جزوه درس آزمایشگاه هوش مصنوعی

دانشگاه فنی و حرفه ای دختران اهواز

استاد: مرضیه زارعی

Marziyeh.zareie@gmail.com

www.m-zareie.ir

سرفصل‌ها:

- **محیط کار**
- **متغیرها**
- **ماتریس و آرایه**
- **دستورات شرطی**
- **کار با تصاویر و پیکسل ها در متلب**
- **الگوریتم ژنتیک**
- **شبکه عصبی**

متلب چیست ؟

MATLAB یک پلتفرم برنامه‌نویسی است که مخصوص مهندسین و دانشمندان طراحی شده است. قلب پلتفرم متلب زبان برنامه نویسی متلب است. زبانی مبتنی بر ماتریس که امکان اجرای عملیات ریاضیات و محاسباتی را به ساده‌ترین شکل ممکن برای شما فراهم کرده است MATLAB . مخفف عبارت "MATrix LABratory" به‌معنی **آزمایشگاه ماتریس** است. از نام نرم افزار متلب پیداست که شما با یک نرم‌افزار برپایه ماتریس سروکار دارید. این یک نرم‌افزار برای انجام کارهای ریاضی است.

ویژگی های متلب برای مهندسان و دانشمندان

زبان متلب ریاضی است

مهندسان و دانشمندان به یک زبان برنامه نویسی نیاز دارند که به‌راحتی عملیات ریاضی انجام دهند. جبر خطی در متلب به‌راحتی قابل استفاده است. البته این سادگی در پیاده‌سازی ایده‌ها در بسیاری از زمینه‌ها مانند تجزیه و تحلیل داده‌ها، پردازش سیگنال و تصویر، طراحی کنترل نیز صادق است.

نرم افزار متلب برای مهندسان و دانشمندان

در متلب همه‌چیز مخصوص مهندسان و دانشمندان طراحی شده است. نام توابع در متلب به‌گونه‌ای انتخاب شده که به‌راحتی در ذهن افراد باقی بماند. نام توابع در متلب بسیار به نام آنها در ریاضی نزدیک است. متلب برای مهندسان و دانشمندان طراحی شده است، نه برنامه‌نویس‌ها... پس اگر برنامه‌نویس‌ها متلب بلد نیستند، دلیل بر بد بودن متلب نیست. بلکه این نرم‌افزار برای مهندسان و دانشمندان است.

جعبه ابزار متلب

متلب شـامل مجموعه زیادی جعبه ابزار (MATLAB Toolbox) اسـت. هر جعبه ابزار کاربرد به‌خصوص خود را دارد. به‌عنوان مثال، اگر شما مهندس عمران، برق و مکانیک هستید، می‌توانید از جعبه ابزار اختصاصی متلب برای عمران، برق و کامپیوتر استفاده کنید. جعبه ابزارهای متلب بسیار زیاد است و شامل بسیاری از رشته‌های علمی می‌شود.

متلب سریع است

نرم افزار با تکنیک‌های پیشـرفته‌ای پیاده‌سـازی شـده اسـت. این ویژگی باعث شـده محاسـبات در ریاضی را برپایه ماتریس‌ها به‌سرعت انجام دهد .

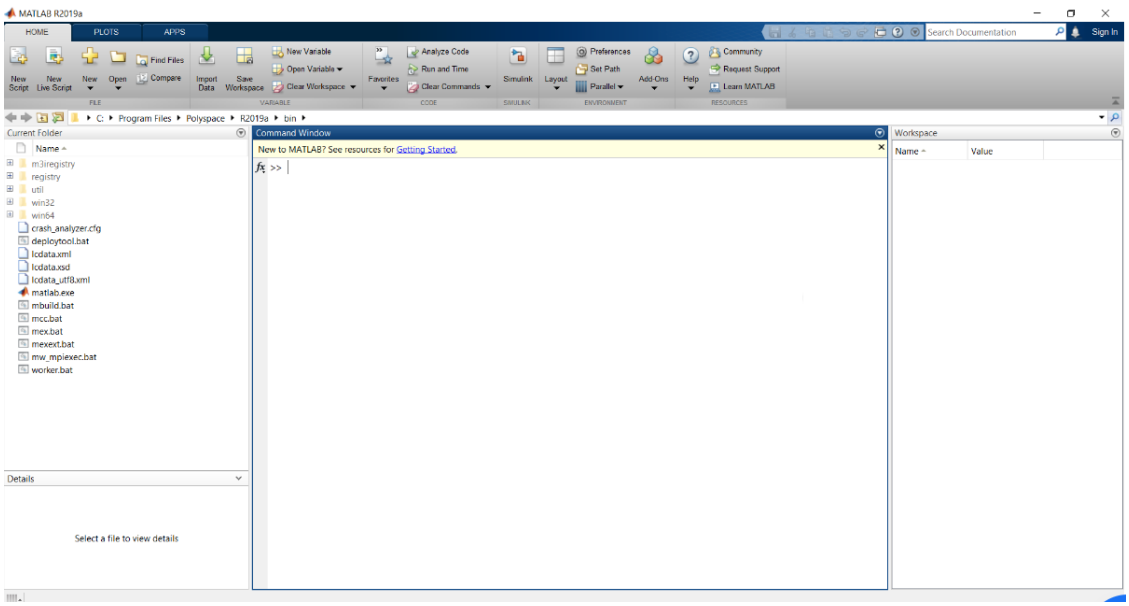
متلب مطمئن است

متلب یک نرم‌افزار پولی است. تیم بزرگی پشت متلب به‌صورت مداوم درحال کار هستند. این باعث می‌شود اطمینان به متلب نسبت به نرم افزارهای سورس- باز بسیار بیشتر باشد.

سرعت پیاده سازی در متلب

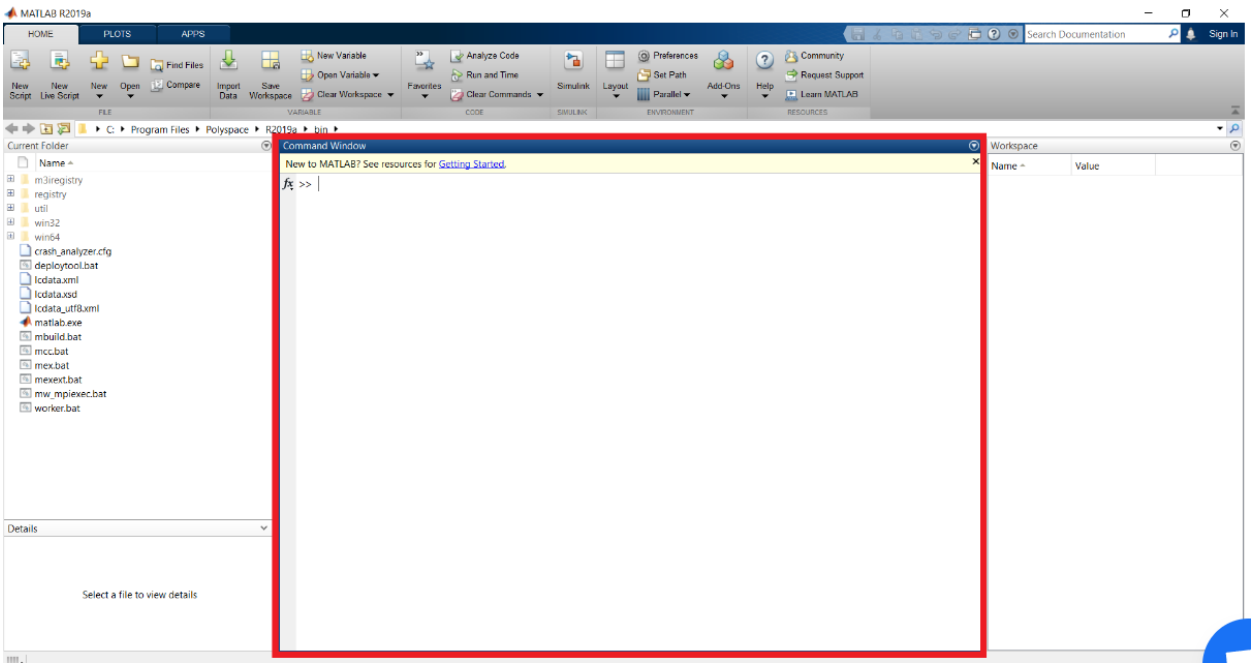
سرعت پیاده‌سازی در متلب بالاست.

شروع کار با متلب



شکل ۱ : آشنایی با محیط نرم افزار متلب

همان‌طور که در پنجره بالا مشاهده می‌کنید، در حالت پیش‌فرض سه پنجره وجود دارد. این پنجره‌ها عبارتنداز: پنجرهWorkspace ، پنجره Command Window در متلب و Current Folder.در حالت نمایش پیش‌فرض بعضی پنجره‌ها مانند Command History فعال نیست. با کلیک روی هرکدام از پنجره‌ها، سربرگ آن پنجره به رنگ آبی درمی‌آید که نشان‌دهنده فعال‌بودن پنجره است.

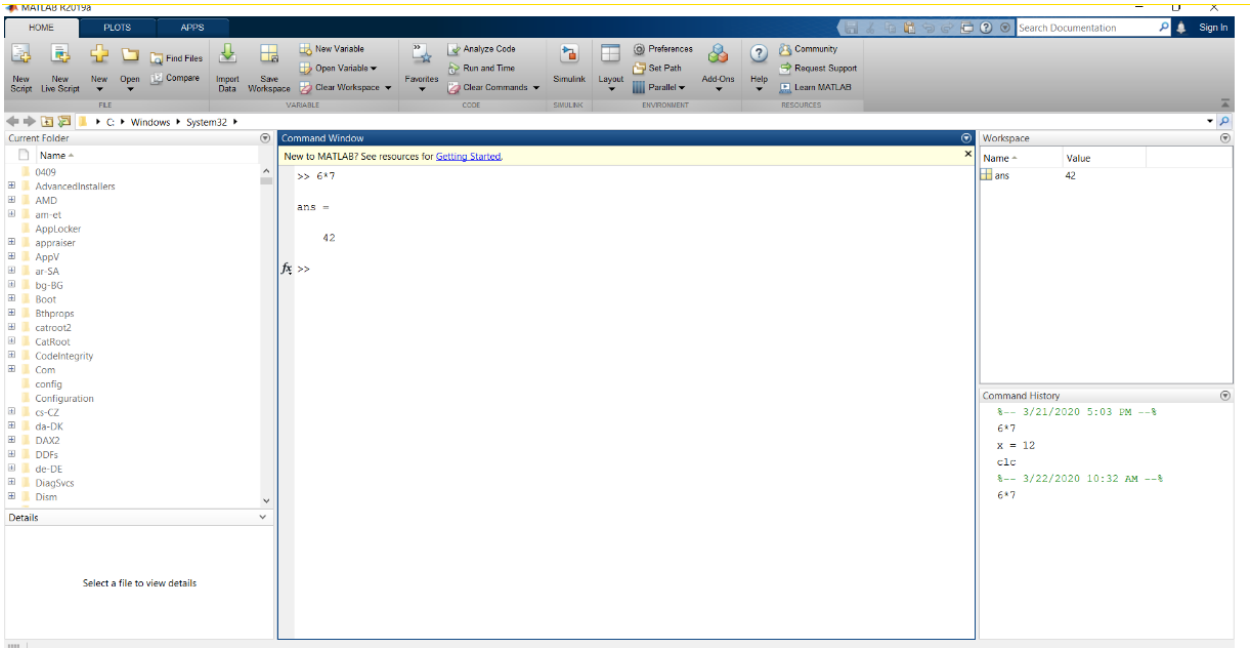


شکل۲ : پنجره Command window در متلب که با کادر قرمز نشان داده شده است.

یکی از مهم‌ترین پنجره‌های متلب، **پنجره Command Window در متلب** است. ترجمه فارسیCommand Window ، پنجره دستور یا فرمان می‌شود. یعنی، پنجره یا محیطی که می‌توانیم دستور، فرمان یا کدهایمان را در آن بنویسیم. این پنجره در تصویر بالا با کادر قرمز نشان داده شده است. در این پنجره یک علامت به شکل >>مشاهده می‌کنید. به این علامت prompt گفته می‌شود. شما باید کدهایتان (دستورات متلب) را جلوی این علامت تایپ کنید. عدم وجود این علامت به این معنی است که متلب آماده دریافت دستور نیست! بعد از وارد کردن دستورات با زدن کلیدEnter ، نتیجه را بلافاصله در همان پنجره می‌توانید مشاهده کنید.

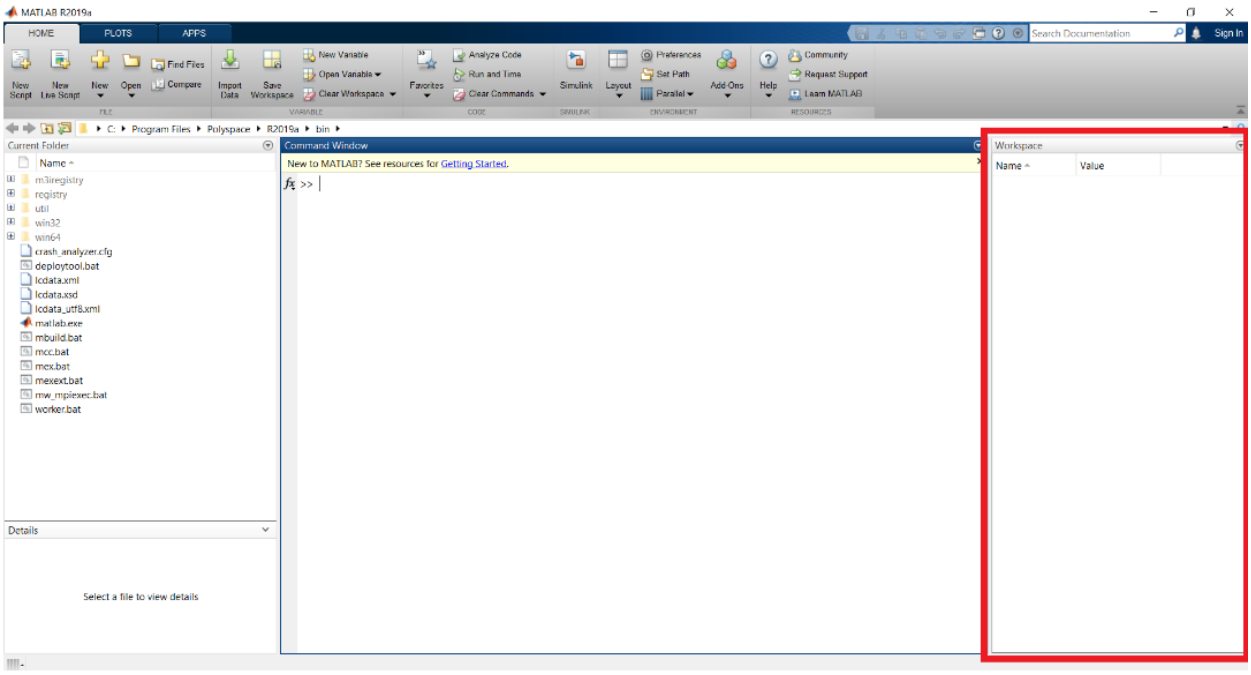
چطور از Command Window استفاده کنیم؟

مثلا فرض‌کنید، به ماشین‌حساب دسترسی ندارید و می‌خواهید از متلب برای انجام محاسبات استفاده کنید! متلب می‌تواند کار یک ماشین‌حساب را انجام دهد! کافی است که محاسبات خودتان را در متلب تایپ کنید و کلید Enter را بزنید. مثلا می‌خواهیم بدانیم ضرب عدد ۶ در ۷ چند می‌شود! کافی است در Command Window تایپ کنیم ۶*۷ و Enter بزنیم.



شکل ۳: نحوه محاسبه ضرب دو عدد ٦ و ٧ در پنجرهCommand Window

پنجره Workspace در متلب



شکل ٤: پنجره Workspace در متلب

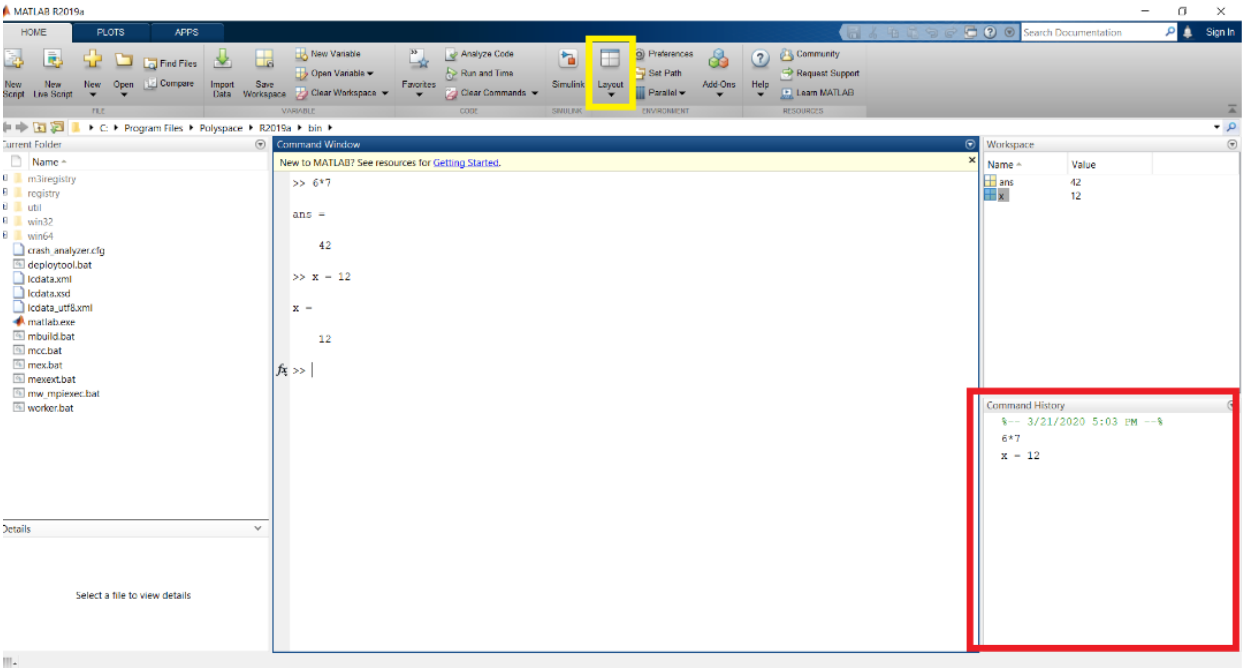
یکی دیگر از پنجره‌های کاربردی متلب، **پنجره workspace** است. این پنجره تمامی متغیرهایی که در Command Window تعریف کرده‌ایم را نشان می‌دهد. یعنی تمام متغیرهایی که قبلا ساختیم، به همراه مقادیرشان در این پنجره نشان داده می‌شوند. مثلا تصویر شکل ۵ نشان می‌دهد که ما تاکنون دو متغیر به نام‌های ans و x تعریف کرده‌ایم. شاید بپرسید که ans از کجا آمد؟ وقتی محاسبه‌ای در متلب انجام داده (مثلا در اینجا ۷*۶) و آن را در هیچ متغیری ذخیره نکنیم، **متلب خودش آن را در متغیر ans ذخیره می‌کند .**چون احتمال می‌دهد، شما به آن عدد نیاز داشته باشید ولی یادتان رفته است که آن را در متغیری بریزید.

نمایش Workspace در متلب

گاهی ممکن است به هر دلیلی، پنجره Workspace در متلب فعال نباشد. در این بخش نحوه نمایش Workspace در متلب را خواهیم گفت. نمایش Workspace در متلب بسیار ساده است. به این منظور کافی است در سربرگHOME ، گزینه Layout را انتخاب کنید. در منوی باز شده تیک Workspace را بزنید. مشاهده خواهید کرد که بعد از این کار، پنجره Workspace در محیط متلب نمایش داده خواهد شد.

