Постепенното обучение може да се прилага както за статични, така и за динамични мрежи, въпреки че се използва по-често с динамични мрежи, като адаптивни филтри.

1. Постепенно обучение на статични мрежи

Нека разглеждаме следната статична мрежа.



Искаме да я обучим постепенно, така че теглата и отклоненията да се актуализират след представяне на всеки вход. В този случай използваме функцията **adapt**, и представяме входовете и целите като последователности.

Да предположим, че искаме да обучим мрежата да създава линейната функция

***t* = 2*p*1 + *p*2**

Тогава входовете ще бъдат:

$$\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix}2\\1\end{bmatrix}, \mathbf{p}_3 = \begin{bmatrix}2\\3\end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix}3\\1\end{bmatrix}$$

А изходите ще бъдат:

$$\mathbf{t}_1 = \begin{bmatrix}4\end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix}5\end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix}7\end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix}7\end{bmatrix}$$

Първоначално настройваме мрежата с нулеви първоначални тегла и отклонения. Задаваме скоростта на обучение на нула, за да покаже ефекта от нарастващото обучение.

```
net = newlin([-1 1;-1 1],1,0,0);
net.IW{1,1} = [0 0];
net.b{1} = 0;
```

За постепенно обучение искаме да представим входовете и целите като последователности:

```
P = {[1;2] [2;1] [2;3] [3;1]};
T = {4 5 7 7};
```

Знае се, че при симулация статичната мрежа мрежата произвежда едни и същи изходи, независимо дали входовете са представени като матрица от едновременни вектори или като клетъчен масив от последователни вектори.

Това обаче не се отнася за обучение на мрежата.

Когато използвате функцията за адаптиране, ако входовете са представени като клетъчен масив на последователни вектори, тогава теглата се актуализират при всеки вход представен (постепенен режим).

Както се вижда в следващия раздел, ако входовете са представени като матрица от едновременни вектори, тогава теглата са актуализира се само след представяне на всички входове (пакетен режим).

За постепенно обучение на мрежата се въвежда командата:

[net,a,e,pf] = adapt(net,P,T);

Мрежовите изходи ще останат нула, тъй като скоростта на обучение е нула, и теглата не се актуализират. Грешките ще бъдат равни на целите:

a = [0] [0] [0] [0]

e = [4] [5] [7] [7]

Ако сега зададем скоростта на обучение на 0,1, можем да видим как е мрежата коригирани при представяне на всеки вход:

net.inputWeights{1,1}.learnParam.lr=0,1;

net.biases{1,1}.learnParam.lr=0.1;

[net,a,e,pf] = adapt(net,P,T);

a = [0] [2] [6,0] [5,8]

e = [4] [3] [1,0] [1,2]

Първият изход е същият, както беше с нулева скорост на обучение, тъй като не актуализацията се извършва до представяне на първия вход. Вторият изход е различни, тъй като теглата са актуализирани. Тежестите продължават да да се променят при изчисляването на всяка грешка. Ако мрежата е способна и скоростта на обучение е зададена правилно, грешката в крайна сметка ще бъде доведена до нула.

2. Постепенно обучение на динамични мрежи

Можем също да обучаваме динамични мрежи постепенно. Всъщност това би било най-честата ситуация. Да вземем линейната мрежа с едно закъснение на входа, който използвахме в предишен пример. Инициализираме тегла на нула и задайте скоростта на обучение на 0,1.

```
net = newlin([-1 1],1,[0 1],0.1);
net.IW{1,1} = [0 0];
net.biasConnect = 0;
```

За да обучим тази мрежа постепенно, ние представяме входовете и целите като елементи от клетъчни масиви.

```
Pi = {1};
P = {2 3 4};
T = {3 5 7};
```

Тук се опитваме да обучим мрежата да сумира текущото и предишното входове за създаване на текущия изход. Това е същата входна последователност, която ние използвани в предишния пример за използване на sim, с изключение на това, че присвояваме първия термин в последователността като начално условие за закъснението. Вече можем последователно обучаване мрежата:

```
[net,a,e,pf] = adapt(net,P,T,Pi);
a = [0] [2,4] [7,98]
e = [3] [2,6] [-1,98]
```

Първият изход е нула, тъй като теглата все още не са актуализирани. Теглата се променят при всяка следваща времева стъпка.

3. Пакетно обучение

Пакетно обучение, при което теглата и пристрастията се актуализират само след всичко входовете и целите са представени, могат да се прилагат както към статични, така и към динамични мрежи. В този раздел обсъждаме и двата типа мрежи.

Пакетно обучение на статични мрежи

Пакетното обучение може да се извърши с помощта на адаптиране или обучение, макар и обучение като цяло е най-добрият вариант, тъй като обикновено има достъп до по-ефективни тренировъчни алгоритми. Постепенното обучение може да се извърши само с adapt;

Нека започнем със статичната мрежа, която използвахме в предишните примери. Скоростта на обучение ще бъде зададена на 0,1.

```
net = newlin([-1 1;-1 1],1,0,0.1);
net.IW{1,1} = [0 0];
net.b{1} = 0;
```

За групово обучение на статична мрежа с adapt, входните вектори трябва да бъдат поставени в една матрица от едновременни вектори.

```
P = [1 2 2 3; 2 1 3 1];
T = [4 5 7 7];
```

Когато извикаме **adapt**, тя ще извика влакове (което е по подразбиране адаптационна функция за линейната мрежа) и learnwh (което е функция за обучение по подразбиране за тегла и отклонения). Следователно, Използва се обучение на Widrow-Hoff.

```
[net,a,e,pf] = adapt(net,P,T);
a = 0 0 0 0
```

e = 4 5 7 7

Обърнете внимание, че всички изходи на мрежата са нула, защото теглата са такива не се актуализира, докато не бъде представен целият комплект за обучение. Ако показваме теглата, които намираме:

»net.IW{1,1}

ans = 4,9000 4,1000

»net.b{1}

ans =

2,3000

Това е различно от резултата, който получихме след едно преминаване на адаптация постепенно актуализиране.

