Тема 2/Занятие 1/Лекция

Базови функции на програмната среда MATLAB за изграждане и обучение на изкуствени невронни мрежи.

1. Предназначение и общо описание на програмната среда MATLAB.

MathWorks предоставя няколко продукта, които са особено подходящи за видове задачи, които могат да бъдат изпълнявани с NeuralNetwork Toolbox, описани в таблица 1.

MATLAB е програмна среда за автоматизиране на числени пресмятания и самостоятелен програмен език от четвърто поколение. Както подсказва името, системата MATLAB (от MATrix LABoratory“ - на български: „матрична лаборатория“) е специално създадена за извършване на матрични изчисления: решаване на системи линейни уравнения, разлагане на матрици и т.н.

Освен числен анализ софтуерната среда позволява изчертаване графики на функции, представяне на данни, програмна реализация на алгоритми, разработка на човеко-машинни интерфейси и интерфейси с други програмни продукти, написани на различни програмни езици.

MatLab се състои от ядро, което представлява съвкупност от програми с различно предназначение и допълнителни компоненти (ToolBox), разширяващи възможностите на ядрото с широк спектър от възможности в областта на анализа на данни, изследването на системи за регулиране, идентификацията, оптимизацията и др.

Продукт	Описание
Control System Toolbox	Инструмент за моделиране, анализиране и проектиране
Data Acquisition Toolbox	MATLAB функции за директен достъп до измерени данни от MATLAB
Database Toolbox	Инструмент за свързване и взаимодействие с повечето ODBC/JDBC бази данни от MATLAB
Financial Toolbox	Функции на MATLAB за количествено финансово моделиране и аналитично прототипиране
Fuzzy Logic Toolbox	Инструмент за подпомагане на овладяването на техниките на размита логика и тяхното приложение при практически проблеми с управлението
Image Processing Toolbox	Пълен пакет за цифрова обработка на изображения и инструменти за анализ за MATLAB
Optimization Toolbox	Инструмент за обща и мащабна оптимизация на нелинейни задачи, както и за линейно програмиране, квадратично програмиране, нелинейни най-малки квадрати
Signal Processing Toolbox	Инструмент за разработване на алгоритъм, сигнал и линеен системен анализ и данни от времеви редове моделиране
Simulink	Интерактивна, графична среда за моделиране, симулиране и създаване на прототипи динамични системи
Spline Toolbox	Инструмент за конструиране и използване на части полиномиални функции
Stateflow	Инструмент за графично моделиране и симулация на сложна контролна логика
Statistics Toolbox	Инструмент за анализ на исторически данни, моделиране системи, разработване на статистически алгоритми и статистика на обучението и преподаването

System Identification Toolbox	Инструмент за изграждане на точни, опростени модели на сложни системи от шумни времеви серии от данни

Основния начин за представяне на данните в Matlab е матрицата. Това дава възможност да се решават задачи, свързани със сложни технически изчисления, изискващи матрично представяне на данните.

В Matlab е вграден специално разработеният собствен програмен език от високо ниво. Той с ориентиран към създаване на технически разчети и инженерно проектиране.

Matlab се състои от следните компоненти:

• развойна среда Matlab;

• програмен език Matlab;

• графична среда с възможности за интерактивен интерфейс;•

библиотеки с готови функции-команди (toolboxes);•

програмен интерфейс с възможност за компилация на разработените потребителски програми на други програмни езици, например C++.

Simulink е съпътстваща на Matlab програма. Сама по себе си тя представлява графична среда за симулационно моделиране на системи и процеси. В Симулинк която библиотеките се състоят не от текстови файлове, а от графични блокове, които се явяват симулационни модели на съответни реални физически елементи, обекти и дори системи.

По подразбиране изчисленията се извършват при стандарт двойна точност (double precision), при което за всяка въведена числена стойност (литерал) или променлива се отделят по 64 бита (8 байта) в режим на плаваща запетая, а за комплексни величини – по 64 бита съответно за реалната и имагинерната компоненти (общо 16 байта). Този факт, наред с използваните модерни числени методи, позволява извършването на сложни математически изчисления в много широк числов диапазон при наличието на минимална грешка от закръгления на междинните резултати.

2. Дефиниране и обучаване на невронна мрежа с използване на вградени функции от Neural Network ToolboxTM.

За да започнем, трябва да декларираме обект вид мрежа от избраната функция, която съдържа променливи и методи за извършване на процес на оптимизация.