پنجره Command History در متلب

همان‌طور که در مقدمه گفته شد، **پنجره Command History** در متلب به‌طور پیش‌فرض فعال نیست. برای فعال کردن این پنجره کافی است روی گزینه layout (کادر زرد در تصویر) کلیک کنید. در لیست باز شده روی گزینه Command History کلیک کنید. لیست جدیدی باز خواهد شد. این لیست شامل سه گزینهpop-up ، dockedو closed است. برای این که پنجره همیشه قابل مشاهده باشد، گزینه docked را انتخاب کنید. با این کار Command Historyبه‌صورت زیر نشان داده می‌شود:



شکل ۵: پنجره Command History در متلب

استفاده از پنجره Command History در متلب

معادل فارسیCommand History، تاریخچه دستور یا فرمان می‌شود .**پنجره Command History تاریخچه دستوراتی که قبلا وارد کرده‌اید را به شما نشان می‌دهد .**هر دستوری که در پنجره Command Window اجرا کرده‌اید، در این پنجره قابل مشاهده است.

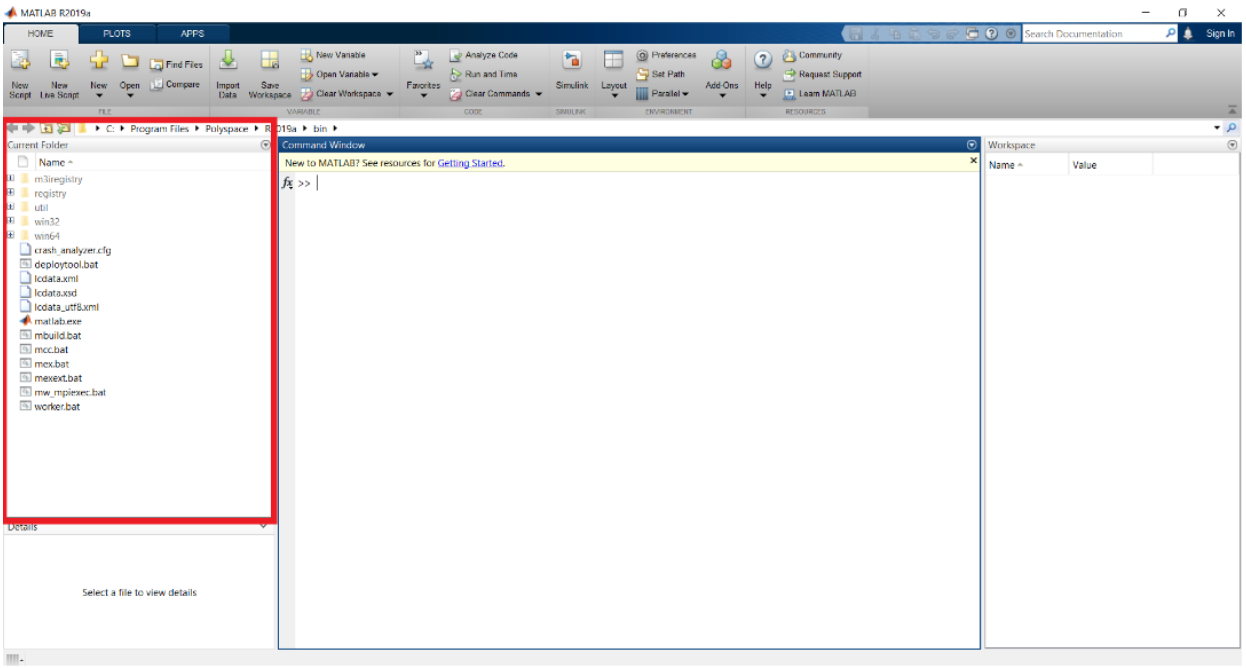
پنجره Command History در متلب به چه دردی می‌خورد؟ ساده است، برای استفاده مجدد از دستورات قبلی که نوشته‌ایم... تصور کنید، شما دیروز در متلب کدنویسی کردید. امروز دوباره متلب را باز کردید و مشغول کار هستید. علاوه بر دستورهای امروز، دستورهایی که دیروز نوشتید هم در پنجره Command History وجود دارد. حتی دستورهای روزها و ماه‌های قبل هم وجود دارد! یعنی با بسته شدن نرم‌افزار متلب، محتویات پنجره Command History پاک نخواهد شد.

ترفند با دابل‌کلیک بر روی هرکدام از دستورها، آن دستور در پنجره Command Windowاجرا خواهد شد.

ترفند در این پنجره قابلیت Drag & Drop وجود دارد. یعنی، با ماوس یک دستور را بکشید و در محیط Command Window بیندازید.

تفاوت دو ترفند بالا در این است که در ترفند دوم، خودتان باید Enter را بزنید تا متلب دستور شما را اجرا کند. اما ترفند اول، خودکار با دابل‌کلیک اجرا می‌شود.

در متلب Current Folder پنجره



شکل ٦: پنجره Current Folder در متلب

پنجره Current Folder در متلب، محتویات پوشه جاری را نشان می‌دهد.

یادآوری: پنجره command window در متلب

در پنجرهcommand window ، جلوی علامت >>می‌توانید دستورات خود را وارد کنید. مثلا برای ضرب دو عدد ۵ و ۳ کافی است بنویسید:

```
>> 3 * 5
ans =
15
```

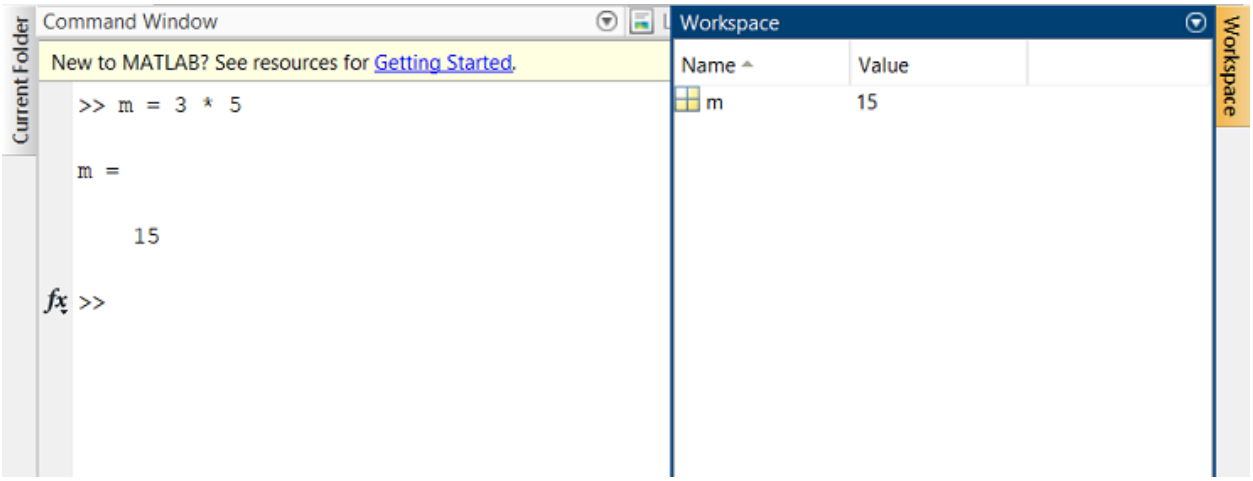
با زدن دکمه Enter خواهید دید که نتیجه در Command Window نمایان می‌شود. متغیرها را نیز همینجا باید بنویسیم! چگونه؟

نحوه تعریف متغیر در متلب

متغیرها برای ذخیره کردن مقادیر استفاده می‌شوند. برای این کار کافی است یک نام برای متغیر انتخاب کنیم. سپس آن را با استفاده از علامت تخصیص یا مساوی (=) با یک مقدار پُر کنیم. مثلا فرض کنید که بخواهیم نتیجه ۵*۳ را در متغیری به نام m ذخیره کنیم. در این صورت کافی است بنویسید m مساوی است با ۳*۵:

```
>> m = 3 * 5
m =
15
```

بعد از اجرای کد بالا، متغیری به نام m تعریف می‌شود که مقدارش برابر با ۳*۵ یعنی ۱۵ است. به پنجره workspace نگاه کنید.



متغیرها بعد از تعریف، در پنجره workspace نمایش داده می‌شوند.

حالا متغیری به نام m اضافه شده که مقدار آن ۱۵ است. مقدار دهی به متغیر در متلب به همین سادگی انجام می‌شود!

عملگر تخصیص در متلب

علامت (=) در دستور بالا چیست؟ علامت تساوی (=) در متلب معادل با عملگر تخصیص است. یعنی عبارت سمت راستِ علامت تساوی (که در مثال قبل ۳ * ۵ است) به متغیر سمت چپ تساوی (که در مثال قبل m است) اختصاص داده می‌شود. وقتی عبارتی مانند x = 3 * 5 را وارد می‌کنید، ابتدا متلب ۳ * ۵ را محاسبه می‌کند و سپس نتیجه (۱۵) را به متغیر x اختصاص می‌دهد.

تمرین ۱: نتیجه ضرب دو عدد ۸ و ۱ را در متغیری بنام s بریزید.

تمرین ۲: یک متغیر به نام y ایجاد کنید که مقدار m / 2 را داشته باشد.

تمرین ۳: دستور m = m + 1 را وارد کنید و ببینید چه اتفاقی می افتد.

قوانین تعریف متغیر در متلب

سه قانون بسیار ساده برای تعریف متغیر در متلب وجود دارد:

۱. متغیرها می‌توانند فقط شامل **حروف، اعداد و علامت "_"** باشند.
۲. متغیرها باید با **حروف شروع شوند.**

۳. متغیرها **حساس به حروف بزرگ و کوچک** هستند.

تمرین ۴: استفاده از کدام متغیرها درست است؟ ۱ velocity (و ۲ s12 (و ۳ m_n (و ۴ m_ (و ۵) ۲۳ mو ۶ Mm (و ۷) ۲ m_و ۸ m___3 (و ۹ s_s_s)

تمرین ۵: مقدار ۲- را به متغیر A اختصاص دهید.

تمرین ۶: مقدار متغیر a را ۱۰ درنظر بگیرید و مقدار ۲ (a + A)/را محاسبه کرده و مقدارش را به متغیر avgAa اختصاص دهید.

متغیر ans در متلب

به **پنجره workspace** نگاه کنید. جهت یادآوری، این پنجره مقدار تمامی متغیرهای تعریف شده را در خود نگه می‌دارد. اکنون، با نوشتن دستور ۳*۵، متغیری به نام ans با مقدار ۱۵ در آن نشان داده می‌شود**ans** .، مخفف answer است. زمانی‌که ما هیچ متغیری برای دستورمان درنظر نگیریم، متلب به‌صورت خودکار جواب را در ans ذخیره می‌کند. چون، شاید بعدا که نیاز داشتیم بتوانیم از آن استفاده کنیم.

علامت semicolon در متلب

در دستوراتی که تاکنون نوشتید، بلافاصله بعد از Enter زدن، نتیجه نشان داده می‌شود. اما با اضافه کردن یک **علامت semicolon** در متلب به انتهای دستور، دیگر خروجی نمایش داده نمی‌شود. قبل از توضیح بیشتر، ابتدا دو دستور زیر را بنویسید تا نتیجه را ببینید:

```
>> p = 10 – 8
p =
2
>> q = 10 – 8;
```

در دستور اول، خروجی p نمایش داده می‌شود. اما در دستور دوم که ; یا همان semicolon

دارد، دیگر مقدار خروجی نمایش داده نشده است. اگرچه همان‌طور که در workspace مشاهده می‌کنید، هر دو متغیر مقدارشان ذخیره شده و جواب نیز درست هست. اما فقط نمایش‌شان متفاوت است. یکی (بدون ;) خروجی را نمایش می‌دهد و دیگری خیر.

شاید این سوال برایتان پیش بیاید که مگر همیشه نباید نتیجه نمایش داده شود؟ **جواب خیر است !**اگرچه در این مثال‌های ابتدایی نمایشِ نتیجه، مطلوب ما هست. اما باید بدانید که گاهی اوقات نمایش نتیجه برایمان دردسرساز خواهد شد! این مسئله پس از آشنایی با آرایه‌ها برای‌تان ملموس‌تر خواهد شد.

مثلا فرض کنید نتیجه کدی که نوشتیم، ماتریسی به ابعاد ۱۰۰۰×۱۰۰۰ باشد. آنگاه بدون semicolon، تمام این یک میلیون عنصر، نمایش داده خواهند شد. بگذریم از اینکه نمایش یک میلیون عدد در Command Window کار مطلوبی نیست. مشکل اینجاست که کلی زمان می‌برد تا یک میلیون عدد چاپ شوند.

تمرین ۷: نتیجه تفریق دو عدد ۸ و ۲ را در متغیری بنام k یک بار با علامت (;) و یک بار بدون (;) بریزید.

Command Historyدر متلب

این دستور را در ننظر بگیرید: m = 3 * 5 برگردیم.

فرض کنید تازه فهمیدیم که این دستور را اشتباه نوشته‌ایم و بجای عدد ۳، باید عدد ۴ می‌نوشتیم. یک راه این است که دستوری که قبلا نوشتیم را مجددا بنویسیم:

```
>> m = 4 * 5
m =
20
```

اما، اگر دستوری طولانی مانند دستور زیر داشته باشیم و فقط بخواهیم یک عدد را عوض کنیم، کار سختی است که همه دستور را دوباره بنویسیم:

```
>> m = 3 * 5 * 6 * 7 * 9 * 10;
```

یک راه ساده برای دسترسی به دستوراتی که قبلا نوشته‌ایم، استفاده از **کلید Up (فلش بالا) و Down**روی کیبورد است. با این دو کلید، به‌راحتی می‌توانید بین دستوراتی که تا اینجا نوشتید سفر کنید و هر کدام را خواستید تغییر دهید! حالا کلید Up را آنقدر بزنید که به دستور m=3*5 برسید و بعد عدد ۳ را با ۴ جایگزین کنید.

تمرین ۸: با کلیدهای Up و Down دستور m = 3 * 5 را انتخاب کنید و بجای عدد ۵، متغیر k قرار دهید.

نحوه حذف یک متغیر در متلب

حذف متغیر در متلب با دستور clear انجام می‌شود. کافی است اسم متغیری که می‌خواهید حذف شود را جلوی دستور clear بنویسید. مثلا برای حذف متغیرm ، کافی است بنویسید:

```
clear m
```

انواع متغیرها در متلب

متغیرهای عددی

این متغیرها میتوانند دارای مقادیر عددی باشند.

مثال:

انتساب مقدار ۲ به متغیر A:

```
A=2
```

متغیرهای رشته ای

چنانچه متغیری را بخواهیم به صورت یک رشته از حروف تعریف کنیم باید از علامت ' استفاده کنیم.

```
s='this is a string'
```

نتیجه:

```
s =
this is a string
```

متغیرهای سمبلیک:

گاهی نیاز است که متغیر تنها به صورت سمبلیک تعریف شود تا با آن معادلاتی را به نمادین حل کنیم. مثلا با حرف x.

چند دستور ساده:

۱- دستور whos:

چنانچه تعداد متغیرهایی که در متلب تعریف کرده اید از حدی بیشتر شود، به سختی می توانید نام آنها را به یاد آورید . برای آنکه بتوانید تمامی متغیرهایی که در متلب تعریف کرده اید را به صورت فهرست وار ببینید تنها کافی است که دستور whos را اجرا کنید. لیست تمامی متغیرهای تعریف شده در متلب در خروجی نمایش داده می شود، همچنین نوع متغیر و فضای نام اختصاص داده شده به آن و اندازه آن را میتوان مشاهده کرد.

مثال:

```
whos
```

نتیجه :

```
Name      Size            Bytes  Class    Attributes

  A         1x1               112  sym
  B         1x1               112  sym
  C         1x1               112  sym
  ans       1x1               112  sym
  x         1x1               112  sym
  y         1x1               112  sym
```

۲- دستور clear:

این دستور برای پاک کردن متغیرهای تعریف شده در متلب به کار می رود . این دستور را می توان به شیوه های زیر به کار برد:

clear	تمامی متغیرهای تعریف شده در متلب را پاک می کند
clear all	تمامی متغیرهای تعریف شده در متلب را پاک می کند
clear x y	تنها متغیرهای x و y را پاک می کند

یک برنامه نویس متلب معمولا اولین دستوری که در برنامه خود به کار می برد دستور clear all می باشد، زیرا امکان این که متغیرهایی که قبلا در متلب توسط برنامه های قبلی تعریف شده اند در برنامه ی جدید اختلال ایجاد کنند، زیاد است.

۳- دستور clc:

این دستور برای پاک کردن اطلاعات نمایش داده شده در پنجره command به کار می رود.

عملگرهای ریاضی مشخص شده در متلب:

در جدول زیر عملگرهای ریاضی در متلب آمده است:

نماد در متلب	عملگر
+	جمع
-	تفریق
*	ضرب
/	تقسیم
^	به توان

مثال :

```
3*2-(3*4)/2+4^2
```

نتیجه :

```
ans =

    16
```

مثال :

برای به توان 2 رساندن عدد 3 این گونه می نویسیم:

```
3^2
```

نتیجه :

```
ans =
```

عملگرهای ریاضی مشخص شده با دستور در متلب:

به جدول زیر توجه کنید.

دستور در متلب	عملگر
sqrt	رادیکال 2
log	Ln (لگاریتم طبیعی)
log10	Log (لگاریتم بر مبنای 10)

مثال :

با دستور sqrt ، رادیکال 2 عدد 4 را محاسبه می کنیم:

```
A=sqrt(4)
```

نتیجه :

```
A =

     2
```

مثال :

با دستورlog10 ، لگاریتم بر مبنای 10 یا log2 لگاریتم بر مبنای 2 و...
لگاریتم بر مبنای 10 عدد 100 را محاسبه می کنیم:

```
A=log10(100)
```

نتیجه :

```
A =

     2
```

دستور format:

به صورت پیش فرض در متلب اعداد تا 4 رقم اعشار نمایش داده می شوند . اگر بخواهیم به فرمت طولانی تر تبدیل کنیم، که تعداد ارقام بیشتری را نشان دهد:

Format long

تا 15 رقم اعشار نمایش میدهد.

مثال :

```
format long
pi
```

نتیجه :

```
ans =

   3.141592653589793
```

برای نمایش اعداد تا دو رقم اعشار از دستور Format short استفاده می‌شود. توجه داشته باشید که هر بار که متلب را ببندیم و باز کنیم به فرمت پیش فرض خود یعنی ۴ رقم اعشار بازمیگردد.

ایجاد نقاط با فواصل خطی:

```
linspace(a,b)
```

که a عدد ابتدا و b عدد انتهاست.

پیش فرض این است که 100 نقطه تعریف شده است.

مثال :

```
Linspace(1,5,5)
```

نتیجه :

```
ans =

     1   2   3   4   5
```

5 نقطه می دهد، با فواصل یکسان.

ایجاد نقاط با فواصل لگاریتمی:

دستور logspace اعداد را به صورت لگاریتمی به صورت های زیر نمایش میدهد.

```
logspace (a,b)
```

دستور فوق بین 10^a و 10^b پنجاه عدد انتخاب میکند

```
logspace (a,b,n)
```

این دستور مانند دستور قبل عمل میکند با این تفاوت که عدد آخر(n) تعداد نقاط را برای ما مشخص میکند.

مثال :

```
logspace (1,2,5)
```

نتیجه :

```
ans =

   10.0000   17.7828   31.6228   56.2341  100.0000
```

گرد کردن اعداد اعشاری در متلب:

چند تابع در متلب برای انجام این کار وجود دارد، به جدول زیر دقت نمایید.

دستور round	گرد کردن به سمت نزدیکترین عدد صحیح
دستور fix	گرد کردن به سمت صفر
دستور ceil	گرد کردن به سمت مثبت بینهایت
دستور floor	گرد کردن به سمت منفی بینهایت

مثال:

دستور round:

چنانچه از دستور round برای گرد کردن عدد اعشاری مورد نظرمان استفاده کنیم ، آنگاه آن عدد اعشاری به نزدیکترین عدد صحیح تبدیل خواهد شد . به مثال زیر توجه کنید:

مثال :

```
round(1.9)
```

نتیجه :

```
ans =

     2
```

دستور fix:

فرض کنید از دستور fix برای گرد کردن یک عدد اعشاری استفاده کنیم ، چون آن عدد اعشاری بین دو عدد صحیح قرار گرفته است باید یکی از آن دو به عنوان گرد شده آن عدد اعشاری انتخاب شود . وقتی می گوییم گرد کردن به سمت صفر یعنی اینکه از بین آن دو عدد صحیح ، عددی انتخاب می شود که به صفر نزدیکتر باشد . به مثال زیر توجه کنید:

مثال :

```
fix(1.9)
```

نتیجه :

```
ans =

     1
```

دستور ceil :

اگر از دستور ceil برای گرد کردن یک عدد اعشاری استفاده کنیم ، از میان دو عدد صحیحی که در دو طرف عدد اعشاری قرار گرفته اند ، عددی انتخاب می شود که به مثبت بینهایت نزدیکتر باشد . به مثال زیر توجه کنید:

مثال :

```
ceil(1.2)
```

نتیجه :

```
ans =

     2
```

دستور floor :

اگر از دستور floor برای گرد کردن یک عدد اعشاری استفاده کنیم ، از میان دو عدد صحیحی که در دو طرف عدد اعشاری قرار گرفته اند ، عددی انتخاب می شود که به منفی بینهایت نزدیکتر باشد .

