

# دستورکار آزمایشگاه هوش محاسباتی

شهریور ۱۴۰۳



# فهرست مطالب

۱۰	.....	۱.۰ چکیده .....
۱۱	.....	آزمایش اول (آشنایی با متلب و پایتون)
۱۱	.....	۱.۱ پیش گزارش .....
۱۱	.....	۲.۱ مقدمه .....
۱۱	.....	۱.۲.۱ آشنایی با کد نویس در متلب .....
۱۳	.....	۲.۲.۱ آشنایی با کد نویسی در پایتون .....
۱۵	.....	۳.۱ شرح آزمایش .....
۱۶	.....	۴.۱ تمرین .....
۱۷	.....	آزمایش دوم(پیاده سازی پرسپترون)
۱۷	.....	۱.۲ پیش گزارش .....
۱۷	.....	۲.۲ مقدمه .....
۱۸	.....	۱.۲.۲ الگوریتم یادگیری .....
۱۹	.....	۳.۲ شرح آزمایش .....
۱۹	.....	۱.۳.۲ پیاده سازی پرسپترون .....
۱۹	.....	۲.۳.۲ داده ورودی .....
۲۰	.....	۳.۳.۲ بهبود مدل .....
۲۰	.....	۴.۲ تمرین .....
۲۳	.....	آزمایش سوم(پیاده سازی شبکه عصبی چند لایه)
۲۳	.....	۱.۳ پیش گزارش .....

۲۳	۲.۳ مقدمه . . . . .
۲۵	۱.۲.۳ الگوریتم پس انتشار خط . . . . .
۲۵	۳.۳ شرح آزمایش . . . . .
۲۵	۱.۳.۳ پیاده سازی شبکه عصبی <sup>۲</sup> لایه . . . . .
۲۶	۲.۳.۳ تست مدل . . . . .
۲۶	۴.۳ تمرین . . . . .
۲۹	<b>آزمایش چهارم (پیاده سازی الگوریتم <math>k - mean</math>)</b>
۲۹	۱.۴ پیش گزارش . . . . .
۲۹	۲.۴ مقدمه . . . . .
۲۹	۱.۲.۴ الگوریتم $k - mean$ . . . . .
۳۰	۳.۴ شرح آزمایش . . . . .
۳۱	۱.۳.۴ پیاده سازی الگوریتم . . . . .
۳۱	۲.۳.۴ تست الگوریتم . . . . .
۳۲	۳.۳.۴ ارزیابی الگوریتم . . . . .
۳۲	۴.۳.۴ محدودیت های $kmean$ . . . . .
۳۳	۴.۴ استفاده از کتابخانه های آماده . . . . .
۳۳	۵.۴ کاهش حجم عکس به وسیله $k - mean$ . . . . .
۳۴	۶.۴ استفاده از کتابخانه های آماده . . . . .
۳۴	۷.۴ تمرین . . . . .
۳۷	<b>آزمایش پنجم(پیاده سازی شبکه عصبی <math>RBF</math>)</b>
۳۷	۱.۵ پیش گزارش . . . . .
۳۷	۲.۵ مقدمه . . . . .
۳۷	۱.۲.۵ ساختار شبکه . . . . .
۳۸	۲.۲.۵ تابع گوسین . . . . .
۳۹	۳.۲.۵ توابع گوسین و $RBF$ . . . . .
۴۰	۴.۲.۵ قانون تازه سازی وزن های شبکه . . . . .

۴۰	.....	۳.۵ شرح آزمایش
۴۱	.....	۱.۳.۵ پیاده سازی شبکه
۴۲	.....	۲.۳.۵ تست شبکه
۴۲	.....	۴.۵ استفاده از Tensorflow
۴۳	.....	۵.۵ تمرین
۴۵	.....	<b>آزمایش ششم (پیاده سازی شبکه عصبی <i>Hopfield</i>)</b>
۴۵	.....	۱.۶ پیش گزارش
۴۵	.....	۲.۶ مقدمه
۴۶	.....	۱.۲.۶ تحلیل فیزیکی حافظه
۴۷	.....	۲.۲.۶ شبکه هاپفیلد
۴۷	.....	۳.۶ شرح آزمایش
۴۸	.....	۴.۶ تمرین
۵۱	.....	<b>آزمایش هفتم(پیاده سازی شبکه عصبی <i>CNN</i>)</b>
۵۱	.....	۱.۷ مقدمه
۵۳	.....	۲.۷ شرح آزمایش
۵۷	.....	۳.۷ تمرین
۶۱	.....	<b>آزمایش هشتم (پیاده سازی شبکه عصبی <i>RNN</i>)</b>
۶۱	.....	۱.۸ پیش گزارش
۶۱	.....	۲.۸ مقدمه
۶۱	.....	۱.۲.۸ ساختار شبکه عصبی بازگشتی
۶۲	.....	۳.۸ شرح آزمایش
۶۲	.....	۱.۳.۸ داده ورودی
۶۳	.....	۲.۳.۸ آموزش مدل
۶۳	.....	۳.۳.۸ بهبود مدل
۶۴	.....	۴.۸ تمرین