Функцията очаква два незадължителни аргумента, представляващи броя на скрити единици (и след това на скритите слоеве) и алгоритъмът за обратно разпространение, който трябва да се използва по време на фазата на обучение. Броят на скритите единици трябва да бъде предоставен като едно цяло число, изразяващ размера на скрития слой, или като вектор на цял ред, чиито елементи показват размера на съответните скрити слоеве. Командата: ***nn = patternne t (3)***

2.1. Създаване на перцептрон . Функция newp

Перцептрон може да бъде създаден с функцията newp

net = newp(PR, S)

където входни аргументи са:

PR е R-на-2 матрица на минимални и максимални стойности за R вход

елементи.

S е броят на невроните.

Обикновено функцията **hardlim** се използва в перцептрони, така че е по подразбиране. Кодът по-долу създава **пецептронна мрежа** с един вход от един елемент вектор и един неврон. Диапазонът за единичния елемент на единичния вход

векторът е [0 2].

net = newp([0 2],1);

Можем да видим каква мрежа е създадена, като изпълним следния код

inputweights = net.inputweights{1,1}

което дава:

входни тегла =

закъснения: 0

initFcn: 'initzero'

научи: 1

learnFcn: 'learnp'

learnParam: []

размер: [1 1]

потребителски данни: [1x1 структура]

weightFcn: 'dotprod'

Мрежовият вход към трансферната функция на hardlim е dotprod, който генерира произведението на входния вектор и матрицата на теглото и добавя отклонението за изчисляване на мрежовият вход.

Функцията за инициализация по подразбиране, **initzero**, се използва за задаване на начални стойности на теглата до нула.

Симулиране. Функция (sim)

Да предположим, че вземем персептрон с един входен вектор от два елемента. Дефинираме мрежата с

net = newp([-2 2;-2 +2],1);

Както отбелязахме по-горе, това ни дава нулеви тегла и отклонения, така че ако искаме конкретно набор, различен от нули, трябва да ги създадем. Можем да зададем двете тегла и едното отместване към -1, 1 и 1, както бяха във фигурата на границата на решението със следния код.

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

net.b{1} = [1];

За да сме сигурни, че тези параметри са зададени правилно, ние ги проверяваме с

net.IW{1,1}

ans =

-1 1

net.b{1}

ans =

Сега нека видим дали мрежата отговаря на два сигнала, по един от всяка страна на перцептронна граница.

p1 = [1;1];

a1 = sim(net,p1)

a1 =

1

и за

p2 = [1;-1]

a2 = sim(net,p2)

a2 =

0

Перцептронът класифицира правилно двата входа.

Имайте предвид, че можем да представим двата входа последователно и да получим изходите също последователно.

p3 = {[1;1] [1;-1]};

a3 = sim(net,p3)

a3 =

[1] [0]

.

Инициализация (init)

Можете да използвате функцията init, за да нулирате мрежовите тегла и отклонения към техните оригинални стойности. Да предположим например, че започвате с мрежата

net = newp([-2 2;-2 +2],1);

За проверка на теглото му с

wts = net.IW{1,1}

което дава, както се очаква,

wts =

0 0

По същия начин можете да проверите дали отклонението е 0 с

bias **= net.b{1}**

което дава

bias =

0

Сега задайте теглата на стойностите 3 и 4 и стойността отклонението на 5 с

net.IW{1,1} = [3,4];

net.b{1} = 5;

Проверете отново теглата и отклонението, както е показано по-горе, за да проверите дали има промяна. Тя е:

wts =

3 4

bias =

5

Сега използвайте init, за да нулирате теглата и отклонението към първоначалните им стойности.

net = init(net);

Можем да проверим, както е показано по-горе, за да потвърдим това.

wts =

0 0

bias =

0

Можем да променим начина, по който се инициализира перцептрон с **init**. Например, можем да предефинираме входните тегла на мрежата и bias initFcns като рандове и след това приложете init, както е показано по-долу.

net.inputweights{1,1}.initFcn = 'rands';

net.biases{1}.initFcn = 'rands';

net = init(net);

Сега проверете теглата и отклонението.

wts =

0,2309 0,5839

bias =

-0,1106

Литература:

1. Demuth, H., M., Beale, Neural Network Toolbox For Use with MATLAB®
2. David Kriesel, A Brief Introduction to Neural Networks, достъпно на http://www.dkriesel.com/en/science/neural_networks, посетено на 12.08.2022 г.
3. Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. — М.: Высшая школа, 2002. — 184 с. — ISBN 5-06-004094-1.
4. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика = Neural Computing. Theory and Practice. — М.: Мир, 1992. — 240 с. — ISBN 5-03-002115-9.

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

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