به مثال زیر توجه کنید:

مثال :

```
floor(1.9)
```

نتیجه :

```
ans =

     1
```

بردارها و ماتریس‌ها در متلب:

نرم‌افزار متلب بر اساس کار کردن با بردارها و ماتریس‌ها ساخته شده است.

یک بردار، فهرستی از اعداد می‌باشد که پشت سر هم چیده شده‌اند. به هر کدام از این اعداد یک عنصر بردار می‌گویند. در متلب به روش‌هایی مختلفی میتوان بردار رسم کرد، به مثال های زیر توجه کنید:

مثال :

```
A=[1,2,3,4]
```

نتیجه :

```
A =

     1     2     3     4
```

روش دوم : استفاده از فاصله برای جدا کردن عناصر بردار و قرار دادن آنها درون براکت

مثال :

```
A=[1 2 3 4]
```

نتیجه :

```
A =

     1     2     3     4
```

ساخت بردارهای قاعده مند:

فرض کنید بخواهیم برداری تعریف کنیم که عناصر آن شامل اعداد ۱ تا ۹ باشد که به ترتیب پشت سرهم باشند ، برای این گونه موارد ، نرم افزار متلب شیوه ای از علامت گذاری را به کار می برد که در مثال زیر نشان داده شده است و دیگر لازم نیست که این ۹ عدد را به شیوه های ذکر شده قبلی در بردار تعریف کنیم:

```
A=1:9
```

نتیجه :

```
A =

     1     2     3     4     5     6     7     8     9
```

تنها تفاوت در این است که میزان افزایش که برابر 1 واحد است را در علامت گذاری ذکر کرده ایم . در واقع دستور A=1:9 نوعی اختصار برای دستور A=1:1:9 در متلب می باشد.

مثال :

فرض کنید بخواهیم اعداد زوج بین 0 تا 10 را به عنوان عناصر یک بردار در متلب تعریف کنیم ، می نویسیم:

```
A=0:2:10
```

نتیجه :

```
A =

     0     2     4     6     8    10
```

اولین عنصر بردار برابر 0 است و هر بار 2 واحد به آن اضافه می شود تا عنصر بعدی بردار ساخته شود تا زمانی که به عدد 10 برسیم که آخرین عنصر بردار می باشد .

میزان افزایش عناصر بردار می تواند یک عدد اعشاری باشد . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم اعداد بین 0 تا 2 را با فاصله 0.2 به عنوان عناصر یک بردار در متلب تعریف کنیم ، می نویسیم:

```
A=1:0.2:2
```

نتیجه :

```
A =

    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
```

میزان تغییر منظم عناصر بردار می تواند به صورت کاهشی باشد . برای این منظور باید یک عدد منفی را ، متناسب با میزان کاهش ، مشخص کنیم . به مثال زیر توجه کنید:

مثال :

```
A=9:-1:1
```

نتیجه :

```
A =

     9     8     7     6     5     4     3     2     1
```

نحوه تعریف ماتریس ها در متلب:

برای تعریف ماتریس ها در متلب ، چندین روش وجود دارد :
روش اول : تعریف عناصر ماتریس با استفاده از علامت ((،)) برای جدا کردن عناصر و استفاده از علامت ((;)) برای جدا کردن ردیف ها از یکدیگر . به مثال زیر توجه کنید:
مثال :

```
A=[1,2,3;4,5,6;7,8,9]
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9
```

روش دوم : تعریف عناصر ماتریس با استفاده از فاصله برای جدا کردن عناصر و استفاده از علامت ((;)) برای جدا کردن ردیف ها از یکدیگر . به مثال زیر توجه کنید:
مثال :

```
A=[1 2 3;4 5 6;7 8 9]
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9
```

روش سوم : نرم افزار متلب برای ایجاد برخی ماتریس های خاص دارای دستوراتی می باشد . برخی از این دستورها عبارتند از:

دستور ones :

این دستور ماتریسی ایجاد می کند که تمامی عناصر آن دارای مقدار عددی 1 می باشند . به مثال زیر توجه کنید:
مثال :
ماتریسی می سازیم که دارای 2 ردیف و 3 ستون باشد و تمامی عناصر آن دارای مقدار عددی 1 باشند:

```
A=ones(2,3)
```

نتیجه :

```
A =

     1     1     1
     1     1     1
```

دستور zeros :

این دستور ماتریسی ایجاد می کند که تمامی عناصر آن دارای مقدار عددی 0 می باشند . به مثال زیر توجه کنید:

مثال :

ماتریسی می سازیم که دارای 3 ردیف و 2 ستون باشد و تمامی عناصر آن دارای مقدار عددی 0 باشند:

```
A=zeros(3,2)
```

نتیجه :

```
A =

     0     0
     0     0
     0     0
```

دستور eye (ساخت ماتریس همانی)

ماتریس همانی ، ماتریسی می باشد که عناصر روی قطر اصلی آن برابر 1 و سایر عناصر آن برابر 0 باشد . چنانچه از دستور eye به صورت eye(n) استفاده کنیم آنگاه دستور eye یک ماتریس همانی با nردیف و n ستون را برمی گرداند . به مثال زیر توجه کنید:

مثال :

```
A=eye(4)
```

نتیجه :

```
A =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

نکته :

همچنین با دستور eye می توان ماتریس هایی با تعداد ردیف و تعداد ستون غیر مساوی ساخت که عناصر روی قطر اصلی آنها برابر 1 و سایر عناصر آنها برابر 0 باشد . به مثال زیر توجه کنید:

مثال :

```
A=eye(4,3)
```

نتیجه :

```
A =

     1     0     0
     0     1     0
     0     0     1
     0     0     0
```

ماتریس تصادفی rand:

ماتریس تصادفی با درجه وارد شده میسازد

rand(n,m)

این دستور ماتریس n*m میسازد که درایه های آن تصادفی می باشد

rand(n)

این دستور ماتریسی مربعی میسازد که درایه های آن تصادفی می باشد

دستور randperm در متلب:

چنانچه از دستور randperm به صورت randperm(n) استفاده کنیم انگاه این دستور در خروجی ، یک بردار را برمی گرداند که در آن بردار ، اعداد صحیح 1 تا n به صورت تصادفی قرار گرفته اند . به مثال زیر توجه کنید:

مثال :

```
A=randperm(9)
```

نتیجه :

```
A =

     6     3     7     8     5     1     2     4     9
```

دستور inv در متلب:

با دستور inv در متلب ، می توانیم معکوس یک ماتریس را محاسبه کنیم . به مثال زیر توجه کنید:

مثال :

```
A=[10 12 13;4 5 6;7 8 9]
B=inv(A)
```

```
   -1.0000   -1.3333    2.3333
    2.0000   -0.3333   -2.6667
   -1.0000    1.3333    0.6667
```

دستور det در متلب:

با استفاده از دستور det در متلب ، می توانیم دترمینان یک ماتریس را محاسبه کنیم . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=det(A)
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

  6.6613e-016
```

نکته :

جهت محاسبه دترمینان و معکوس یک ماتریس حتما باید ماتریس مربعی باشد.

محاسبه ترانهاده یک ماتریس در متلب:

ترانهاده یک ماتریس ، ماتریسی می باشد که در آن ، نسبت به ماتریس اولیه ، جای سطرها و ستون ها با هم عوض شده باشد . یعنی عناصر سطر اول به جای عناصر ستون اول و عناصر ستون اول نیز به جای عناصر سطر اول قرار گیرند و عناصر سایر سطرها و ستون ها نیز به همین شکل جایشان با یکدیگر عوض شود . در متلب برای آنکه ترانهاده یک ماتریس را محاسبه کنیم باید پس از نام آن ماتریس ، علامت ' را به کار ببریم . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=A'
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

     1     4     7
     2     5     8
     3     6     9
```

انجام عملیات ریاضی بر روی عناصر یک ماتریس:

در برنامه های متلب ، بسیار زیاد پیش می آید که بخواهیم یک عمل ریاضی را بر روی تمامی عناصر یک ماتریس انجام دهیم . نرم افزار متلب برای این موارد از نوعی علامت گذاری استفاده می کند که در مثال زیر بیان شده است:

مثال :

فرض کنیم بخواهیم تمامی عناصر ماتریس A را به توان 2 برسانیم ، می نویسیم:

```
A=[1 2 3 4]
B=A.^2
```

نتیجه :

```
A =

     1     2     3     4


B =

     1     4     9    16
```

نکته :

اهمیت علامت نقطه ((.)) بسیار زیاد است زیرا این علامت است که مشخص می کند که عمل ریاضی مشخص شده بر روی هر عنصر ماتریس صورت گیرد و در صورتی که علامت نقطه گذاشته نشود آن عمل ریاضی بر روی کل ماتریس انجام می شود که مطمئنا نتیجه ای غیر از آنچه می خواستیم به ما خواهد داد .

توجه داشته باشید که در صورت قرار ندادن علامت نقطه، چنانچه آن عمل ریاضی قابل اجرا روی کل ماتریس نباشد، متلب پیام خطایی صادر میکند.

.

مثال :

فرض کنید عناصر متناظر دو ماتریس A و B را بخواهیم تک تک در هم ضرب کنیم ، می نویسیم:

```
A=[1 2 3 4]
B=[2 3 4 5]
C=A.*B
```

نتیجه :

```
A =

     1     2     3     4


B =

     2     3     4     5


C =

     2     6    12    20
```

اشاره به عناصر مختلف یک ماتریس

گاهی لازم است که به یک ردیف یا یک ستون یک ماتریس ، اشاره داشته باشیم . برای این منظور می توان از علامت ((:)) استفاده نمود . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=A(2,:)
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

     4     5     6
```

همانطور که میبینید در دستور B=A(2,:) عدد ۲ برای شماره ردیف و علامت : برای شماره ستون به کار رفته است. یعنی اینکه عناصری مورد نظرمان است که شماره ردیف آنها برابر ۲ باشد اما شماره ستون آنها میتواند شماره هر ستونی باشد. بنابراین حاصل برابر تمامی عناصر ردیف دوم ماتریس A میباشد.

اشاره به چند عنصر متوالی از یک ردیف با یک ستون ماتریس

ممکن است بخواهیم از یک ردیف با یک ستون تنها به چند عنصر متوالی آن اشاره کنیم. به این مثال توجه کنید:

```
A=[1 2 3 4;5 6 7 8;9 10 11 12]
B=A(2,2:4)
```

نتیجه :

```
A =

     1     2     3     4
     5     6     7     8
     9    10    11    12


B =

     6     7     8
```

در اینجا عدد ۲ برای شماره ردیف و عبارت ۲:۴ برای شماره ستون نوشته شده است. بنابراین B برابر عناصری از ماتریس خواهد بود که شماره ردیف آنها برابر ۲ و شماره ستون آنها از ۲ تا ۴ می باشد.

اشاره به چند عنصر غیر متوالی از یک ردیف یا یک ستون ماتریس:

گاهی ممکن است عناصر مورد نظرمان متوالی نباشد. در اینگونه موارد نمیتوانیم از علامت : استفاده کنیم و باید شماره ردیف یا ستون عناصر مورد نظرمان را درون علامت‌های [] قرار بدهیم.

به مثال زیر توجه کنید.

```
A=[1 2 3;4 5 6;7 8 9]
B=A(2,[1 3])
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

     4     6
```

محاسبه مینیمم (min) یک ماتریس در متلب:

در متلب برای محاسبه مینیمم (min) یک ماتریس از دستور min استفاده می شود . اگر A یک ماتریس دو بعدی باشد دستور min(A) ، برداری را بر می گرداند که در آن مینیمم هر ستون ماتریس Aمشخص شده است و دستور
min(min(A))، مینیمم ماتریس A را محاسبه می کند . به مثال زیر توجه کنید:

مثال :

```
A=[1 2;3 4]
B=min(A)
C=min(min(A))
```

نتیجه :

```
A =

     1     2
     3     4


B =

     1     2


C =

     1
```

محاسبه ماکزیمم (max) یک ماتریس در متلب:

در متلب برای محاسبه ماکزیمم (max) یک ماتریس از دستور max استفاده می شود . اگر A یک ماتریس دو بعدی باشد دستور max(A) ، برداری را بر می گرداند که در آن ماکزیمم هر ستون ماتریس A مشخص شده است و دستور
max(max (A))، ماکزیمم ماتریس A را محاسبه می کند . به مثال زیر توجه کنید:

مثال :

```
A=[1 2;3 4]
B=max(A)
C=max(max(A))
```

نتیجه :

```
A =

     1     2
     3     4


B =

     3     4


C =

     4
```

دستور length در متلب:

در متلب با استفاده از دستور length می توانیم طول یک بردار (تعداد کل عناصر بردار) را محاسبه کنیم . به مثال زیر توجه کنید:

مثال :

```
A=[7 8 9]
B=length(A)
```

نتیجه :

```
A =

     7     8     9


B =

     3
```

البته دستور length برای ماتریس ها نیز می تواند مورد استفاده قرار بگیرد . اگر دستور length را برای یک ماتریس به کار ببریم ، این دستور تعداد ردیف ها و تعداد ستون های ماتریس را محاسبه می کند و هر کدام از این دو عدد که بزرگتر باشد را در خروجی نمایش خواهد داد . به مثال زیر توجه کنید :

مثال :

```
A=[1 2 3;4 5 6]
B=length(A)
```

نتیجه :

```
A =

     1     2     3
     4     5     6


B =

     3
```

اجرای دستورات شرطی با دستور if در متلب:

از دستور if در متلب برای اجرای دستورات شرطی استفاده می شود . یعنی اینکه در ابتدا شرط یا شرط هایی توسط متلب چک می شود و اگر آن شرط یا شرط ها برآورده شده باشد آنگاه متلب دستورات مشخص شده را اجرا خواهد کرد . به مثال زیر توجه کنید:

مثال :

```
A=5
if A>=0
    B=A
end
if A<=0
    B=-A
end
```

نتیجه :

```
A =

     5


B =

     5
```

همان طور که مشاهده می کنید از دو دستور if استفاده کرده ایم . هدف این است که مقدار B برابر قدرمطلق A باشد ، بنابراین اگر A مساوی یا بزرگتر از صفر باشد باید B را برابر A قرار دهیم و اگر A مساوی یا کوچکتر از صفر باشد باید B را برابر –A قرار دهیم.
دقت کنید که در پایان دستور if حتما باید end نوشته شود تا نرم افزار متلب بداند که دستور if پایان یافته است.

دستور if به همراه else :

همان طور که گفتیم زمانی که از دستور if در متلب استفاده می کنیم ، متلب شرط یا شرط هایی را چک می کند و در صورت برآورده شدن آنها ، دستورات را اجرا می کند . اما شاید بخواهیم به متلب اعلام کنیم که اگر شرط یا شرط ها برآورده نشدند آنگاه چه دستوراتی را اجرا کند . در اینگونه موارد دستور if را با else به کار می بریم . به مثال زیر توجه کنید:

مثال :

در مثال قبلی از دو دستور if استفاده کردیم اما این بار همان مثال را تنها با یک دستور if می نویسیم:

```
A=5
if A>=0
    B=A
else
    B=-A
end
```

نتیجه :

```
A =

     5


B =

     5
```

هدف این بوده است که B برابر قدرمطلق A باشد ، ابتدا متلب چک می کند که A مساوی یا بزرگتر از صفر هست یا نه ، اگر باشد آنگاه B را برابر A قرار می دهد و چون شرط برآورده شده است دستورات نوشته شده برای else را نادیده می گیرد . اما اگر A مساوی یا بزرگتر از صفر نباشد آنگاه متلب تنها دستورات مربوط به else را اجرا می کند.

دستور if به همراه elseif :

گاهی نیاز داریم که چندین شرط به صورت پی در پی چک شوند ، اگر اولین شرط صحیح بود دستورات مربوط به ان اجرا شوند و دستورات مربوط به سایر شرط ها نادیده گرفته شوند ، اما اگر شرط اول برآورده نشده بود شرط دوم چک شود و در صورت برآورده شدن شرط دوم ، دستورات مربوط به آن اجرا شود و دستورات مربوط به شرط های باقیمانده نادیده گرفته شود ، در صورت برآورده نشدن شرط دوم آنگاه شرط سوم چک شود و همین طور تا آخر . در اینگونه موارد باید از دستور if به همراه elseif استفاده کنیم . به مثال زیر توجه کنید:

مثال :

همان مثال قبل را این بار با استفاده از elseif می نویسیم . تنها تفاوت این است که حالت خاص A=0را جداگانه بررسی کرده ایم:

```
A=5
if A>0
    B=A
elseif A==0
    B=0
else
    B=-A
end
```

نتیجه :

```
A =

     5


B =

     5
```

دقت شود که برای چک کردن شرط تساوی حتما باید از دو علامت تساوی به صورت == استفاده شود ، زیرا علامت = در متلب برای نسبت دادن مقدار به متغیرها در نظر گرفته شده است و بنابراین برای چک کردن شرط تساوی مجبوریم از علامت == استفاده کنیم.

دستور while:

در این دستور دقیقا مانند if شرطی وارد میشود ولی نحوه کنترل برنامه بدین صورت است که تا زمانی که شرط صادق باشد دستورات داخل عبارت while تکرار خواهند شد.

مثال :

همان مثال قبل را این بار با استفاده از elseif می نویسیم . تنها تفاوت این است که حالت خاص A=0را جداگانه بررسی کرده ایم:

```
clc
close all
clear all
x=.5
while (sin(x)<.8)
    x=x+.1;
end
x
sin (x)
```

نتیجه :

```
x =

    1.0000


ans =

    0.8415
```

ترسیم گرافیکی توابع در متلب

دستور ezplot:

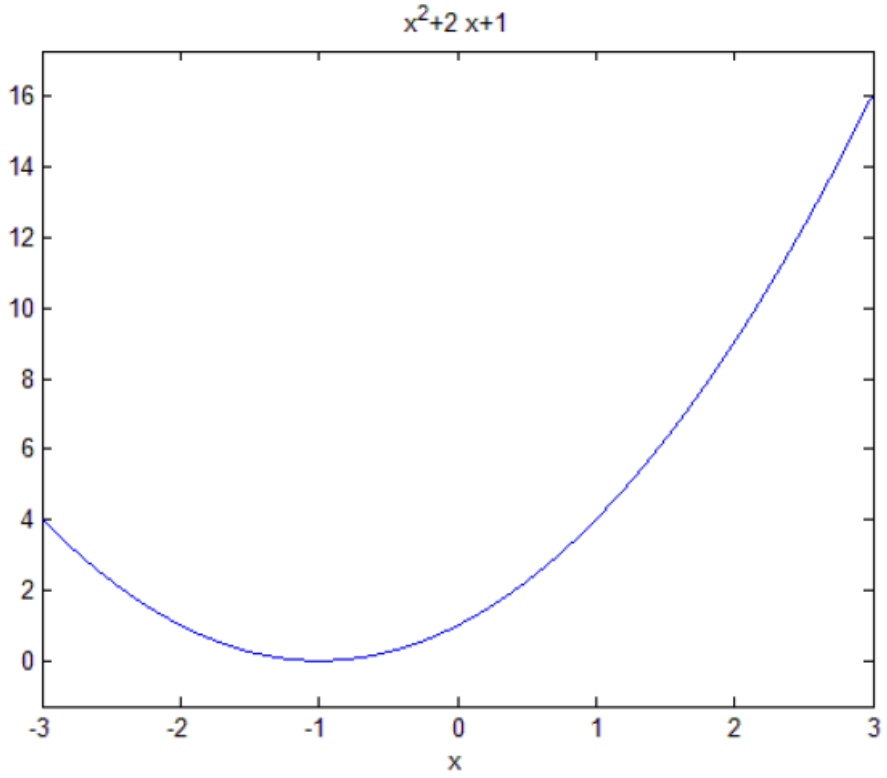
در متلب با استفاده از دستور ezplot می‌توانیم توابع را به صورت گرافیکی رسم کنیم. تنها کافیست که ابتدا عبارت تابع و سپس محدوده ای که میخواهیم تابع در آن محدوده رسم شود را مشخص کنیم. به مثال زیر توجه کنید:

مثال :

```
ezplot('x^2+2*x+1',[-3,3])
```

نتیجه :

متلب یک پنجره جدید را باز کرده و نتیجه را به صورت یک شکل نمایش می دهد:



عبارت [-3,3] ، محدوده ای که می خواهیم تابع رسم شود را مشخص کرده است.