## آزمایش نهم (شناسایی به کمک شبکه عصبی)

۶۵	۱.۹	پیش گزارش
۶۵	۲.۹	مقدمه
۶۶	۱.۲.۹	شناساگر استاتیکی
۶۶	۲.۲.۹	شناساگر پیوسته
۶۷	۳.۲.۹	مدل موازی
۶۷	۴.۲.۹	ترکیب شناساگر استاتیکی و مدل موازی
۶۹	۳.۹	شرح آزمایش
۶۹	۱.۳.۹	پیاده سازی شناساگر استاتیکی
۷۱	۲.۳.۹	پیاده سازی شناساگر پیوسته
۷۳	۳.۳.۹	تست شناساگر شبیه سازی شده
۷۴	۴.۹	تمرین

## آزمایش دهم (کنترل به کمک شبکه عصبی)

۷۷	۱.۱۰	پیش گزارش
۷۷	۲.۱۰	مقدمه
۷۷	۱.۲.۱۰	کنترل دینامیک معکوس
۷۸	۳.۱۰	شرح آزمایش
۷۹	۴.۱۰	تمرین

## آزمایش یازدهم (پیاده سازی الگوریتم $DQN$ )

۸۱	۱.۱۱	پیش گزارش
۸۱	۲.۱۱	مقدمه
۸۲	۱.۲.۱۱	مفاهیم اولیه یادگیری تقویتی
۸۸	۳.۱۱	شرح آزمایش
۸۹	۴.۱۱	تمرین

۹۱	.....	۱.۱۲ پیش گزارش
۹۱	.....	۲.۱۲ مقدمه
۹۱	.....	۱.۲.۱۲ خودرو زمینی بدون سرنشین
۹۲	.....	۲.۲.۱۲ ربات نقاش ۲ بعدی
۹۲	.....	۳.۱۲ شرح آزمایش
۹۳	.....	۱.۳.۱۲ آزمایش ۱- شناسایی خودرو زمینی بدون سرنشین
۹۳	.....	۲.۳.۱۲ آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی
۹۴	.....	۳.۳.۱۲ آزمایش ۳ - کنترل خودروی زمینی بدون سرنشین
۹۴	.....	۴.۱۲ تمرین

۹۵	.....	آزمایش دوازدهم (آشنایی با مفاهیم فازی)
۹۵	.....	۱.۱۳ پیش گزارش
۹۵	.....	۲.۱۳ مقدمه
۹۶	.....	۱.۲.۱۳ تابع عضویت
۹۶	.....	۲.۲.۱۳ نقطه عبور
۹۶	.....	۳.۲.۱۳ عملگرها در منطق فازی
۹۶	.....	۴.۲.۱۳ اجتماع
۹۷	.....	۳.۱۳ شرح آزمایش
۹۸	.....	۴.۱۳ تمرین

۱۰۱	.....	آزمایش سیزدهم (کنترلر فازی)
۱۰۱	.....	۱.۱۴ پیش گزارش
۱۰۱	.....	۲.۱۴ مقدمه
۱۰۱	.....	۱.۲.۱۴ جعبه ابزار فازی متلب
۱۰۴	.....	۳.۱۴ شرح آزمایش
۱۰۵	.....	۴.۱۴ تمرین

۱۰۹	.....	آزمایش چهاردهم (کنترلر PID فازی)
-----	-------	----------------------------------