Сега нека извършим същото партидно обучение с помощта на влак. Тъй като Правилото на Widrow-Hoff може да се използва в инкрементален или пакетен режим, може да бъде извиква се от **adapt** или **train**. Има няколко алгоритъма, които могат да бъдат само използвани в пакетен режим (напр. Levenberg-Marquardt), и така тези алгоритми може да се извика само с **train**.

Мрежата ще бъде настроена по същия начин.

net = newlin([-1 1;-1 1],1,0,0.1);

net.IW{1,1} = [0 0];

net.b{1} = 0;

В този случай входните вектори могат или да бъдат поставени в матрица на едновременни вектори или в клетъчен масив от последователни вектори. Във **train** всеки клетъчен масив от последователни вектори се преобразува в матрица от едновременни вектори. Това е така, защото мрежата е статична и защото тренирайте винаги работи в пакетен режим. Обикновено се използва

работа в паралелен режим когато е възможно, защото има по-ефективен MATLAB изпълнение.

P = [1 2 2 3; 2 1 3 1];

T = [4 5 7 7];

Сега сме готови да обучим мрежата. Ще го обучим само за една епоха, тъй като използвахме само едно преминаване на **adapt**. Функцията за обучение по подразбиране за линейната мрежа е **trainc**, а функцията за обучение по подразбиране за тегла е **learnwh**, така че трябва да получим същите резултати като нас получен с помощта на **adapt** в предишния пример, където по подразбиране адаптационната функция беше train.

net.inputWeights{1,1}.learnParam.lr = 0,1;

net.biases{1}.learnParam.lr = 0,1;

net.trainParam.epochs = 1;

net = train(net,P,T);

Ако покажем теглата след една епоха на обучение, намираме:

»net.IW{1,1}

ans = 4,9000 4,1000

»net.b{1}

ans =

2,3000

Това е същият резултат, който получихме с обучението в пакетен режим в adapt.

При статични мрежи функцията за адаптиране може да реализира инкрементални или пакетно обучение в зависимост от формата на входните данни. Ако данните са представен като матрица от едновременни вектори, ще

се извърши пакетно обучение. Ако данните се представят като последователност, ще се извърши постепенно обучение. Това не е вярно за **train**, която винаги изпълнява пакетно обучение, независимо от това от формата на входа.

Пакетно обучение на динамични мрежи

Обучението на статични мрежи е относително лесно. Ако използваме **train** мрежата се обучава в пакетен режим и входовете се преобразуват в едновременни вектори (колони на матрица), дори и да са първоначално предавани като последователност (елементи от клетъчен масив).

Ако използваме **adapt**, форматът на входа определя метода на обучение. Ако входовете се предават като последователност, тогава мрежата се обучава в инкрементален режим. Ако входовете се предават като едновременни вектори, тогава се използва обучение в пакетен режим.

При динамичните мрежи обикновено се извършва обучение в пакетен режим.

Нека разгледаме отново линейната мрежа със закъснение. Ние използваме коефициент на обучение от 0,02. (Когато се използва алгоритъм с градиентно спускане е необходимо да се използва по-малка скорост от тази за обучение в пакетен режим.

Градиентите се сумират заедно, преди да се определи стъпката на промяна на тегловите коефициенти.

```
net = newlin([-1 1],1,[0 1],0.02);
net.IW{1,1}=[0 0];
net.biasConnect=0;
net.trainParam.epochs = 1;
```

Pi = {1};

P = {2 3 4};

T = {3 5 6};

Искаме да обучим мрежата със същата последователност, която използвахме за постепенно обучение по-рано, но този път искаме да актуализираме само теглата след като всички входове са приложени (пакетен режим).

Мрежата се симулира в последователен режим, тъй като входът е последователен, но теглата се актуализират в пакетен режим.

net=train(net,P,T,Pi);

Тежестите след една епоха на тренировка са

»net.IW{1,1}

ans = 0,9000 0,6200

Това са различни тегла от тези, които бихме получили с инкрементална тренировка, където тежестите ще се актуализират три пъти по време на една преминаване на обучение. За пакетно обучение тежестите са само актуализиран веднъж във всяка епоха.

Литература:

1. David Kriesel, A Brief Introduction to Neural Networks, достъпно на http://www.dkriesel.com/en/science/neural_networks, посетено на 12.08.2022 г.

2. Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. — М.: Высшая школа, 2002. — 184 с. — ISBN 5-06-004094-1.

3. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика = Neural Computing. Theory and Practice. — М.: Мир, 1992. — 240 с. — ISBN 5-03-002115-9.

4. Хайкин С. Нейронные сети: полный курс = Neural Networks: A Comprehensive Foundation. 2-е изд. — М.: Вильямс, 2006. — 1104 с. — ISBN 0-13-273350-1.

5. Гульнара Яхъяева, Лекция 3. Персептроны. Обучение персептрон, достъпно на https://intuit.ru/studies/courses/88/88/print_lecture/20531