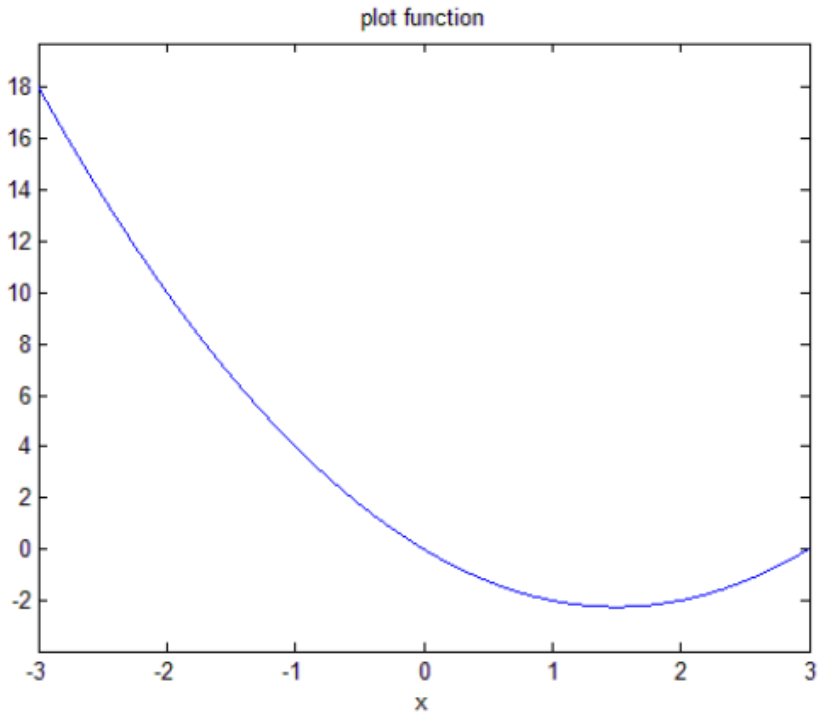
مشخص کردن عنوان برای شکل خروجی:

چنانچه بخواهیم که شکل ترسیم شده توسط دستور ezplot و یا هر روش دیگر) دارای عنوان خاصی باشد ، باید در خط بعدی پس از دستور ezplot از دستور title استفاده کنیم . این دستور یک رشته (مشخص شده با علامت ') را دریافت کرده و به صورت عنوان در بالای شکل نمایش می دهد . به مثال زیر توجه کنید:

مثال :

```
ezplot('x^2-3*x',[-3,3])
title 'plot function'
```

نتیجه :



مشاهده می کنید که عبارت plot function در بالای شکل نمایش داده شده است.

برای بستن پنجره ای که شکل را نمایش می دهد میتوان از روش های زیر استفاده کرد:

- در پنجره command کلمه close را تایپ کرده و اینتر بزنید.
- بر روی دکمه close در بالای پنجره شکل کلیک کنید.
- از منوی فایل گزینه close را انتخاب کنید.

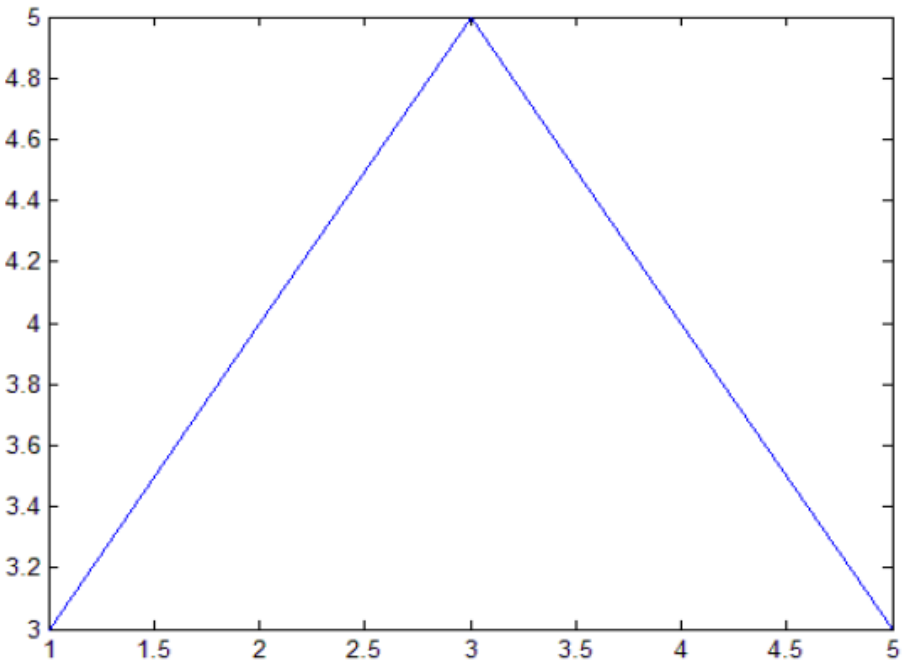
دستور plot :

دستور plot بردارهایی از اعداد را دریافت کرده و آنها را به صورت شکل ترسیم می کند . در واقع دستور plot مقادیر گسسته را که هر کدام به صورت یک نقطه می باشند پشت سرهم قرار می دهد و سپس آنها را با خط به هم وصل می کند تا بتوانیم آنها را به صورت یک شکل پیوسته ببینیم و بدین ترتیب به ارتباط کلی آنها پی ببریم . به مثال زیر توجه کنید:

مثال :

```
x=[1 2 3 4 5]
y=[3 4 5 4 3]
plot(x,y)
```

نتیجه :



دستور xlabel و دستور ylabel :

همان طور که می دانید در نرم افزار متلب ، دستورات مختلفی همچون plot و ezplot برای ترسیم شکل و منحنی ها به کار می روند . این دستورات به محورهای افقی و عمودی یا عنوان اختصاص نمی دهند و یا اگر عنوان اختصاص بدهند ممکن است آن عنوان مد نظر ما نبوده باشد . ممکن است بخواهیم در شکل ترسیم شده توسط این دستورات ، محورهای افقی و عمودی دارای عنوان خاصی باشند . برای این منظور در متلب از دو دستور xlabel و ylabel استفاده می شود.

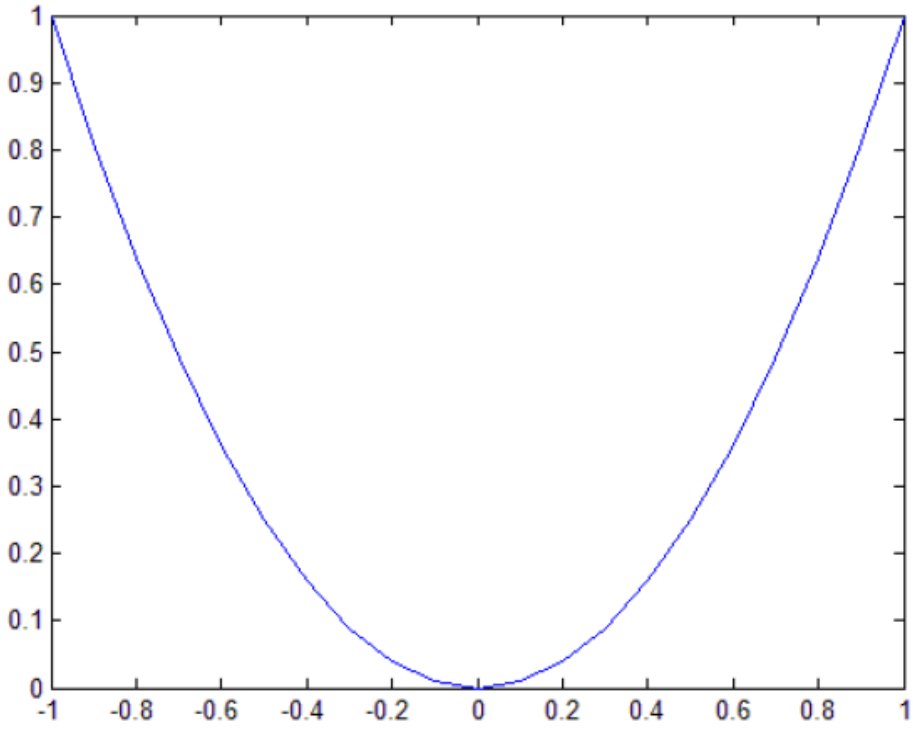
از دستور xlabel در متلب برای تعیین یک عنوان برای محور افقی شکل استفاده می شود و همچنین دستور ylabel نیز برای تعیین یک عنوان برای محور عمودی شکل به کار می رود . برای آشنایی با نحوه استفاده از دو دستور xlabel و ylabel به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم تابع $y = x^2$ را در بازه [-1,1] با دستور plot رسم کنیم . می نویسیم:

```
x=-1:0.1:1;
plot(x,x.^2)
```

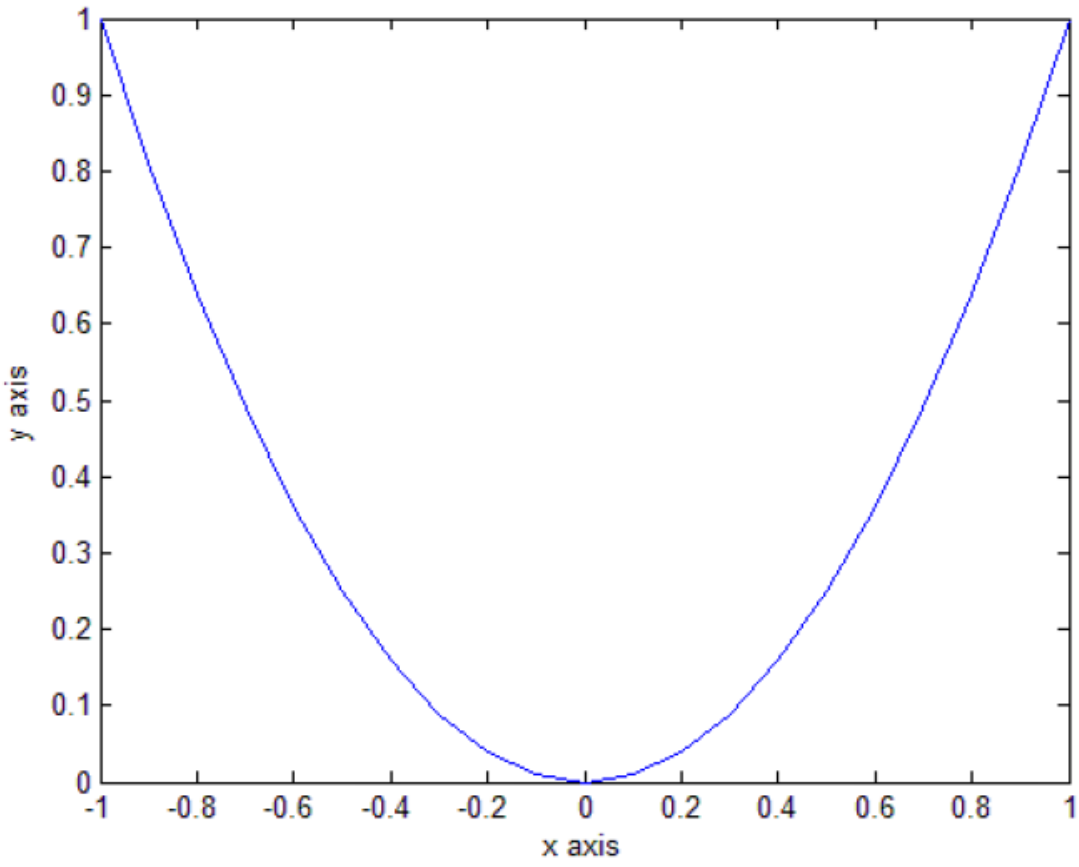
نتیجه :



مشاهده می کنید که شکل خروجی دارای عنوان برای محور افقی و عنوان برای محور عمودی نمی باشد . حال این بار با دستور xlabel و دستور ylabel برای هر دو محور شکل ، عنوان تعیین می کنیم

```
x=-1:0.1:1;
plot(x,x.^2)
xlabel('x axis')
ylabel('y axis')
```

نتیجه :



مشاهده می کنید که دستور xlabel('x axis') تعیین کرده است که محور افقی دارای عنوان x axis باشد و دستور ylabel('y axis') نیز تعیین کرده است که محور عمودی دارای عنوان y axis باشد.

تعیین رنگ خطوط منحنی های رسم شده با دستور plot در متلب:

در متلب با استفاده از دستور plot می توانیم منحنی های مختلف را رسم کنیم . اما دستور plot ، خطوط منحنی را با یک رنگ پیش فرض نمایش می دهند . چنانچه بخواهیم خطوط منحنی با رنگی دیگر نمایش داده بشوند باید عبارت مربوط به آن رنگ را درون پرانتز این دستورات بنویسیم . برای تعیین رنگ باید در میان دو علامت ' یک حرف انگلیسی را که نشان دهنده آن رنگ می باشد بنویسیم.

در متلب برای هر رنگ یک حرف انگلیسی در نظر گرفته شده است . لیست این حروف در جدول زیر نمایش داده شده است:

رنگ	حرف متناظر برای آن رنگ
Red	r
Green	g
Blue	b
Cyan	c
Magenta	m
Yellow	y
Black	k
White	w

شیوه های نمایش خطوط منحنی (Line Style Specifiers) :

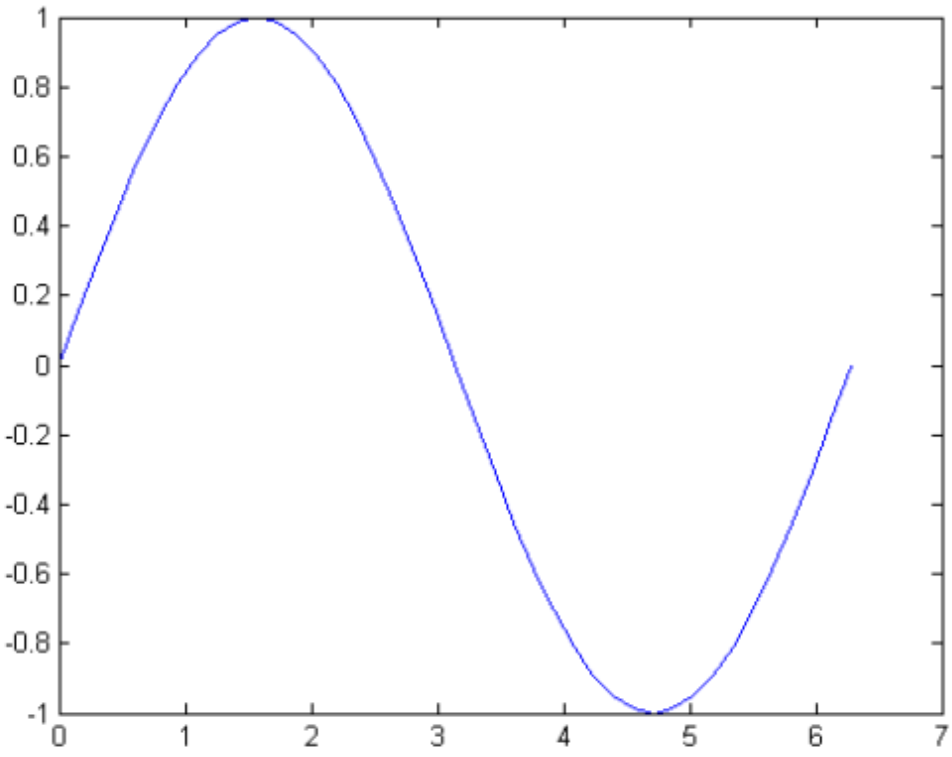
شیوه های نمایش خطوط منحنی (Line Style Specifiers) در جدول زیر خلاصه شده است:

Specifier	Line Style
'-'	Solid line (default)
'--'	Dashed line
':'	Dotted line
'-.'	Dash-dot line
'none'	No line

مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'-')
```

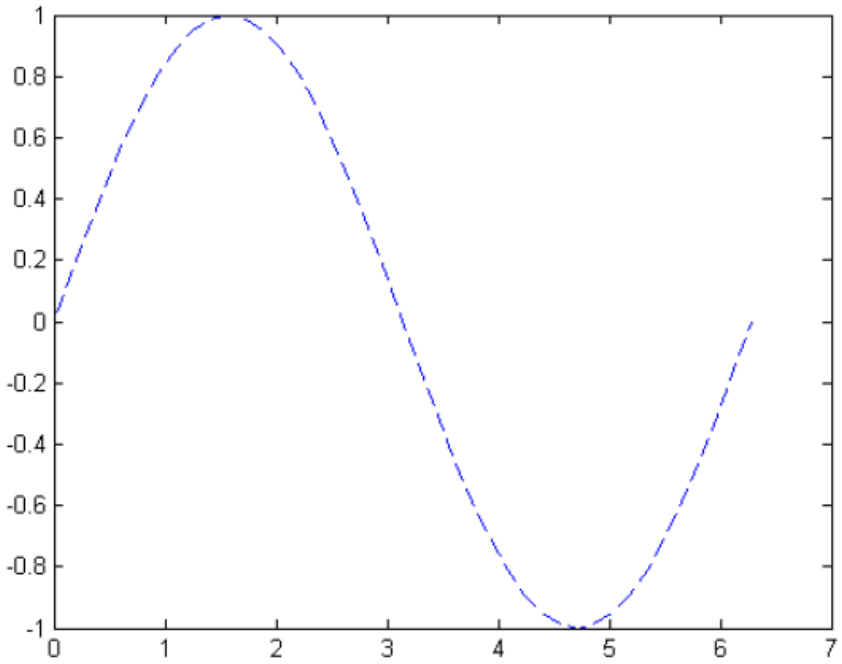
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'--')
```