۱۰۹	.....	۱.۱۵ پیش گزارش
۱۰۹	.....	۲.۱۵ مقدمه
۱۰۹	.....	۱.۲.۱۵ کنترل <i>PID</i>
۱۱۳	.....	۳.۱۵ شرح آزمایش
۱۱۳	.....	۴.۱۵ تمرین
۱۱۵	.....	<b>آزمایش پانزدهم</b> ( <i>c – mean</i> )
۱۱۵	.....	۱.۱۶ پیش گزارش
۱۱۵	.....	۲.۱۶ مقدمه
۱۱۶	.....	۱.۲.۱۶ آشنایی با جعبه ابزار <i>c – mean</i> فازی متلب
۱۱۷	.....	۲.۲.۱۶ ابزار فازی در محیط <i>command – line</i> متلب
۱۱۷	.....	۳.۱۶ شرح آزمایش
۱۱۷	.....	۱.۳.۱۶ دسته بندی با استفاده از جعبه ابزار <i>fcm</i>
۱۱۸	.....	۲.۳.۱۶ دسته بندی <i>fcm</i> در <i>Command – line</i>
۱۱۹	.....	۴.۱۶ تمرین
۱۲۱	.....	<b>آزمایش شانزدهم</b> ( <i>anfis</i> )
۱۲۱	.....	۱.۱۷ پیش گزارش
۱۲۱	.....	۲.۱۷ مقدمه
۱۲۱	.....	۱.۲.۱۷ جعبه ابزار <i>anfis</i> متلب
۱۲۲	.....	۳.۱۷ شرح آزمایش
۱۲۳	.....	۴.۱۷ تمرین
۱۲۵	.....	<b>پروژه بخش فازی</b>
۱۲۵	.....	۱.۱۸ پیش گزارش
۱۲۵	.....	۲.۱۸ مقدمه
۱۲۵	.....	۱.۲.۱۸ خودرو زمینی بدون سرنشین
۱۲۶	.....	۲.۲.۱۸ ربات نقاش ۲ بعدی

۱۲۶	۳.۱۸	شرح آزمایش
۱۲۷	۱.۳.۱۸	آزمایش ۱ - کنترل خودروی زمینی بدون سرنشین
۱۲۷	۲.۳.۱۸	آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی
۱۲۹		<b>پیوست</b>
۱۲۹	آ	ربات نقاش ۲ بعدی
۱۳۰	۱.آ	ترسیم با ربات های نقاش
۱۳۱	ب	робات زمینی ( <i>UGV</i> )
۱۳۱	ب.۱	مدلسازی
۱۳۷	ب.۲	ارتباط با <i>UGV</i>

## ۱۰ چکیده

در این آزمایشگاه به پیاده سازی نرم افزاری و سخت افزاری مفاهیم بیان شده در درس هوش محاسباتی پرداخته می شود.

گروه های در نظر گرفته شده برای این آزمایشگاه، گروه های ۳ نفره است. در ۱۱ آزمایش اول به پیاده سازی مفاهیم شبکه عصبی و یادگیری تقویتی پرداخته می شود سپس در قالب یک پروژه به پیاده سازی سخت افزاری مفاهیم بیان شده پرداخته خواهد شد. در آزمایش دوازدهم تا پانزدهم به پیاده سازی مفاهیم سیستم های فازی پرداخته می شود . در آزمایش ۱۶ با ترکیب سیستم های عصبی و فازی آشنا خواهید شد و سپس در قالب یک پروژه مفاهیم بیان شده بر روی سیستم های در نظر گرفته شده پیاده سازی سخت افزاری می شوند.

هر آزمایش از ۴ بخش پیش گزارش ، مقدمه ، شرح آزمایش و تمرین تشکیل شده است. پیش گزارش هر آزمایش را قبل از اجرای آزمایش به مسئول آزمایشگاه تحويل دهید. در قسمت مقدمه پیش نیازهای لازم برای هر آزمایش بیان شده است. در قسمت شرح آزمایش روند کلی و هدف هر آزمایش بیان شده است. در قسمت تمرین سوالاتی درباره آزمایش انجام شده در نظر گرفته شده است. تمرین هر آزمایش را قبل از شروع آزمایش بعد به مسئول مربوطه تحويل دهید(مسائل شبیه سازی را در moodle درس بار گزاری کنید).

# آزمایش اول (آشنایی با متلب و پایتون)

## ۱.۱ پیش گزارش

۱. کاربردهای زبان برنامه نویسی پایتون را بیان کنید.
۲. تفاوت زبان برنامه نویسی متلب و پایتون چیست؟
۳. مزیت های زبان پایتون در برابر متلب و برعکس را بیان کنید.

## ۲.۱ مقدمه

در این آزمایش قصد داریم با زبان های برنامه نویسی متلب و پایتون آشنا شویم. در ادامه آزمایش ها در محیط متلب یا پایتون اجرا می شوند. در ابتدا مروری کوتاه بر روی کد نویسی در محیط های متلب و پایتون انجام می شود.