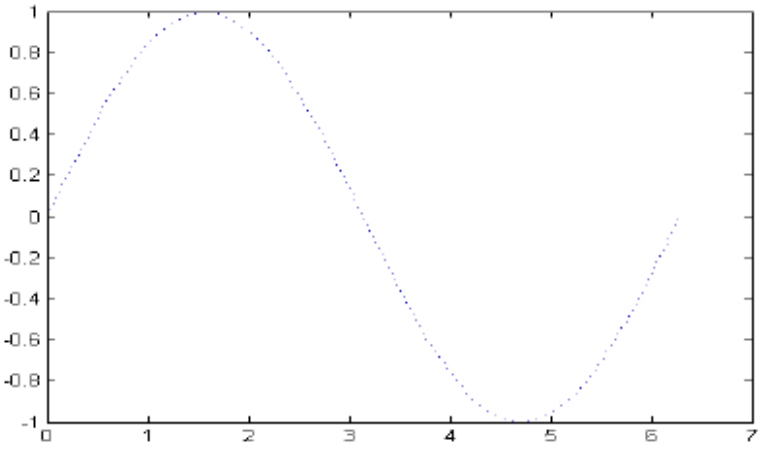
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),':')
```

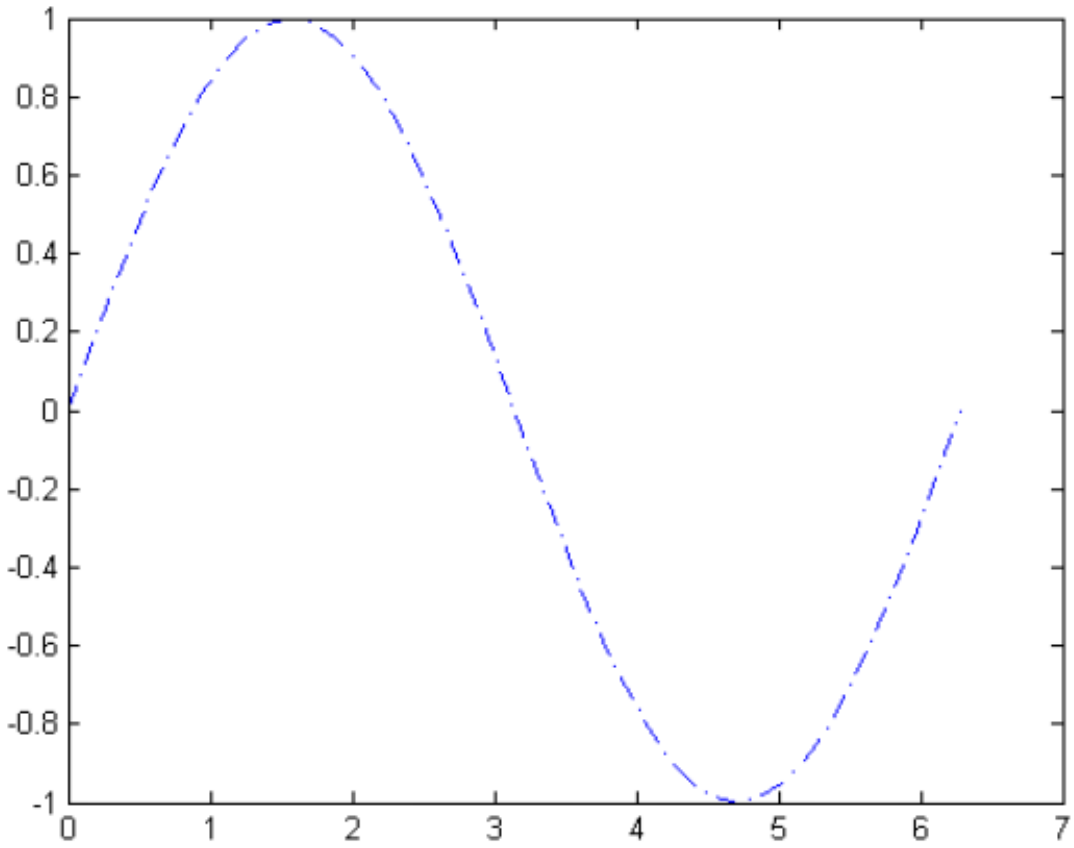
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'-.')
```

نتیجه :



شیوه های نمایش نقاط (Marker Specifiers) :

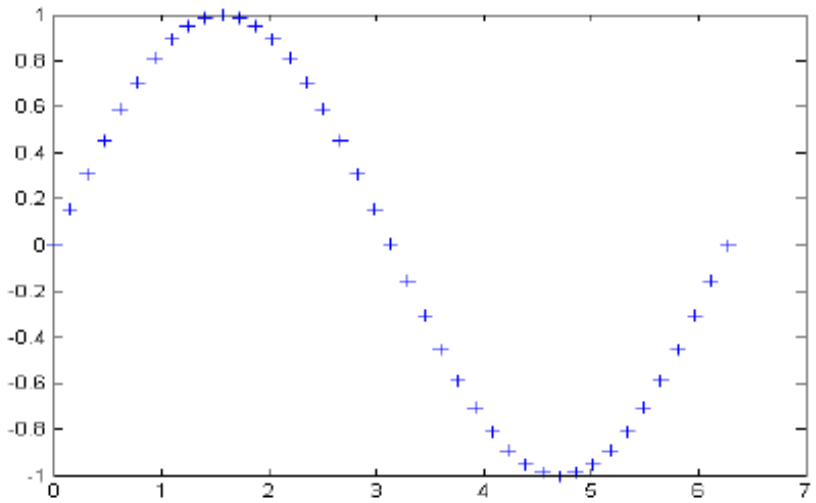
شیوه های نمایش نقاط (Marker Specifiers) در جدول زیر خلاصه شده است:

Specifier	Marker Type
'+'	Plus sign
'o'	Circle
'*'	Asterisk
'.'	Point
'x'	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond
'^'	Upward-pointing triangle
'v'	Downward-pointing triangle
'>'	Right-pointing triangle
'<'	Left-pointing triangle
'pentagram' or 'p'	Five-pointed star (pentagram)
'hexagram' or 'h' '	Six-pointed star (hexagram)
'none'	No marker (default)

مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'+')
```

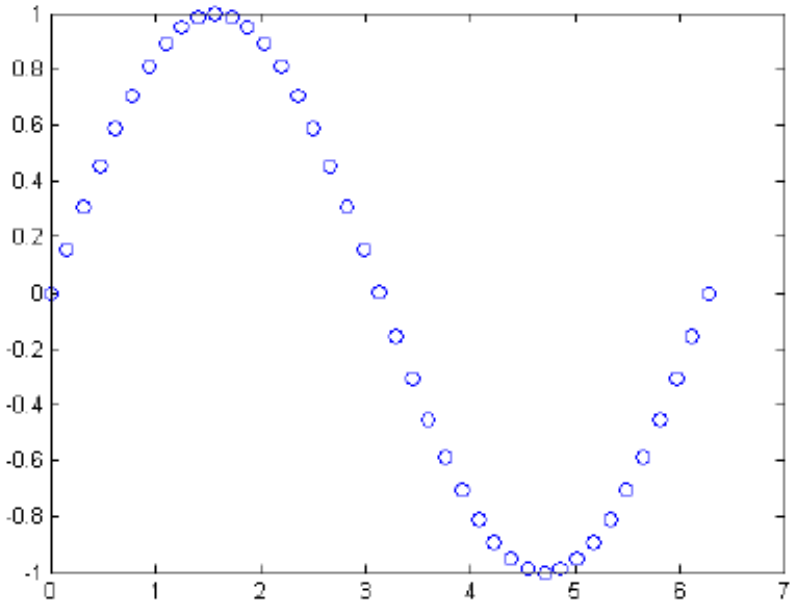
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'o')
```

نتیجه :

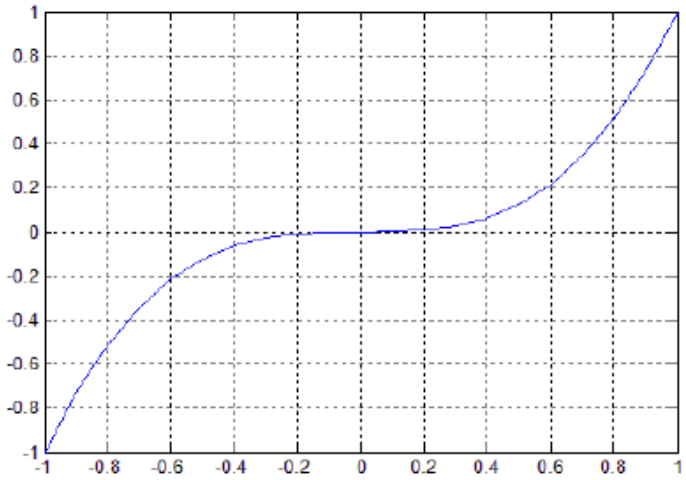


نمایش پس زمینه شکل به صورت چهارخانه با دستور grid در متلب:

با دستور grid در متلب می توانیم پس زمینه یک شکل را به صورت چهارخانه درآوریم . مثلا فرض کنید با دستور plot یک منحنی را رسم کرده باشیم . چنانچه با دستور grid ، پس زمینه منحنی را به صورت چهارخانه درآوریم ، در این صورت راحت تر می توانیم مقادیر متناظر با هر نقطه از منحنی را تشخیص بدهیم . به مثال زیر توجه کنید:

```
x=-1:0.1:1;
plot(x,x.^3)
grid
```

نتیجه :



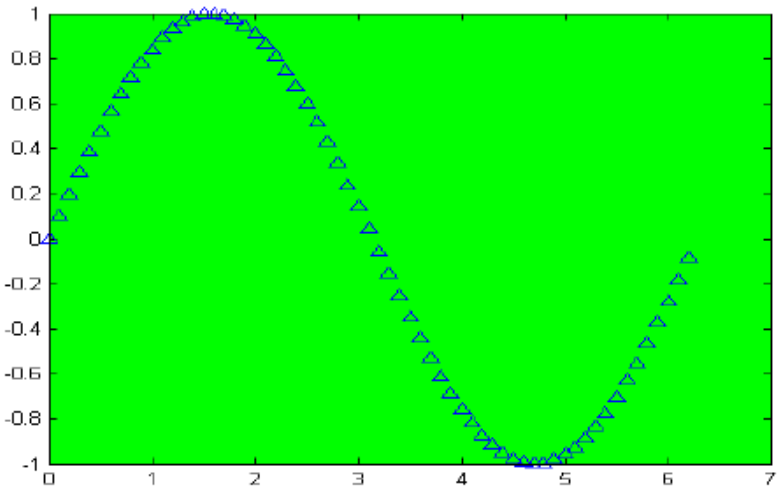
تعیین رنگ پس زمینه شکل ها با دستور whitebg در متلب:

همان طور که می دانید ، نرم افزار متلب به طور خودکار رنگ پس زمینه شکل ها را سفید در نظر می گیرد اما ممکن است در مواردی بخواهیم رنگ پس زمینه شکل ، رنگ دیگری باشد . در این موارد می توانیم از دستور whitebg در متلب استفاده کنیم . باید دقت کنید که اگر رنگ پس زمینه شکل ها را تغییر دادید و دوباره خواستید به حالت اولیه ، یعنی رنگ سفید ، برگردد باید مجددا از دستور whitebgاستفاده کرده و این بار رنگ سفید را انتخاب کنید . به مثال زیر توجه کنید:

```
clear all
close all
clc

t=0:0.1:2*pi;
x=sin(t);
plot(t,x,'^')
whitebg('green')
```

مشاهده می کنید که باید رنگ مورد نظرمان را درون پرانتز دستور whitebg بنویسیم.

نتیجه :



ساخت یک m-file در متلب:

برای ساخت یک m-file جدید می توانید از هر یک از روش های زیر استفاده کنید :

-1در بالای پنجره اصلی نرم افزار متلب بر روی گزینه New script کلیک کنید . این گزینه به شکل  می باشد .

-2با نگه داشتن کلید Ctrl و فشار دادن کلید N از کیبورد ، این کار را انجام دهید .

-3در پنجره Command بنویسید edit و سپس کلید enter از کیبورد را فشار بدهید .

هر یک از روش های بالا را که انتخاب کنید ، نتیجه این است که متلب یک پنجره خالی باز می کند که می توانید در آن دستورات خود را اجرا کنید .

توصیه می شود اولین دستوری که در یک m-file می نویسید ، دستور clear all باشد تا تمامی متغیرهایی که قبلا در متلب تعریف شده است را پاک کند و اختلالی در روند اجرای برنامه ایجاد نشود .

باید دقت داشته باشید که در نرم افزار متلب ، m-fileها برای دو هدف اصلی به کار می روند ، کاربرد اول آن نوشتن برنامه های پیچیده وطولانی و کاربرد دوم آن ساخت تابع می باشد .

پس از آنکه دستورات برنامه را در m-file نوشتیم ابتدا باید با استفاده از گزینه Save در بالای همان پنجره m-file ، آن را ذخیره کنیم . همچنین با نگه داشتن کلید Ctrl و فشار دادن کلید S ، می توانید این کار را انجام دهید .

سپس برای اجرای برنامه باید بر روی گزینه Save and run که به شکل  می باشد کلیک کنید تا نتایج برنامه در پنجره Command نمایش داده شود . همانطور که از نام این گزینه مشخص است ، این گزینه عمل ذخیره کردن را هم انجام می دهد یعنی اگر تغییراتی در برنامه ایجاد کنید و سپس بر روی این گزینه کلیک کنید ، این تغییرات در m-file ذخیره می شود . اگر قبلا فایل ذخیره نشده باشد ابتدا از شما می خواهد که نامی برای آن انتخاب کرده و سپس آن را ذخیره کنید .

m-fileها دارای پسوند m می باشند (به عنوان مثال. (program.m :

نوشتن توضیحات در m-file :

زمانی که یک برنامه طولانی بنویسید ، به دلیل حجم زیاد دستورات ، ممکن است بخشی از روند برنامه نویسی را فراموش کنید . گذشت زمان نیز بسیار تاثیرگذار است و گاهی آن قدر از زمان نوشتن برنامه گذشته است که خود برنامه نویس مجبور می شود برنامه را بارها بخواند تا درک کند که از چه روش هایی استفاده کرده است و گاهی نوشتن یک برنامه جدید به صرفه تر است و زمان کمتری نیاز دارد . بر حسب تجربه ثابت شده است که با استفاده از 2 تکنیک زیر می توان این مشکل را تا حد زیادی برطرف کرد .

-1انتخاب هوشمندانه نام متغیرها به گونه ای که هدف استفاده از آنها را بتوان از نامشان به طور کامل درک کرد .

-2می توانیم هنگام نوشتن برنامه ، توضیحاتی را در کنار کدها بنویسیم تا با خواندن آنها خود برنامه نویس یا هر شخص دیگری به راحتی درک کند که روش های استفاده شده در برنامه چیست.

در متلب چنانچه از علامت درصد ((%)) استفاده کنیم ، تمامی نوشته های بعد از علامت درصد به صورت توضیح در نظر گرفته می شوند . به مثال زیر توجه کنید:

مثال :

```
x=2
% ali eskandari
y=3
```

پردازش تصویر

پردازش تصویر (Image Processing) به مجموعه‌ای از تکنیک‌هایی اطلاق می‌شود که با هدف تبدیل (Convert) یک تصویر به قالب دیجیتال (Form) و انجام اعمال محاسباتی بر روی آن شکل گرفته‌اند. هدف از انجام اعمال محاسباتی مرتبط با پردازش تصویر در متلب، تولید نسخه‌ای بهبود یافته (Enhanced) از تصاویر دیجیتالی و یا استخراج اطلاعات با معنی و مفید از آن‌ها است. تغییراتی که بر اثر پردازش تصویر دیجیتالی، روی تصاویر اتفاق می‌افتند، معمولا به طور خودکار و بر پایه مجموعه‌ای از الگوریتم‌های به دقت طراحی شده انجام می‌شوند. این دسته از الگوریتم‌های پردازش تصویر در متلب، از لحاظ محاسباتی بسیار دقیق (Accurate) و بهینه (Optimized)هستند.

حوزه پردازش تصویری، یک حوزه چند رشته‌ا (Multidisciplinary) در شاخه علوم کامپیوتر (Computer Science)است که بخشی از مفاهیم خود را از رشته‌های علمی دیگر نظیر ریاضیات (Mathematics)، فیزیک (Physics) و مهندسی برق (Electrical Engineering) به اشتراک گرفته است. همچنین، حوزه پردازش تصویر، هم‌پوشانی بسیار زیادی با حوزه‌های تحقیقاتی نظیر بازشناسی الگو(Pattern Recognition) ، یادگیری ماشین (Machine Learning)، هوش مصنوعی (Artificial Intelligence) و بینایی کامپیوتر (Computer Vision)دارد.

برای انجام عملیات محاسباتی متناظر با پردازش تصویر در متلب، ابتدا باید تصاویر دیجیتال از طریق واسط‌هایی (Interfaces) نظیر اسکنر نوری (Optical Scanner) و دوربین‌های دیجیتال (Digital Cameras)تولید شوند. سپس، تصاویر دیجیتال تولید شده تحلیل (Analyze) می‌شوند. در مرحله بعد، تصاویر دیجیتالی از طریق فرآیندهایی نظیر فشرده‌سازی داده‌ها (Data Compression)، بهبود تصاویر(Image Enhancements) ، فیلتر تصاویر (Image Filtering)و سایر موارد، مورد دستکاری عددی (Numerical Manipulation) قرار گرفته و در نهایت، تصاویر خروجی مطلوب تولید می‌شوند.

قوه محرکه (Driving Force) توسعه تکنیک‌های پردازش تصویر در متلب، نیاز محققان این حوزه به استخراج اطلاعات (Information Extraction) مفید و بامعنی از تصاویر دیجیتالی و تفسیر آن‌ها است. امروزه، از تکنیک‌های توسعه داده شده در حوزه پردازش تصویر، در شاخه‌های مختلفی نظیر پزشکی(Medicine) ، صنعت(Industry) ، نظامی(Military) ، دستگاه‌های الکترونیکی کاربردی (Consumer Electronics) و سایر موارد استفاده می‌شود.

در پزشکی، از تکنیک‌های پردازش تصویر در دستگاه‌ها و روش‌های تصویربرداری تشخیص پزشکی (Diagnostic Imaging Modalities)نظیر برش‌نگاری با انتشار پوزیترون (Positron Emission Tomography | PET)، برش‌نگاری محوری کامپیوتر (Computerised Axial Tomography | CAT)، تصویربرداری رزونانس مغناطیسی (Magnetic Resonance Imaging | MRI)و تصویربرداری رزونانس مغناطیسی کارکردی (functional Magnetic

Resonance Imaging | fMRI)استفاده می‌شود. بسیاری از تکنیک‌های تشخیص پزشکی، توسط الگوریتم‌های پردازش تصویر در متلب پیاده‌سازی می‌شوند.

الگوریتم‌های پردازش تصویر در متلب، مجموعه‌ای از توابع هستند که قابلیت‌های محیط محاسبات عددی متلب را گسترش می‌دهند. تولباکس پردازش تصویر در متلب، مجموعه‌ای از الگوریتم‌های مرجع استاندارد (Reference-Standard Algorithms) را برای کاربردهای پردازش، تحلیل و نمایش بصری تصاویر و همچنین توسعه الگوریتم‌های پردازش تصویر در متلب فراهم می‌آورد.

از الگوریتم‌های پردازش تصویر در متلب، می‌توان برای بخش‌بندی تصاویر (Image Segmentation)، بهبود تصاویر، کاهش نویز (Noise Reduction) در تصاویر، تبدیلات هندسی (Geometric Transformation) ، انطباق تصویر (Image Registration)و انجام عملیات پردازش تصویر ۳-بعدی (3D Image Processing) استفاده کرد.

خواندن تصاویر در متلب

برای خواندن تصاویر در محیط متلب، از تابع (imread) استفاده می‌شود. قالب دستوری (Syntax)این تابع به شکل زیر است:

```
imread('filename');
```

در این تابع، آرگومان (filename) رشته‌ای است که نام کامل تصویر و فرمت (Extension) آن را نمایش می‌دهد. به عنوان نمونه:

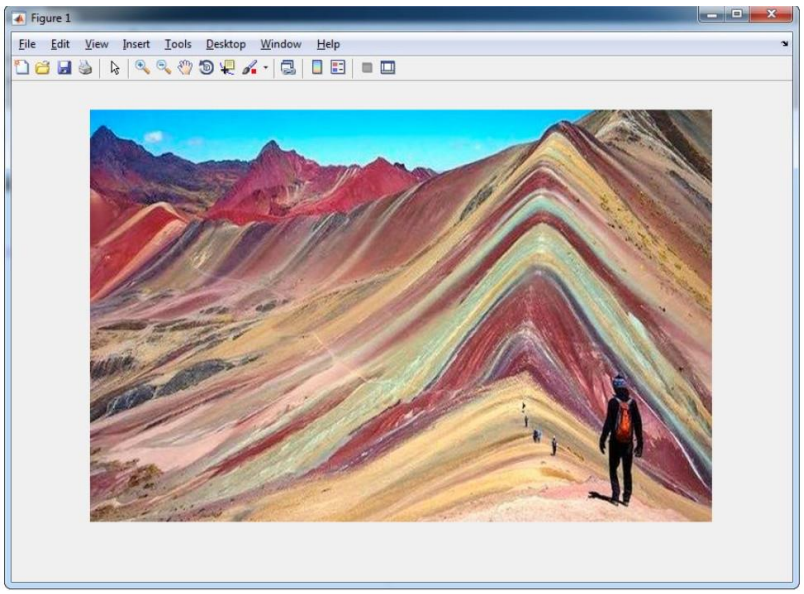
```
F = imread(Penguins_grey.jpg);
```

```
G = imread(Penguins_RGB.jpg);
```

شایان توجه است زمانی که آدرس فیزیکی محل ذخیره‌سازی تصویر، در آرگومان (filename) قید نشده باشد، تابع (imread) تصویر را از دایرکتوری (Directory) کنونی متلب خواهد خواند. در شرایطی که نیاز باشد تا تصویر از دایرکتوری دیگری در متلب خوانده شود، لازم است تا آدرس کامل تصویر در آرگومان (filename) مشخص شود.