### ۱.۲.۱ آشنایی با کد نویس در متلب

متلب یکی از پرکاربرد ترین زبان های برنامه نویسی به شمار می رود. متلب دارای کتابخانه های آماده و مفید در حوزه های مختلفی مانند پردازش تصویر و هوش مصنوعی است. در این قسمت قصد داریم مروری کوتاه بر چگونگی برنامه نویسی در این محیط داشته باشیم. در ابتدا با چگونگی تعریف تابع در متلب آشنا خواهیم شد سپس مروری کوتاه بر روی آرایه ها در متلب انجام می شود با گذر از این موضوع به ترسیم نمودار در متلب پرداخته می شود. این بخش با معرفی کردن ابزار *help* در متلب پایان می پذیرد.

توابع در برنامه نویسی توسعه دادن کد را آسان تر می کنند. با کمک توابع توضیح کارکرد کد نیز ساده تر می شود.  
برای ایجاد یک تابع در متلب قدم های زیر را دنبال کنید:

۱. بر روی منوی *New* کلیک کرده و گزینه‌ی *Function* را انتخاب می‌کنیم

۲. در طرف چپ مساوی داخل کروشه خروجی‌های تابع با ذکر نام و تفکیک با ایجاد یک فاصله با ویرگول، مشخص می‌شوند.

۳. در طرف راست مساوی نیز، داخل کروشه ورودی‌های تابع به صورت مشابه تعریف می‌شوند.

۴. در خط بعد روابط میان ورودی و خروجی‌ها تعریف می‌شود.

## آرایه‌ها و ماتریس‌ها

آرایه‌ها و ماتریس‌ها کاربرد زیادی در انواع مسایل برنامه نویسی دارند.  
آرایه‌ها در متلب به وسیله یک کروشه تعریف می‌شوند. اعضای آرایه با استفاده از ، و یا فاصله از یکدیگر جدا می‌شوند.

شکل شماره ۱.۱ تعریف یک آرایه که دارای ۴ عضو است را در متلب نمایش می‌دهد.

$$a = [1 \ 2 \ 3 \ 4]$$

شکل ۱.۱: تعریف آرایه در متلب

ماتریس‌ها در متلب همانند آرایه‌ها به وسیله کروشه مشخص می‌شوند سطر‌های یک ماتریس با استفاده از ”؛“ یا ”،“ از یکدیگر جدا می‌شوند.

شکل ۲.۱ چگونگی تعریف یک ماتریس در محیط متلب را نمایش می‌دهد.

$$a = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 10]$$

شکل ۲.۱: تعریف ماتریس در متلب

یکی از ویژگی های نرم افزار متلب به تصویر کشیدن داده های مختلف و رسم نمودار است.

متلب امکان ترسیم نمودارهای ۲ بعدی و ۳ بعدی را فراهم می سازد.

برای رسم نمودار ۲ بعدی از دستور  $plot(x, y)$  استفاده می شود. در این دستور  $x$  داده های منتظر بر روی محور افقی و  $y$  داده های منتظر بر روی محور عمودی را نمایش می دهد. دستور  $plot$  ورودی های خود را به صورت آرایه دریافت می کند.

برای رسم نمودار های ۳ بعدی در متلب ابتدا با استفاده از دستور  $meshgrid$  بازه مربوط به داده های  $x$  و  $y$  مشخص می شود. سپس محور  $z$  بر حسب  $x$  و  $y$  تعریف می شود. در نهایت با استفاده از دستور  $mesh(x, y, z)$  نمودار به صورت ۳ بعدی رسم می شود.

## راهنمای متلب

در هنگام کد نویسی با متلب ممکن است با مشکلات و سوالات مختلفی روبرو شوید.

متلب ابزار مفیدی برای یافتن پاسخ به سوالات و حل کردن مشکلات مختلف است. *help* با باز کردن محیط متلب در گوشه سمت راست و بالای صفحه محیط مستطیل شکل برای جستجو را مشاهده می کنید. با جستجو کردن کلیدواژه خود در این قسمت می توانید درباره کلید واژه جستجو شده اطلاعات بیشتری استخراج کنید.

برای کد نویسی در متلب به دستورات مختلف مانند *while* و *for* و *if* احتیاج خواهد داشت. با نحوه کدنویسی به وسیله این دستورات به کمک ابزار *help* در متلب می توانید آشنا شوید.

### ۲.۲.۱ آشنایی با کد نویسی در پایتون

پایتون یکی از زبان های برنامه نویسی پرکاربرد در حوزه هوش است. سرعت اجرای بالا یکی از مزیت های این زبان برنامه نویسی است.