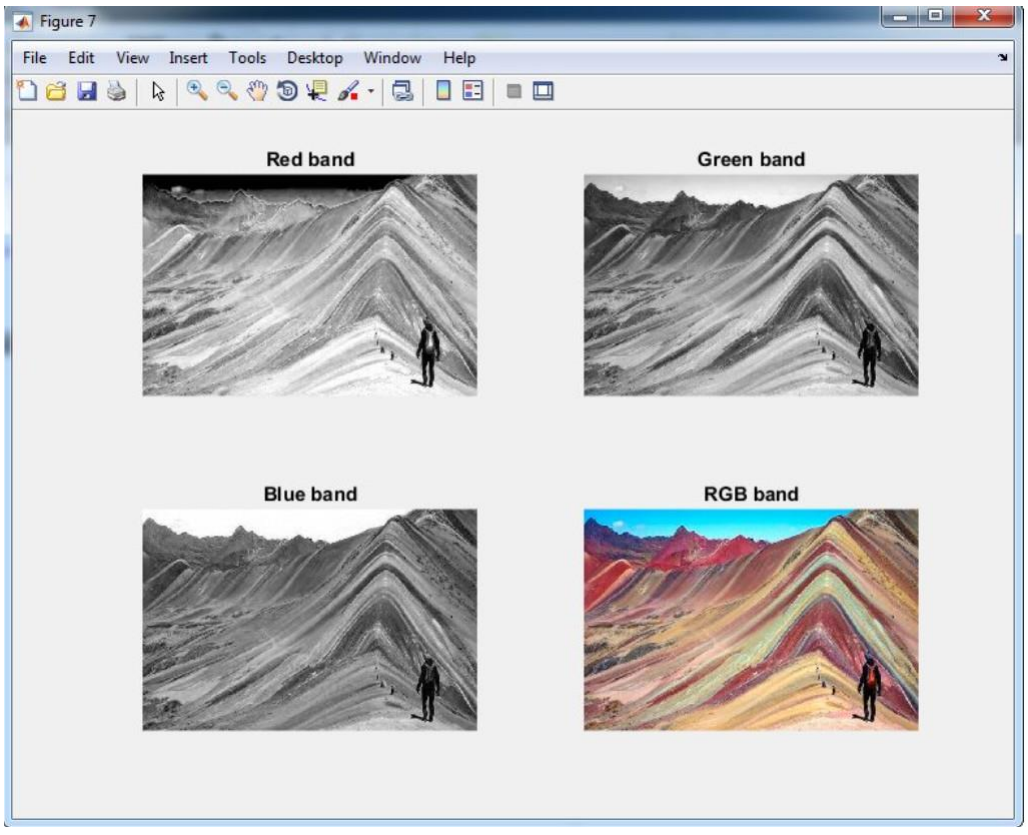
برای نمایش اولیه تصاویر در محیط MATLAB به راحتی می توانید از فرمان imshow استفاده کنید. فرمان های ذیل را به کار برید و نتایج آن را مشاهده نمایید.

```
figure
imshow('Geology1.jpg')%
figure
imshow('biologicalpestcontrol.jpg')
```



این تصویر به صورت رنگی است. یعنی دارای باندهای قرمز، سبز و آبی هستند. می توان کل ماتریس تصویر را با استفاده از فرمان imread به محیط MATLAB فراخوانی نمود. در فرمان های ذیل، تصویر فوق را خوانده و باندهای مختلف آن نمایش داده می شود.

```
Geo = imread('Geology1.jpg');
figure
subplot(2,2,1)
imshow(Geo(:,:,1))
title('Red band')
subplot(2,2,2)
imshow(Geo(:,:,2))
title('Green band')
subplot(2,2,3)
imshow(Geo(:,:,3))
title('Blue band')
subplot(2,2,4)
imshow(Geo)
title('RGB band')
```



دستور imfinfo در متلب

با استفاده از دستور imfinfo در متلب مجموعه‌ای از اطلاعات و ویژگی‌های تصویر در اختیار شما قرار می‌گیرد. این اطلاعات شامل اندازه تصویر (طول و عرض تصویر)، حجم تصویر، نحوه کدینگ تصویر، مسیر تصویر در هارد و غیره هست. آدرس تصویر مدنظرتان را به‌عنوان ورودی به دستور imfinfoدر متلب بدهید. مانند کد زیر:

```
info = imfinfo('ngc6543a.jpg')
```

دستور hdrread در متلب

تصاویر hdr مخفف تصاویر high dynamic range هستند. این تصاویر شامل دستورات جداگانه‌ای برای فراخوانی در متلب هستند. دستور hdrread در متلب برای فراخوانی تصویر hdr در متلب استفاده می‌شود. تصاویرhdr ، تصاویر RGB هستند و از همه مهم‌تر بازه تغییر اعداد آنها [inf 0]هست. درحالی‌که در تصاویر معمولی بازه اعداد [۰ ۲۵۵] هست. این مساله نشان میدهد که ما با تصاویری سر و کار داریم که میزان رنگ بیشتری در آنها وجود داره. نحوه نمایش این تصاویر هم کمی متفاوت با نمایش تصاویر معمولی هست. پس از خواندن تصویر hdr مطابق کدهای زیر باید با استفاده از دستور tonemap تصویر hdr را برای نمایش رندر (render) کرد و سپس از دستور imshow برای نمایش تصویر استفاده نمود. یک نمونه مثال ببینید:

```
hdr = hdrread('office.hdr');

rgb = tonemap(hdr);

imshow(rgb)
```

دستور hdrwrite در متلب

از دستور hdrwrite در متلب برای ذخیره تصاویر hdr استفاده می‌شود. نحوه استفاده از این دستور دقیقا شبیه دستور imwrite هست و هیچ تفاوتی با هم ندارند. یعنی کافی است، نام تصویر موردنظر برای ذخیره و همچنین آدرس محل ذخیره را به دستور hdrwrite بدهید. مانند مثال زیر:

```
hdr = hdrread('office.hdr');


hdrwrite(hdr,'E:\howsam\hdr.png')
```

الگوریتم‌های تکاملی

الگوریتم‌های تکاملی از روش ها و عملیات ابتدایی برای حل مسئله استفاده می‌کنند و در طی یک سری از تکرارها به راه‌حل مناسب برای مسئله می‌رسند. این الگوریتم‌ها غالباً از یک جمعیت حاوی راه‌حل‌های تصادفی شروع می‌کنند و در طی هر مرحله تکرار سعی در بهتر کردن مجموعه راه‌حل‌ها دارند. در آغاز کار تعدادی از اعضای جامعه به‌صورت تصادفی حدس زده‌شده، سپس تابع هدف یا برازندگی برای هر یک از این اعضا محاسبه و نخستین نسل ایجاد خواهد شد. اگر هیچ‌یک از معیارهای خاتمه بهینه‌سازی دیده نشوند، ایجاد نسل جدید آغاز خواهد شد.

اعضا برحسب میزان شایستگی‌شان برای تولید فرزندها انتخاب می‌شوند. این افراد به‌عنوان والدین محسوب می‌شوند و بازترکیب فرزندان را تولید می‌نمایند. سپس تمامی فرزندها با یک مقدار معینی از احتمال، یعنی همان جهش، تغییر ژنتیکی می‌یابند. اکنون میزان شایستگی (برازندگی) فرزندان تعیین و در اجتماع جایگزین والدین شده و نسل جدید را ایجاد می‌نمایند. این چرخه آن‌قدر تکرار می‌شود تا یکی از معیارهای پایان بهینه‌سازی کسب شود.

برای به‌کارگیری یک روش مناسب بهینه‌سازی برای یک مسئله، ابتدا باید همه جوانب مسئله به‌طورکلی مطالعه و تحلیل شود. سپس با درک کامل موضوع، اقدام به انتخاب یک روش مناسب بهینه سازی شود. برای بهینه سازی، می توان از الگوریتم های تکاملی مختلف استفاده کرد که از آن جمله می توان به الگوریتم های **کلونی مورچه، ژنتیک، ازدحام ذرات، الگوریتم زنبور** و الگوریتم های پویا اشاره کرد. در این نوع الگوریتم ها یک جواب سراسری سریع برای حل مسئله پیدا می شود که هدف یافتن یک جواب به‌صورت سریع است. در ادامه به تشریح الگوریتم ژنتیک می‌پردازیم.

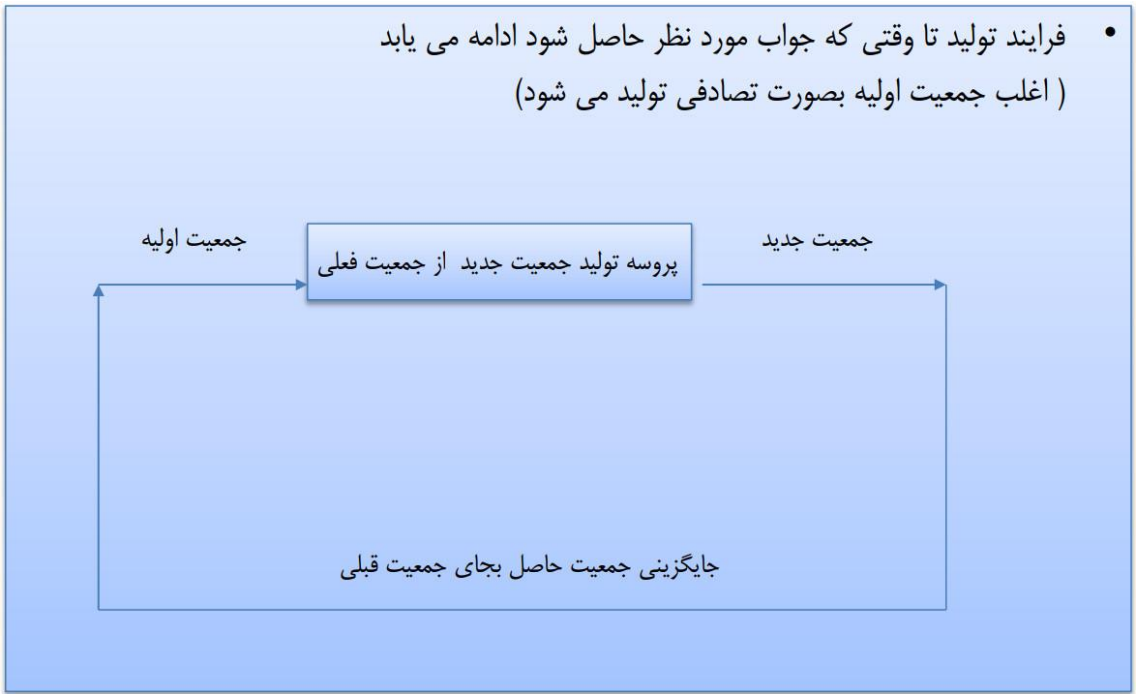
الگوریتم ژنتیک

الگوریتم ژنتیک (Genetic Algorithm - GA)-تکنیک جستجویی در علم رایانه برای یافتن راه‌حل تقریبی برای بهینه‌سازی و مسائل جستجو است. الگوریتم ژنتیک نوع خاصی از الگوریتمهای تکامل است که از تکنیکهای زیست‌شناسی مانند وراثت و جهش استفاده می‌کند. الگوریتم ژنتیک توسط جان هالند در سال ۱۹۶۷ ابداع شده‌است. کاربرد اصلی الگوریتم ژنتیک در کامپیوتر است، اما روشهایی از ژنتیک در مهندسی صنایع، برنامه‌ریزی تولید، مدیریت تولید، مدیریت فناوری اطلاعات و مدیریت صنعتی نیز قابل استفاده است.

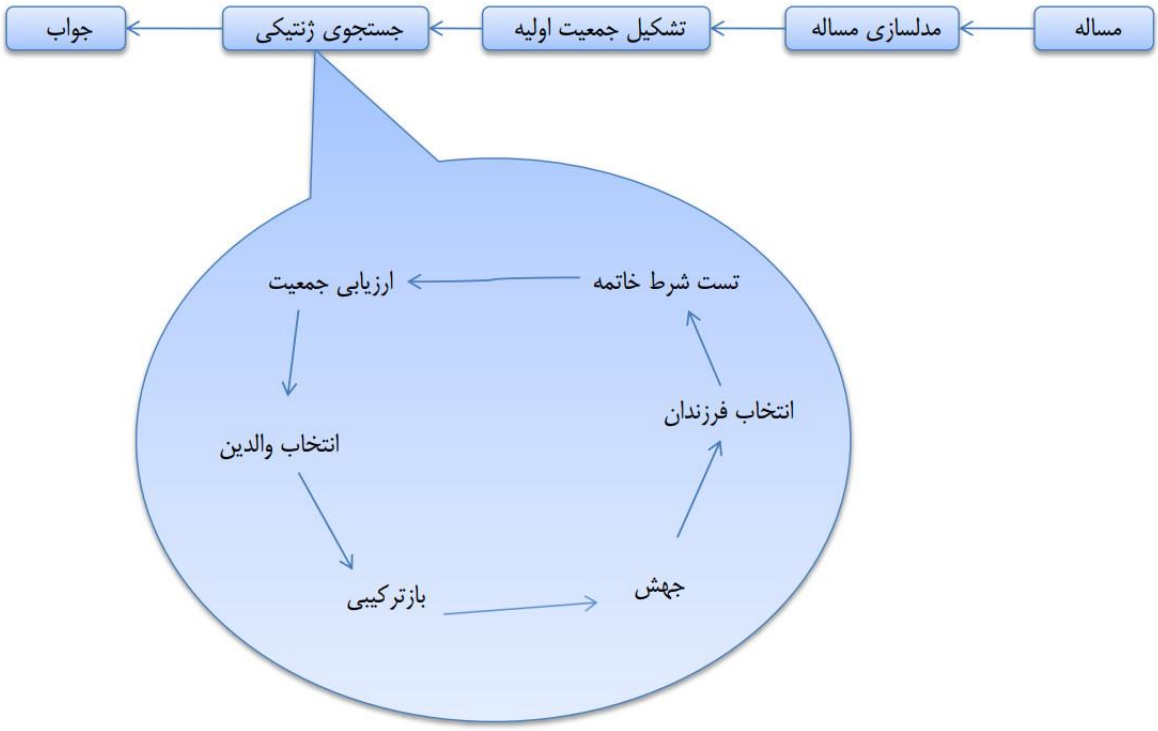
الگوریتم‌ژنتیک،الهامی‌ازعلم‌ژنتیک‌ونظریهٔ‌تکامل‌داروین‌است‌وبراساس‌بقای‌برترینهایاانتخاب طبیعی‌استواراست.یک‌کاربردمتداول‌الگوریتم‌ژنتیک،استفاده‌ازآن‌بعنوان‌تابع‌بهینه کننده‌است.الگوریتم‌ژنتیک‌ابزارسودمندی‌دربازشناسی‌الگو،انتخاب‌ویژگی،درک‌تصویرویادگیری‌ماشینی‌است.

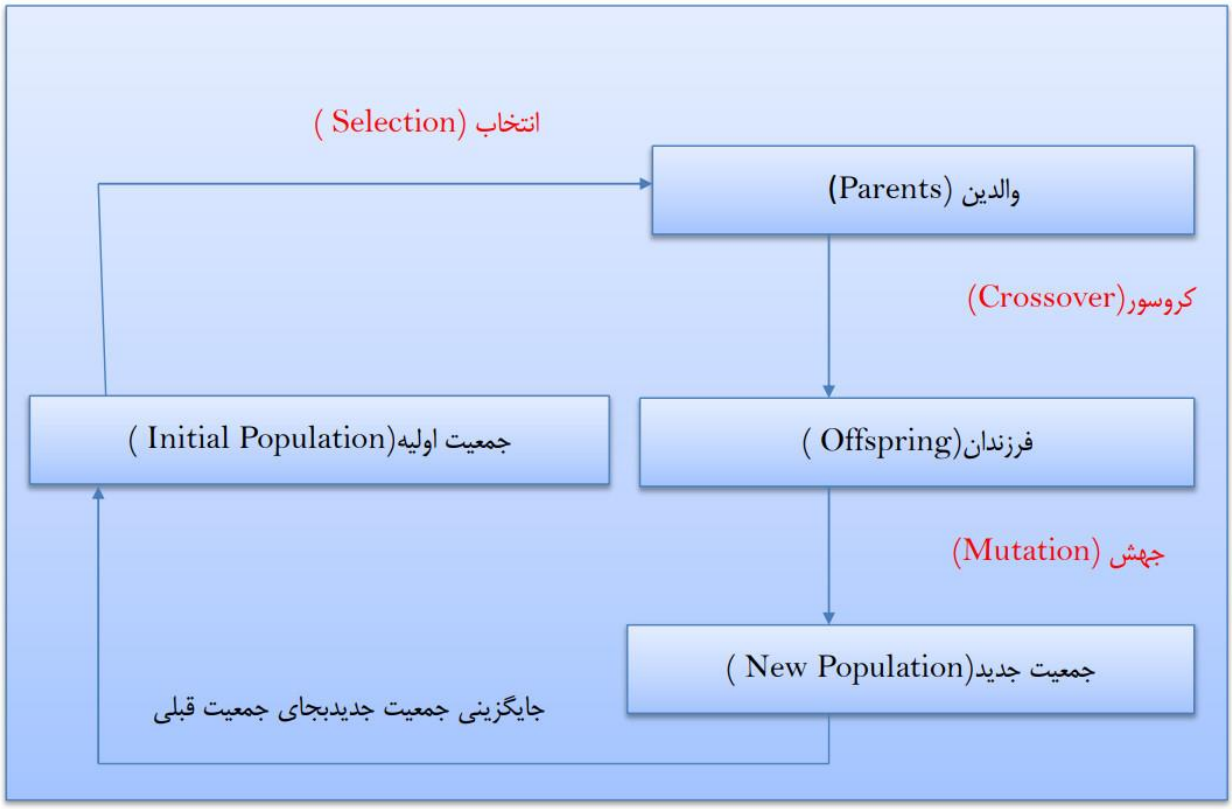
در علوم کامپیوتر و ریاضیات الگوریتم جستجو الگوریتمی است که یک مساله را به عنوان ورودی میگیرد و پس از ارزیابی راه حل های ممکن یک راه حل برای آن مساله بر میگرداند. هنگامی که یک مساله را حل میکنیم بدنبال راه حل بهینه و یا به عبارتی بهترین پاسخ از بین پاسخ های ممکن هستیم. یعنی مانند طبیعت یک جمعیت از موجودات را تشکیل می دهند و با اعمالی بر روی این مجموعه به یک مجموعه بهینه و یا موجود بهینه دست می یابند.

نمودار گردشی فرایند یک الگوریتم ژنتیک



ساختار الگوریتم ژنتیک





در الگوریتم ژنتیک با الهام از طبیعت، هدف حل یک مساله جستجو و رسیدن به پاسخ بهینه است. برای این کار ابتدا باید با دو پارامتر زیر آشنایی پیدا کنیم:

- **جمعیت : (Population)** مجموعه ای از پاسخ های ممکن
- **کروموزم :** در الگوریتم ژنتیک هر کروموزوم یک راه حل و پاسخ ممکن برای مساله است. در واقع مجموعه ای از کروموزوم ها جمعیت را تشکیل میدهند. برای نمایش کروموزوم ها معمولا از رشته های دودویی و یا اعداد حقیقی استفاده میکنند. هر کروموزوم از تعدادی ژن تشکیل میشود. برای مثال در یک رشته دودویی هر بیت معادل یک ژن از کروموزوم می باشد

حال باید از بین جمعیت، کروموزوم های خوب برای بقا و تکثیر باقی مانده و کروموزوم های نا مناسب از بین بروند. فقط به این نکته باید دقت کرد که میزان جمعیت ما ثابت باقی می ماند و گونه های نا مناسب حذف میشوند .مسائل پیش رو:

۱. طراحی ساختار کروموزوم
۲. ایجاد جمعیت اولیه
۳. روش انتخاب کروموزم ها
٤. نحوه مشخص کردن کیفیت کروموزوم ها(Fitness)
٥. معیار توقف الگوریتم

ساختار هر کروموزوم و جمعیت اولیه با توجه به شرایط اولیه مساله ایجاد میشوند و باید در کل الگوریتم شرایط مساله را حفظ کنند.

تابع برازش

ارزیابی جمعیت: (Fitness) برای اینکه بتوانیم موجودات بهتر را درون جمعیت تشخیص بدهیم بایستی معیاری را تعریف کنیم که بر اساس آن موجودات بهتر را تشخیص دهیم. به این کار، یعنی تعیین میزان خوبی یک موجود، ارزیابی آن موجود می گویند. ارزیابی، اینگونه است که بر حسب اینکه موجود چقدر خوب است یک عدد به آن نسبت می دهیم، این عدد که برای موجودات بهتر بزرگتر یا کوچکتر است را شایستگی آن موجود می نامیم.

به عنوان مثال در صورتی که به دنبال مینیمم یک تابع هستیم، مقدار شایستگی را می توانیم ورودیهایی که مقادیر تابع برای آنها کمتر است در نظر بگیریم که ورودیهای بهتری هستند.

بسته به نوع مساله ما می خواهیم شایستگی را بیشینه و یا کمینه کنیم.

عملگرهای الگوریتم ژنتیک

- انتخاب
- تقاطع
- جهش

۱- انتخاب (Selection)

- ✓ سوق دادن جستجو به بخشهایی از فضا که امکان یافتن جوابهای با کیفیت بالاتر وجود دارد.
- ✓ نسل جدیدی از راه حل ها را با انتخاب والدینی که بالاترین Fitness را دارند تولید می کند.

والدین : در هر نسل تعدادی از عناصر جمعیت این فرصت را پیدا می کنند که تولید مثل کنند. به این عناصر که از میان جمعیت انتخاب می شوند، والدین می گویند.

روشهای مختلفی برای انتخاب والدین وجود دارند. در زیر به چند مورد از این روشها اشاره می کنیم:

- ✓ **انتخاب تمام جمعیت بعنوان والدین** :در واقع هیچگونه انتخابی انجام نمی دهیم (همه عناصر انتخاب می شوند)
- ✓ **انتخاب تصادفی** :بصورت تصادفی تعدادی از موجودات جمعیت را بعنوان والدین انتخاب می کنیم، این انتخاب می تواند با جایگذاری یا بدون جایگذاری باشد. در این روشها عناصر با شایستگی بیشتر شانس بیشتری برای انتخاب شدن بعنوان والدین را دارند.
- ✓ **سایر روشها :**این روشها با استفاده از تکنیک هایی سعی می کنند که انتخاب هایی را ارائه دهند، که هم رسیدن به جواب نهایی را تسریع کنند و هم اینکه کمک می کنند که جواب بهینه تری پیدا شود.

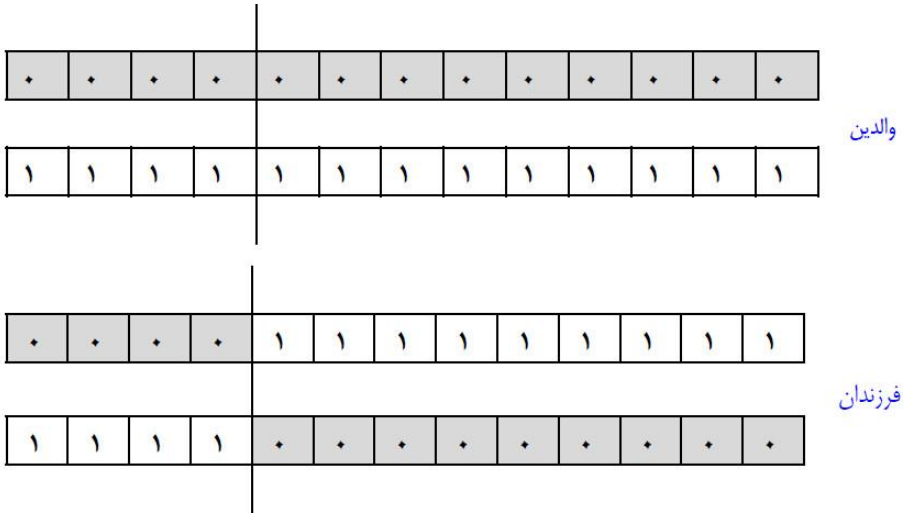
۲- تقاطع (یا ترکیب مجدد Recombination/Crossover)

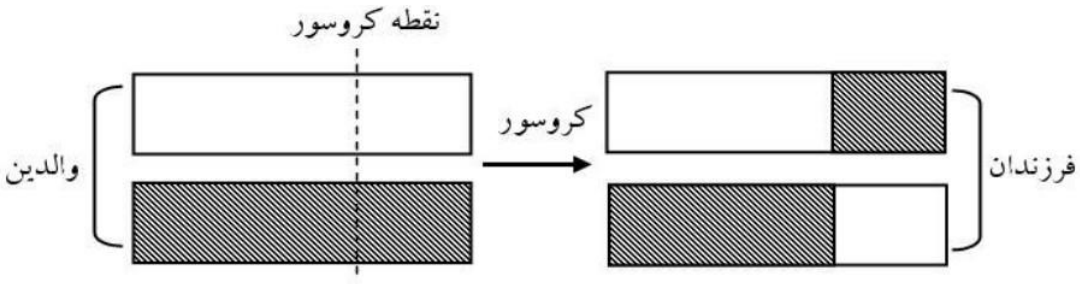
- ✓ امکان ترکیب جوابهای جزیی (partial solutions) یافت شده و در نتیجه بدست آوردن جوابهایی با کیفیت بالاتر را فراهم می آورد.

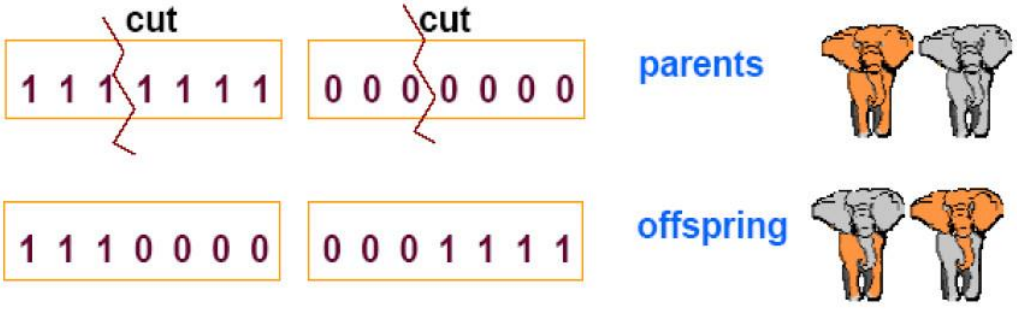
در جریان عمل بازترکیبی به صورت اتفاقی بخشهایی از کروموزوم ها با یکدیگر تعویض میشوند. این موضوع باعث می شود که فرزندان ترکیبی از خصوصیات والدین خود را به همراه داشته باشند و دقیقاً مشابه یکی از والدین نباشند.

هدف تولید فرزند جدید می باشد به این امید که خصوصیات خوب دو موجود در فرزندشان جمع شده و یک موجود بهتری را تولید کند.

نحوه انجام عملیات بازترکیبی:



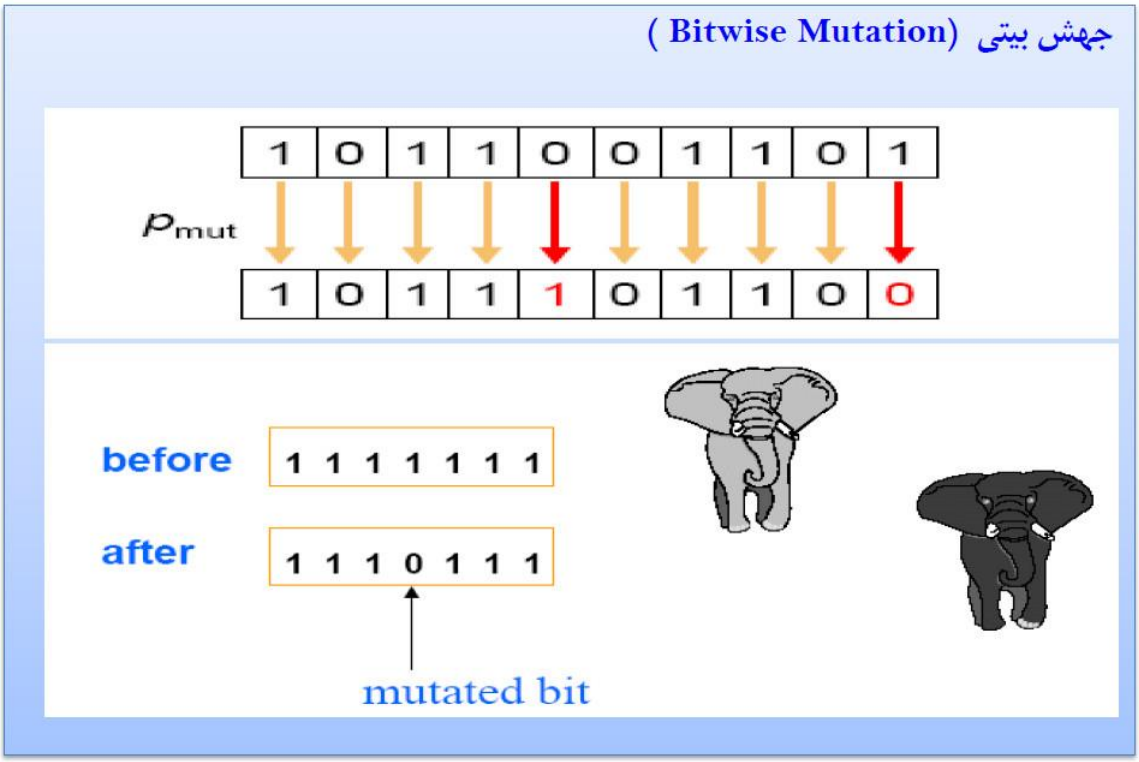




۳- جهش (Mutation)

✓ ویژگی تصادفی بودن و امکان فرار از نقاط بهینه محلی را فراهم میآورد.

برای انجام جهش به این صورت عمل می کنیم:
بصورت تصادفی تعدادی از کروموزوم های فرزند را انتخاب می کنیم به صورت تصادفی مقادیر یک یا چند ژن وی را تغییر می دهیم.



شرط خاتمه الگوریتم

چون که الگوریتم های ژنتیک بر پایه تولید و تست می باشند، جواب مساله مشخص نیست و نمی دانیم که کدامیک از جواب های تولید شده جواب بهینه است تا شرط خاتمه را پیدا شدن جواب در جمعیت تعریف کنیم. به همین دلیل، معیارهای دیگری را برای شرط خاتمه در نظر می گیریم:

- تعداد مشخصی نسل: می توانیم شرط خاتمه را مثلاً ۱۰۰ دور چرخش حلقه اصلی برنامه قرار دهیم.

- عدم بهبود در بهترین شایستگی جمعیت در طی چند نسل متوالی
- واریانس شایستگی جمعیت از یک مقدار مشخصی پائین تر بیاید و یا اینکه در طی چند نسل متوالی مشخص، تغییر نکند.
- بهترین شایستگی جمعیت از یک حد خاصی کمتر شود.

✓شرایط دیگری نیز می توانیم تعریف کنیم و همچنین می توانیم ترکیبی از موارد فوق را به عنوان شرط خاتمه به کار ببندیم.

و در انتها شبه کدی که یک الگوریتم ژنتیک را شرح می‌دهد در زیر آورده شده است:س

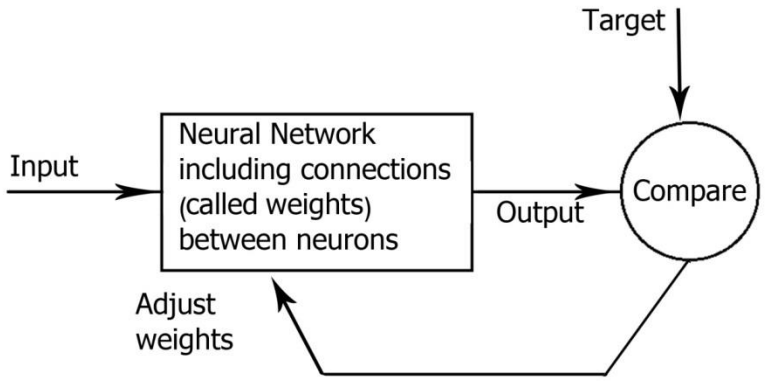
```
GA()
  initialize population
  find fitness of population

  while (termination criteria is reached) do
    parent selection
    crossover with probability pc
    mutation with probability pm
    decode and fitness calculation
    survivor selection
    find best
  return best
```

شبکه عصبی

شبکه عصبی از عناصر به هم مرتبط تشکیل شده است. این عناصر از مطالعات انجام گرفته در زمینه ی سیستم های عصبی الهام گرفته شده اند. به عبارت دیگر هدف شبکه های عصبی، کوشش برای ساخت ماشین هایی است که همانند مغز انسان عمل می کنند. این ماشین ها متشکل از اجزایی می باشند که مشابه عصب های بیولوژیکی رفتار می کنند. کار یک شبکه عصبی، ایجاد یک الگوی خروجی بر اساس الگوی ورودی ارائه شده به شبکه می باشد. طبقه بندی الگو ها فرایندی است که طی آن الگو ها در یک گروه یا گروهی دیگر دست بندی می شوند. اکنون که این جمله ها را می خوانید، مغز شما در حال مرتب نمودن سیگنال هایی است که چشم شما دریافت کرده است به گونه ای که مغز می تواند حروف موجود در صفحه را تشخیص دهد و این حروف را به صورت کلمه، جمله یا پاراگراف به یکدیگر بچسباند. عمل بازشناسی حروف منفرد، یک فرایند بازشناسی الگو است. یعنی نماد های موجود در صفحه الگوهایی هستند که باید بازشناسی شوند.

شبکه های عصبی از عناصر عملیاتی ساده ای ساخته می شوند که به صورت موازی در کنار هم عمل می کنند. این عناصر از سیستم های عصبی زیستی الهام گرفته شده اند. در طبیعت، عملکرد شبکه های عصبی از طریق نحوه اتصال بین اجزا تعیین می شود. بنابراین ما می توانیم یک ساختار مصنوعی به تبعیت از شبکه های طبیعی بسازیم و با تنظیم مقادیر هر اتصال، تحت عنوان وزن اتصال، نحوه ی ارتباط بین اجزای آن را تعیین نماییم.

پس از تنظیم یا همان آموزش شبکه عصبی، اعمال یک ورودی خاص به آن، منجر به دریافت پاسخ خاصی می شود. همانطور که در شکل دیده می شود، شبکه بر مبنای تطابق و همسنجی بین ورودی و هدف، سازگار می شود تا اینکه خروجی شبکه و خروجی مورد نظر ما (هدف) بر هم منطبق گردند. عموماً تعداد زیادی از این زوج های ورودی و خروجی به کار گرفته می شوند تا در این روند که از آن تحت عنوان یادگیری نظارت شده یاد می شود، شبکه آموزش داده شود.



نحوه عملکرد شبکه های عصبی

از شبکه های عصبی برای پیاده سازی توابع پیچیده در زمینه های مختلف از جمله تشخیص الگو، تشخیص هویت، طبقه بندی، پردازش صحبت و تصویر و سیستم های کنترلی استفاده می شود. امروزه شبکه های عصبی را برای حل مسائل دشواری که حل آنها به روش های معمول دشوار می باشد، استفاده می کنند. عموماً برای آموزش شبکه های عصبی از روش های نظارت شده استفاده می شود اما می توان شبکه ها را با روش های آموزش غیر نظارتی نیز آموزش داد.

تاریخچه شبکه های عصبی

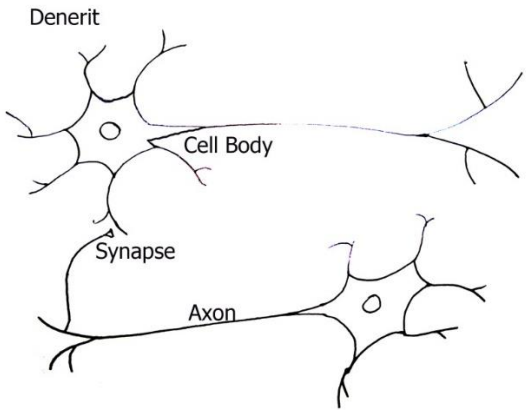
در سال ۱۹۴۹ مدل MP شبکه های عصبی مصنوعی، توسط مک کلوچ و پیت مطرح شد که یک مدل خطی ساده بود. از همان قرن نوزدهم به طور همزمان اما جداگانه از سویی نروفیزیولوزیست‌ها سعی کردند سیستم یادگیری و تجزیه و تحلیل مغز را کشف کنند، و از سوی دیگر ریاضیدانان تلاش کردند تا مدل ریاضی بسازند، که قابلیت فراگیری و تجزیه و تحلیل عمومی مسائل را دارا باشد. اولین کوشش‌ها در شبیه‌سازی با استفاده از یک مدل منطقی توسط مک کلوچ و والتر پیتز انجام شد که امروزه بلوک اصلی سازنده اکثر شبکه‌های عصبی مصنوعی است. این مدل فرضیه‌هایی در مورد عملکرد نرون‌ها ارائه می‌کند. عملکرد این مدل مبتنی بر جمع ورودی‌ها و ایجاد خروجی است. چنانچه حاصل جمع ورودی‌ها از مقدار آستانه بیشتر باشد اصطلاحا نرون برانگیخته می‌شود. نتیجه این مدل اجرای توابع ساده مثل AND و OR بود.

سال ۱۹۶۹ آغاز افول موقت شبکه های عصبی بود، زیرا عدم توانایی شبکه های عصبی در حل مسائل غیر خطی آشکار شد. ANN ها در آن زمان فقط قادر به حل مسائلی بودند که می شد پاسخ های آن را توسط یک خط در محور مختصات از هم جدا کرد.

این دوره از رکود که بیش از بیست سال طول کشید، عملاً مطالعات در این زمینه متوقف شد. اما خیلی زود معلوم شد که در خلال سالیان رکود تحقیقات در مورد شبکه های عصبی، محققانی نظیر ویدرو، کوهونن و گراسبرگ، مطالعه و کار بر روی این موضوع را ادامه داده اند.

الهام از طبیعت

مغز انسان شامل بیش از 10^{11} نرون عصبی می باشد که از طریق حدود 10^{4} اتصال به ازای هر نرون به هم متصل شده اند. به طور کلی می توان سه قسمت کلی برای نرون ها در نظر گرفت: دندریت، بدنه سلول و آکسون. دندریت ها دریافت کننده های درخت شکل از جنس فیبرهای عصبی هستند که سیگنال های الکتریکی را به بدنه ی سلول منتقل می کنند. بدنه ی سلول این سیگنال ها را جمع کرده و یک حد آستانه بر روی آنها اعمال می کند. در نهایت آکسون یک فیبر عصبی بلند است که این سیگنال ها را از بدنه ی سلول به نرون های دیگر متصل می کند. به نقطه اتصال بین آکسون یک سلول عصبی با دندریت سلول های عصبی دیگر سیناپس گفته می شود. شکل زیر یک طرح ساده از دو نرون عصبی را نشان می دهد.



دو نرون عصبی زیستی

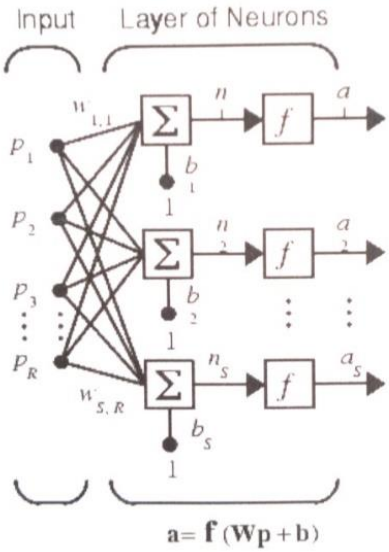
چرا از شبکه های عصبی استفاده می کنیم؟

شبکه های عصبی، با قابلیت قابل توجه آن ها در استنتاج معانی از داده های پیچیده یا مبهم، می تواند برای استخراج الگو ها و شناسایی روش هایی که آگاهی از آن ها برای انسان و دیگر تکنیک های کامپیوتری بسیار پیچیده و دشوار است به کار گرفته شود. یک شبکه عصبی آموزش دیده، می تواند به عنوان یک متخصص در مقوله اطلاعاتی که برای تجزیه تحلیل به آن داده شده به حساب آید. از این متخصص می توان برای براورد وضعیت های دلخواه جدید استفاده کرد. از مزیت های دیگر شبکه های عصبی می توان به موارد زیر اشاره کرد:

- یادگیری انطباق پذیر: قابلیت یادگیری نحوه انجام وظایف بر پایه اطلاعات داده شده برای تمرین و تجربه های مقدماتی.
- سازماندهی توسط خود: یک شبکه عصبی مصنوعی می تواند سازماندهی یا ارائه اش را، برای اطلاعاتی که در طول دوره ی یادگیری دریافت می کند، خودش ایجاد کند.
- عملکرد بهنگام: محاسبات شبکه های عصبی مصنوعی می تواند به صورت موازی انجام شود.
- تحمل اشتباه بدون ایجاد وقفه در هنگام کد گزاری اطلاعات: خرابی جزئی یک شبکه منجر به تنزل کارایی متناظر با آن می شود.