در این قسمت مروری کلی بر کد نویسی در محیط پایتون انجام می شود. در ابتدا با چگونگی تعریف تابع در پایتون آشنا خواهیم شد سپس مروری کوتاه بر روی آرایه ها در پایتون انجام می شود با گذر از این موضوع به ترسیم نمودار در پایتون پرداخته می شود. این بخش با معرفی کردن ابزار *help* در پایتون پایان می پذیرد.

تابع در پایتون به وسیله کلید واژه `def` تعریف می شود. بعد از نوشتن کلید واژه اسم تابع و سپس ورودی های تابع مشخص می شود. بعد از مشخص کردن ورودی با قرار دادن : می توانید بدن تابع خود را بنویسید. در نهایت پارامتر خروجی تابع به وسیله `return` مشخص می شود.

شکل ۳.۱ چگونگی تعریف تابع در پایتون را نمایش می دهد.

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

شکل ۳.۱: تعریف تابع در پایتون

## آرایه ها و ماتریس ها

یکی از کتابخانه های مفید در پایتون ، کتابخانه `numpy` است. آرایه ها و ماتریس ها با استفاده از این کتابخانه در محیط پایتون تعریف می شوند.

برای استفاده از کتابخانه ابتدا با استفاده از دستور `numpyimport` کتابخانه مورد نظر را وارد کد خود کنید. سپس با استفاده از دستور `(np.array()` می توانید آرایه ها و ماتریس های خود را ایجاد کنید.

شکل ۱.۴ یک مثال از چگونگی تعریف آرایه ها در پایتون را نشان می دهد. همان طور که در شکل نمایش داده شده است اعضای آرایه به وسیله ”،“ از یکدیگر جدا می شوند.

شکل ۱.۵ یک مثال از تعریف ماتریس در پایتون را نشان می دهد. همان طور که در شکل نمایش داده شده است در درون کروشه اصلی هر یک از سطرهای ماتریس در یک کروشه مشخص می شود. اعضای هر سطر با استفاده از ”،“ از یکدیگر جدا می شوند.

```
np.array([1, 2, 3])
```

شکل ۴.۱: تعریف آرایه در پایتون

```
np.array([[1, 2], [3, 4]])
```

شکل ۵.۱: تعریف ماتریس در پایتون

در پایتون نیز همانند متلب امکان ترسیم نمودارهای ۲ بعدی و ۳ بعدی وجود دارد. برای رسم نمودار در پایتون می‌توانید از کتابخانه *matplotlib* استفاده کنید. برای استفاده از این کتابخانه لازم است در ابتدا کتابخانه را *import* کنید. برای *import* کردن کتابخانه کافی است دستور *import matplotlib* را اجرا کنید. بعد از *import* کردن کتابخانه با استفاده از دستور *matplotlib.pyplot* می‌توانید داده‌های خود را رسم کنید. همانند متلب *x* آرایه‌ای از داده‌های متناظر بر روی محور افقی و *y* آرایه‌ای از داده‌های متناظر با هر *x* می‌باشد.

### راهنمای پایتون

برای استفاده از ابزار راهنمای پایتون برای آشنایی بیشتر با این زبان برنامه نویسی می‌توانید (*help*) را در کنسول واقع شده در سمت راست محیط برنامه نویسی وارد کنید. این تابع کلیدوازه مورد جستجو را به عنوان ورودی دریافت می‌کند.

## ۳.۱ شرح آزمایش

در این قسمت به پیاده سازی یک تابع ساده در محیط متلب و پایتون پرداخته می‌شود. توجه کنید تابع یک بار در محیط پایتون و یک بار در محیط متلب باید پیاده سازی شود.

تابع *sigmoid* و مشتق آن از پرکاربرد ترین توابع در شبکه عصبی هستند. تابع *sigmoid* و مشتق آن توسط معادلات ۱.۱ توصیف شده اند.

$$\begin{aligned} f(x) &= \frac{2}{1 + \exp(-x)} - 1 \\ \dot{f}(x) &= \frac{2\exp(-x)}{(1 + \exp(-x))^2} = \frac{1 - (f(x))^2}{2} \end{aligned} \quad (1.1)$$

تابعی به اسم *sigmatrix* بنویسید که یک ماتریس و ابعاد آن را از ورودی دریافت کند و دارای ۲ خروجی به صورت زیر باشد. ( $A_{ij}$  اعضای ماتریس ورودی را نشان می‌دهد و  $\text{sigmoid}$  مشتق تابع *sigmoid* است.  $n$  و  $m$  نیز به ترتیب برابر با تعداد سطرها و ستون‌های ماتریس ورودی است):

$$1. \text{ خروجی اول : } \sum_{i=1}^n \sum_{j=1}^m \text{sigmoid}(A_{ij})$$

$$\sum_{i=1}^n \sum_{j=1}^m \text{sigmoid}(A_{ij})$$

برای پیاده سازی تابع خواسته شده مراحل زیر را دنبال کنید:

۱. یک تابع بنویسید که یک عدد را از ورودی دریافت کند و مقدار تابع  $\text{sigmoid}$  را در آن نقطه باز گرداند.
۲. تابعی بنویسید که یک عدد را از ورودی دریافت کند و مقدار مشتق تابع  $\text{sigmoid}$  در آن نقطه را بازگرداند.
۳. یک تابع به اسم  $\text{sigmatrix}$  بنویسید که ماتریس و ابعاد آن را از ورودی دریافت می کند و برای تمام اعضای ماتریس توابع نوشته شده در قسمت ۱ و ۲ را صدا می زند (توسط یک حلقه  $f\circ$ ) سپس توسط عمل جمع خروجی های ۱ و ۲ خواسته شده را تولید کنید.

بعد از پیاده سازی تابع ماتریس  $M$  را که توسط معادلات ۱.۲ توصیف شده است به همراه تعداد سطرها و ستون های آن به تابع نوشته شده بدهید و خروجی را مشاهده کنید.

$$M = \begin{bmatrix} 1 & 0 & \sin(\pi/4) \\ 0 & 1 & \sin(\pi/2) \\ 1 & 0 & 1 \end{bmatrix} \quad (۲.۱)$$

## ۴.۱ تمرین

۱. تابعی بنویسید که یک ماتریس و ابعاد آن را از ورودی دریافت کند و  $\sum_{i=1}^n \sum_{j=1}^m \tanh(A_{ij})$  را باز گرداند. نمایش دهنده ماتریس ورودی تابع  $\tanh(x)$  (تابع  $\tanh(x)$  به صورت زیر تعریف می شود).

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (۳.۱)$$

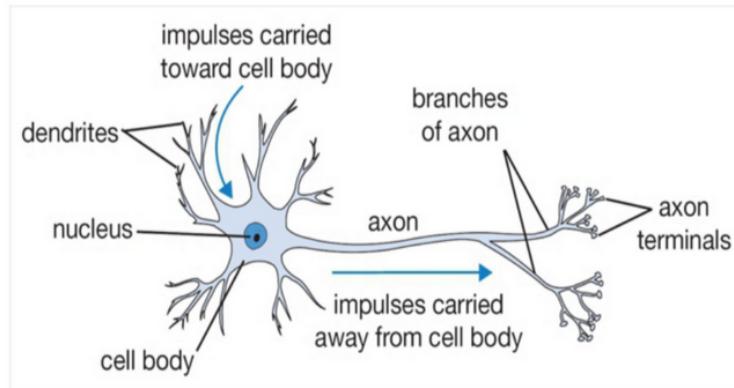
# آزمایش دوم(پیاده سازی پرسپترون)

## ۱.۲ پیش گزارش

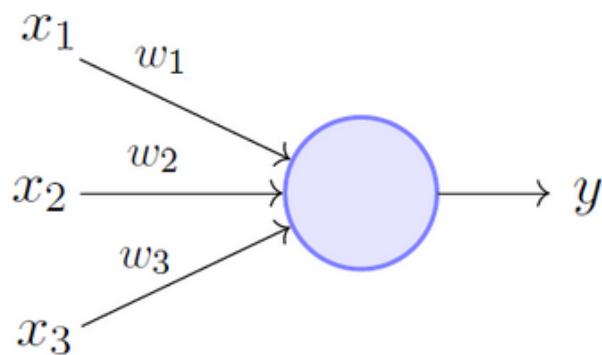
۱. نقش *perceptron* در شبکه عصبی چیست؟
۲. یک مدل ساده از *perceptron* رسم کنید
۳. شبهایت مدل رسم شده در قسمت ۲ را با ساختار *perceptron* در بدن انسان بیان کنید.