معماری شبکه عصبی

یک یا چند نرون در کنار هم یک لایه از شبکه را تشکیل می دهند. یک شبکه می تواند از یک یا چند لایه این چنینی تشکیل شود. در شکل زیر یک شبکه تک لایه با R ورودی و S نرون نشان داده شده است.



معماری شبکه عصبی

در این شبکه اعضای بردار ورودی P به همه ی نرون ها اعمال می شوند و پس از ضرب در بردار وزن ها و جمع بایاس به تابع انتقال اعمال شده و خروجی حاصل می گردد. خروجی شبکه بالا یک لایه خواهد بود. هیچ لزومی ندارد که تعداد ورودی ها (R) با تعداد نرون ها (P) برابر باشند.

شبکه های پس انتشار

شبکه های پس انتشار یا BackPropagation که به اختصار BP خوانده می شوند، یک شبکه ی چند لایه با تابع انتقال غیرخطی و قاعده ی یادگیری ویدرو- هوف می باشد.

الگوریتم LMS یا روش یادگیری ویدروهوف مبتنی بر روش تقریبی از Steepest descen است. در این روش خطای مجذور میانگین از بدست آوردن مجذور خطا از هر تکرار بدست می‌آید. در واقع این الگوریتم شبکه را به سمت تنها نقطه ی مینیمم می برد، مشروط به اینکه نرخ یادگیری آن مناسب اختیار شود.

از بردار ورودی و هدف، در راستای آموزش این نوع شبکه برای تقریب زدن یک تابع، یافتن رابطه بین ورودی و خروجی و دسته بندی ورودی ها استفاده می شود. یک شبکه ی BP با دارا بودن

بایاس، یک لایه sigmoid و یک لایه ی خروجی خطی، توانایی تخمین زدن هر تابعی با تعداد نقاط ناپیوستگی محدود را داراست.

BP استاندارد یک الگوریتم با کاهش شیب می باشد که در آن وزن های شبکه در جهت خلاف شیب تابع کارایی حرکت می کنند. لغت پس انتشار به رفتار شبکه BP در محاسبه ی شیب در شبکه های غیر خطی چند لایه اشاره دارد.

شبکه های feedforward با الگوریتم BP

این شبکه ها اغلب دارای یک یا چند لایه ی مخفی از نرون های Sigmoid بوده و از لایه ی پایانی خطی استفاده می کنند. وجود چند لایه از نرون ها با یک تابع انتقال غیر خطی به شبکه اجازه می دهد که توانایی یادگیری رابطه خطی و غیر خطی را بین ورودی ها و خروجی ها داشته باشد. لایه ی خروجی این شبکه این امکان را می دهد که خروجی خارج از محدوده ۱+ و ۱- داشته باشد. البته اگر به خروجی در محدوده ی ۰ و ۱ نیاز داشته باشیم، می توان از تابع Logsig در لایه خطی استفاده کرد.

شبکه های عصبی ایستا و پویا

شبکه های عصبی را می تواند به دو دسته تقسیم کرد:

- شبکه های ایستا
- شبکه های پویا

شبکه های عصبی ایستا یا feedforward فاقد اجزای پسخوردی و تاخیری می باشند. به طور خلاصه در شبکه های پیش خور، فعالیت نورون ها از لایه ی ورودی به سمت لایه ی خروجی انتشار می یابد و تمامی اتصالات در لایه ها رو به جلو است. در این گونه شبکه ها ما نباید نگران این باشیم که آیا بردارهای ورودی به ترتیب زمانی مشخصی اتفاق می افتد یا خیر. بنابراین ما می توانیم با ورودی ها به صورت همزمان رفتار کنیم. دقت داشته باشید که اگر شبکه ایستا باشد ورودی حتماً

همزمان خواهد بود. در واقع ترتیب ورودی ها اصلاً مهم نیست. خروجی این شبکه ها مستقیماً از طریق اتصالات رو به جلو شبکه تعیین می شود.

ویژگی های مهم شبکه های پیش خور

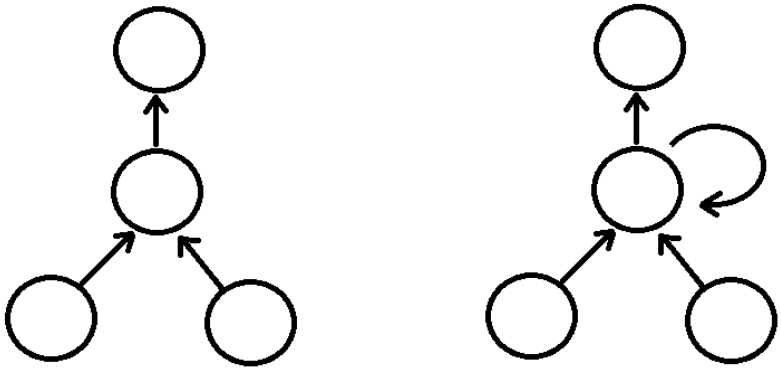
- در شبکه های پیش خور اطلاعات تنها در یک جهت یعنی از سمت ورودی به خروجی جریان دارند. البته این در مورد عملکرد شبکه است و نه یادگیری.
- در این شبکه ها هر ورودی یک خروجی دارد.
- مسئله زمان در آن ها قابل لحاظ نیست.
- تمامی اتصالات رو به جلو هستند.

در شبکه های پویا خروجی شبکه علاوه بر ورودی فعلی آن، به ورودی ها و خروجی های قبل و حالات شبکه نیز بستگی دارد. زمانیکه یک شبکه شامل تاخیر می شود طبیعتاً ترتیبی از بردارهای ورودی که با یک ترتیب خاص زمانی اتفاق می افتد، ورودی شبکه را تشکیل خواهند داد. شبکه های پویا به دو دسته کلی تقسیم می شوند: شبکه هایی که فقط دارای اتصالات رو به جلو هستند و شبکه هایی که دارای اتصالات پسخوردی می باشند. از نوع دوم معمولاً تحت عنوان شبکه های recurrent یاد می شود.

شبکه های عصبی recurrent

در شبکه های بازگشتی، خروجی نرون ها در لایه های بالاتر، به ورودی نرون های لایه های اولیه باز می گرد و به همین دلیل به آن ها شبکه های بازگشتی می گویند. اگر این مدل شبکه ی عصبی را یک گراف جهت دار در نظر بگیریم، این شبکه ها دارای دور هستند.

دو شکل زیر را در نظر بگیرید:

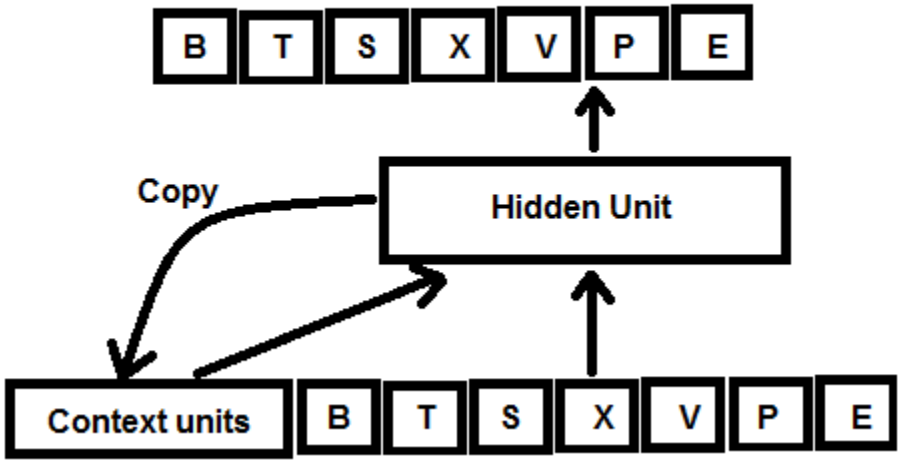


مقایسه دو شبکه

در شکل سمت چپ یک شبکه ی پیش خور ساده نشان داده شده است. شکل سمت راست مانند شکل سمت چپ است با این تفاوت که یک اتصال اضافی از لایه ی مخفی به خودش دارد. هر زمان که یک الگو ارائه می شود، واحد محاسبه فقط به عنوان FF فعال می شود. ورودی این شبکه شامل یک انعکاس حالت شبکه است. زمانی که الگوی بعدی به شبکه ارائه می شود، حالت لایه ی مخفی و لایه ی خروجی، تابعی از همه ی حالت هایی است که شبکه تا کنون دیده است.

رفتار شبکه بر اساس History آن است. یک بار که اجازه ی پس خورد به اتصال داده می شود، توپولوژی شبکه به سادگی اتفاق می افتد، یعنی می توان به هر لایه ای، حتی به خودش متصل شد.

فعالیت قرار گرفته در شبکه برای این است که Item بعدی را توسط Item موجود در زمان t و یک نمایش داخلی از وضعیت مجموعه ای از لایه های پنهان از گام زمانی قبلی پیش بینی کند. شکل زیر الگوریتم به کار گرفته شده در شبکه های بازگشتی را به سادگی شرح می دهد:



الگوریتم شبکه ی بازگشتی

الگوریتم شبکه ی بازگشتی ساده به این شکل است که شبکه یاد می گیرد که خروجی ها درست بر اساس پالس داده شده ی جاری، که بر پایه ی اطلاعاتی است که در گروه های Context اتفاق می افتد، تولید شود. هر خانه یک مخزن از واحد ها را نشان می دهد. هر پیکان رو به جلو، یک مجموعه ی کامل از ارتباطات قابل آموزش را، از هر واحد ارسالی به هر واحد دریافتی در مخزن بعدی را نشان می دهد. پیکان رو به عقب از لایه ی پنهان به لایه ی زمینه یک عملیات کپی را معنی می کند.

شبکه های عصبی recurrent یا بازگشتی دارای کاربردهای متنوعی در شاخه های علمی مختلف می باشند. این شبکه ها به جز در موارد تقریب توابع، در دسته بندی الگو ها نیز توانایی بسیار بالایی دارند.

یادگیری در شبکه های بازگشتی

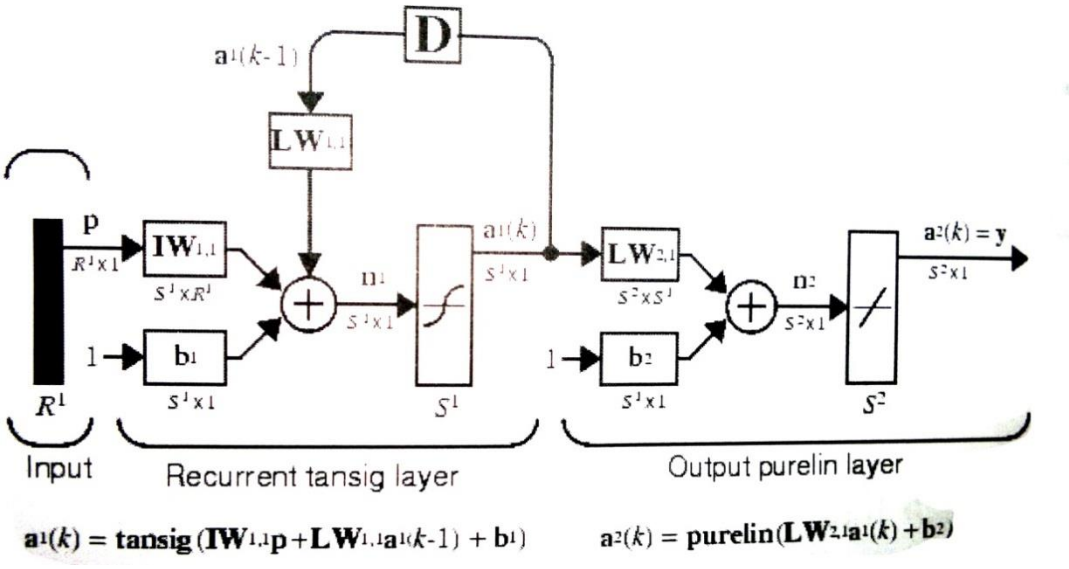
آموزش در شبکه های بازگشتی به سه نوع کلی آموزش با نظارت، بی نظارت و یادگیری تقویتی تقسیم می شود. در روش یادگیری با نظارت داده های آموزشی به همراه خروجی صحیح موجود است. الگوریتم آموزش دهنده ی شبکه می بایست، داده های آموزشی را به شبکه داده و خروجی کنونی شبکه را با خروجی صحیح مقایسه نماید، سپس این الگوریتم معلم می تواند، وزن های شبکه

را طوری تغییر دهد که خروجی شبکه به خروجی مطلوب نزدیک تر شود، و یا به عبارتی خطای خروجی شبکه به طور کلی کم شود. این آموزش به شبکه های عصبی به امیدی انجام می شود که این شبکه ها خاصیت تعمیم به دست آورده و برای ورودی های جدید، خروجی مطلوب را تولید نمایند. دو نوع ازشبکه های عصبی بازگشتی عبارت اند از: شبکه های عصبی المان و شبکه های عصبی هاپفیلد.

شبکه ی بازگشتی المان

شبکه های المان یک شبکه پس انتشار دو لایه به همراه یک پسخورد از خروجی لایه مخفی به ورودی لایه مخفی می باشند. این پسخورد به شبکه در تشخیص الگوهای زودگذر و وابسته به زمان کمک می کند. به عبارت دیگر این پسخورد به شبکه اجازه ی یادگیری تشخیص الگو را می دهد.

ویژگی اختصاصی معماری شبکه های المان در داشتن یک لایه ی اضافی از نرون ها است که عملکرد های اخیر را در نرون های لایه ی پنهان ذخیره می کند و بعد از به تاخیر انداختن این مقادیر، برای واحد زمان آن ها را به عنوان ورودی اضافی به نرون های لایه ی پنهان باز می گرداند. معماری این شبکه دو لایه در شکل زیر نشان داده شده است.

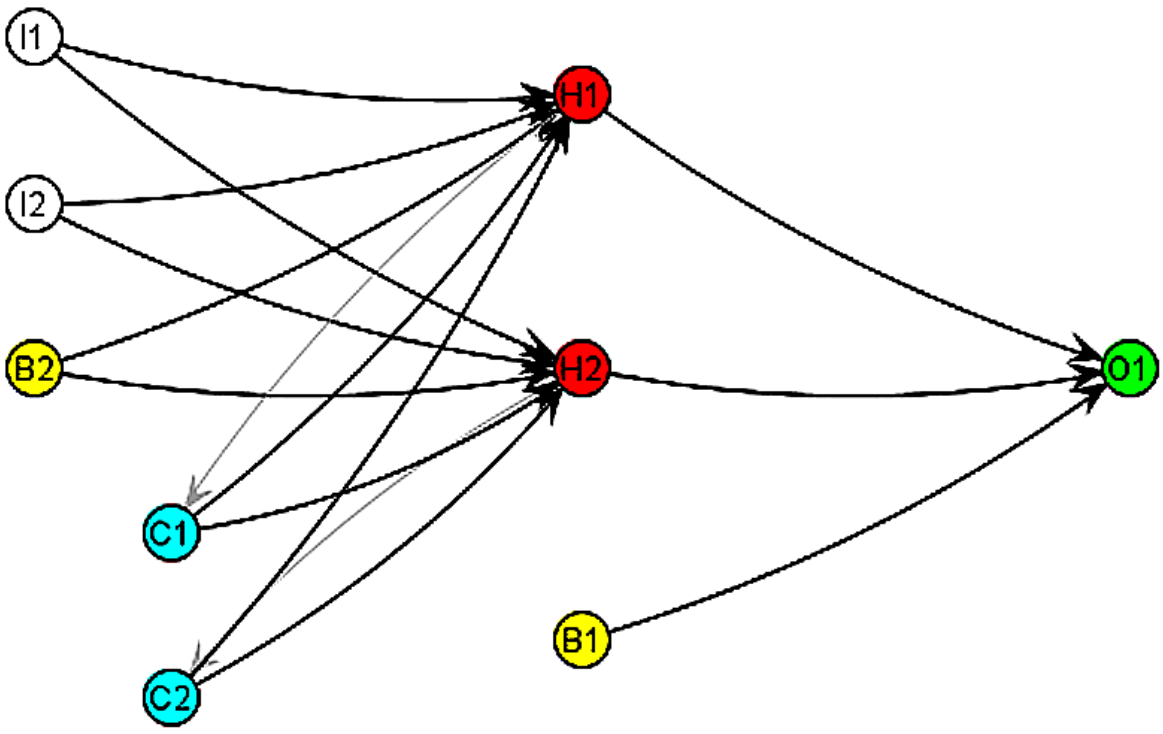


معماری شبکه المان

شبکه المان در لایه مخفی خود از نرون های tansig و در لایه خروجی از تابع pureline استفاده می کنند. این ترکیب خاص در شبکه های دو لایه قادر به تخمین هر تابعی با تعداد محدود ناپیوستگی می باشد. در این راستا شبکه باید در لایه مخفی از تعداد نرون های کافی برخوردار باشد. استفاده از نرون های بیشتر در این لایه باعث منطبق شدن بیشتر آن بر تابع مورد نظر می شود.

باید به این نکته توجه کرد که تفاوت اصلی این شبکه ها با شبکه های دو لایه معمول در پسخورد موجود در ساختار آنها می باشد. تاخیر موجود در این پسخورد اطلاعات مربوط به گام قبلی را در گام فعلی در اختیار شبکه قرار می دهد. بنابراین اگر دو شبکه المان متفاوت با وزن ها و بایاس های یکسان داشته باشیم که به آنها در گام زمانی مشخصی ورودی مشخصی داده شود باز هم خروجی ها می تواند متفاوت باشد و این مسئله به خاطر تفاوت در حالت های پسخوردی است.

در بیان دیگر شبکه ی المان یک ماشین حالت محدود است که یاد می گیرد که چه حالتی را به یاد آورد. در شکل زیر یک شبکه المان نشان داده شده است.



شبکه المان

این شبکه دارای دو نرون ورودی و دو نرون مخفی و یک خروجی است. همچنین دو نرون تحت عنوان context وجود دارد. Context ها ورودی را از لایه های مخفی دریافت می کنند و خروجی را دوبار به لایه های مخفی پاس می دهند. لایه ی context همیشه خروجی لایه ی مخفی را نگه می دارند و اطلاعات مربوطه را در iteration بعدی باز پخش می کنند. این کار به این خاطر است که به شبکه اجازه داده می شود که یک نمونه ی کوچکی از حافظه را فرا بگیرد.

ایجاد یک شبکه المان

این شبکه توسط تابع newelm قابل تعریف است. لایه های مخفی معمولاً دارای تابع انتقال tansig می باشند که برای تابع newelm به صورت پیش فرض در نظر گرفته شده است. لایه خروجی نیز معمولاً از تابع purelin استفاده می کند. شرح این توابع در ادامه بررسی می شود.

تابع یادگیری پیش فرض برای شبکه های پس انتشار trainbfg است. البته از تابع trainlm نیز می توان استفاده کرد. زیرا این تابع سریع تر عمل می کند. البته این مسئله الزاماً درباره ی شبکه های المان صادق نیست. تابع کارایی مورد استفاده mse می باشد.

شبکه های هاپفیلد برای ذخیره سازی یک یا چند بردار هدف پایا کاربرد دارند. می توان به این بردارهای پایا به صورت حافظه ای نگاه کرد که شبکه آنها را زمانی که با یک ورودی مشابه مواجه می شوند، به یاد می آورد. شبکه های المان با تابع newelm و شبکه های هاپفیلد با تابع newhop قابل ایجاد می باشند.