## ۲.۲ مقدمه

در این آزمایش قصد داریم به پیاده سازی *perceptron* در محیط پایتون بپردازیم. ساختار شبکه عصبی با ساختار مغز انسان شباهت زیادی دارد. مغز انسان در خود تعداد بسیار زیادی از نورون‌ها را جای داده است تا اطلاعات مختلف را پردازش کرده و جهان اطراف را بشناسد. به صورت ساده، نورون‌ها در مغز انسان اطلاعات را از نورون‌های دیگر به وسیله‌ی دنдрولیدها می‌گیرند. این نورون‌ها اطلاعات ورودی را با هم جمع کرده و اگر از یک حد آستانه‌ای فراتر رود به اصلاح فعال می‌شود و این سیگنال فعال شده از طریق آکسون‌ها به نورون‌های دیگر متصل می‌شود. شکل ۱.۲ مدل کلی پرسپترون در مغز انسان را نمایش می‌دهد. با الهام گیری از مدل پرسپترون در مغز انسان، مدل پرسپترون در شبکه عصبی ایجاد شد. مدل کلی پرسپترون در شبکه عصبی در شکل ۲.۲ نمایش داده شده است. ورودی پرسپترون مجموع وزن داری از ورودی‌های اصلی شبکه است و خروجی آن یک مقدار دودویی که نشان دهنده کلاس داده ورودی است، می‌باشد.



شکل ۱.۲: مدل پرسپترون در مغز



شکل ۲.۲: مدل پرسپترون در شبکه عصبی

خروجی پرسپترون بر اساس ورودی های آن در معادلات ۱.۲ توصیف شده است.

$$y = \begin{cases} 1 & \text{if } wx > 0 \\ 0 & \text{if } wx < 0 \end{cases} \quad (1.2)$$

وزن های پرسپترون توسط الگوریتم یادگیری پرسپترون تازه سازی می شوند. در ادامه به بررسی این الگوریتم پرداخته می شود.

## ۱.۲.۲ الگوریتم یادگیری

هدف الگوریتم یادگیری پرسپترون پیدا کردن بهترین وزن های شبکه برای تشخیص درست کلاس داده های ورودی است.

الگوریتم یادگیری پرسپترون در شکل ۳.۲ نمایش داده شده است. در این الگوریتم وزن های شبکه در ابتدا با استفاده از اعداد تصادفی مقدار دهی اولیه می شوند. سپس تا هنگامی وزن ها همگرا نشده اند، وزن ها بر اساس تفات بین کلاس پیش بینی شده توسط پرسپترون و کلاس اصلی داده ورودی تازه سازی می شوند.

- Given  $P$  training pairs  $\{x_1, d_1, x_2, d_2, \dots, x_p, d_p\}$  where  $x_i$  is  $(n \times 1)$ ,  $d_i$  is  $(1 \times 1)$ ,  $i = 1, \dots, P$
  - The augmented input vectors are  $y_i = [x_i \ -1]^T$ , for  $i = 1, \dots, P$
  - In the following,  $k$  is training step and  $p$  is step counter within training cycle.
    - Choose  $\eta > 0$ ,  $\lambda = 1$ ,  $E_{max} > 0$
    - Initialized weights at small random values,  $w$  is  $(n \times 1) \times 1$
    - Initialize counters and error:  $k \leftarrow 1$ ,  $p \leftarrow 1$ ,  $E \leftarrow 0$
    - Training cycle begins here. Set  $y \leftarrow y_p$ ,  $d \leftarrow d_p$ ,  $o = f(w^T y)$  ( $f(\text{net})$  is sigmoid function)
    - Update weights  $w \leftarrow w + \frac{1}{2}\eta(d - o)(1 - o^2)y$
    - Find error:  $E \leftarrow \frac{1}{2}(d - o)^2 + E$
    - If  $p < P$  then  $p \leftarrow p + 1$ ,  $k \leftarrow k + 1$ , go to step 4, otherwise, go to step 8.
    - If  $E < E_{max}$  the training is terminated, otherwise  $E \leftarrow 0$ ,  $p \leftarrow 1$  go to step 4 for new training cycle.

### شکل ۳.۲: الگوریتم یادگیری پرسپترون

۳.۲ شرح آزمایش

در این قسمت به پیاده سازی پرسپکترون در محیط پایتون پرداخته می شود.

### ۱.۳.۲ پیاده سازی پرسپیترون

برای پیاده سازی پرسپیکرون یک کلاس کلی به پرسپیکرون ایجاد کنید و هر یک از اعمال زیر را به صورت یک تابع در کلاس

پنویسند:

۱. تابعی بنویسید که وزن و  $threshold$  را به صورت رندم مقدار دهی اولیه کنید.

۲. تابع $y$  بنویسید که $x$  را معرف هر داده است بگرد و خروجی شکه را محاسبه کند

۳. تابع $y$  بنویسید برای یادگیری شیوه مطابق شکل ۲.۲ بنویسید و روای این تابع $x$  (داده ورودی) و $y$  (خروجی واقعی)

برای ورودی شبکه) و نرخ پادگیری است

داده ورودی ۲.۳.۲

پایی چک کردن عملکرد الگوریتم از سلطان موجود در *sklearn* استفاده کنید.

این دیتاست شامل ۵۶۹ داده و ۳۰ متغیر است. خروجی هر داده به صورت ۰ و ۱ است که نشان دهنده وجود و یا عدم وجود سرطان در شخص مورد نظر است. قسمتی از داده را برای یادگیری و قسمت دیگر را برای تست الگوریتم در نظر بگیرید.

برای عملکرد بهتر الگوریتم بهتر است ابتدا داده ورودی  $scale$  کنید و سپس به صورت ورودی به شبکه داده شود.

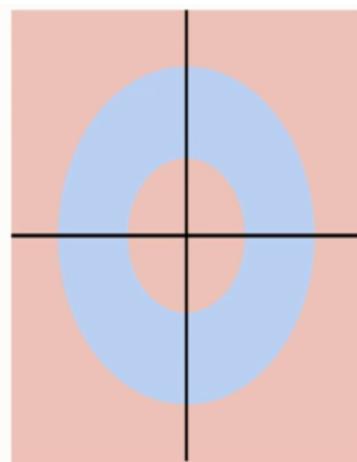
### ۳.۳.۲ بهبود مدل

با تغییر دادن نرخ یادگیری و نسبت داده یادگیری و تست سعی کنید عملکرد مدل را بهبود ببخشید(دقت به عنوان معیار عملکرد الگوریتم در نظر گرفته می شود)

### ۴.۲ تمرین

۱. در الگوریتم پیاده سازی شده هیچ مقدار بایاسی برای پرسپترون در نظر گرفته نشده است . الگوریتم پیاده سازی شده را با درنظر گرفتن بایاس دوباره پیاده سازی کنید.(برای در نظر گرفتن بایاس کافی است ۱ بعد به ماتریس ورودی خود اضافه کنید و مقدار آن را برابر ۱ قرار دهید). اضافه کردن بایاس چه تاثیری بر روی عملکرد الگوریتم می گذارد؟

۲. فرض کنید میخواهیم کلاس داده هایی به شکل ۴.۲ را توسط پرسپترون پیش بینی کنیم(نقاط صورتی رنگ متعلق به کلاس صفر و نقاط دیگر متعلق به کلاس ۱ در نظر گرفته می شود) عملکرد الگوریتم را پیش بینی کنید؟ آیا الگوریتم یادگیری همگرا می شود؟



شکل ۴.۲: داده ها و کلاس متناظر با آن ها

۳. کد پیاده سازی شده در آزمایشگاه را دوباره در محیط متلب پیاده سازی کنید.

۴. توابع  $XOR$ ,  $OR$ ,  $AND$  را توسط پرسپترون پیاده سازی کنید. آیا در تمام موارد نتیجه دلخواه و قابل قبول با پرسپترون حاصل می شود؟

۵. ساختار داده Iris را از سایت <https://archive.ics.uci.edu/ml/datasets/iris> دانلود کنید و ستون های داده را توضیح دهید. سعی کنید با استفاده از مدل پرسپترون مدلی طراحی کنید که مشخص کند داده ورودی

---

متغّلّق به چه نوع گلی است.

٦. با استفاده از دستور `sklearn.datasets.make_regression` یک دیتاست خطی ایجاد کنید و با تغییر لایه خروجی مدل رفتار پرسپترون را بررسی کنید و به نمایش بکشید.



# آزمایش سوم(پیاده سازی شبکه عصبی چند لایه)

## ۱.۳ پیش گزارش

۱. محدودیت های مدل پرسپترون را بیان کنید.

۲. ساختار کلی یک شبکه عصبی ۲ لایه را رسم کنید

۳. مراحل الگوریتم پس انتشار خطا را با جزئیات بیان کنید.

## ۲.۳ مقدمه

مدل در نظر گرفته شده برای پرسپترون از محدودیت های زیادی رنج می برد برای مثال تنها توابع جدایذیر خطی امکان دسته بندی توسط این نوع مدل را دارند. مدل پرسپترون حتی توانایی پیش بینی تابع ساده  $XoR$  را ندارد.

شبکه های عصبی چندلایه مدل توسعه یافته پرسپترون هستند و مشکلات مدل پرسپترون را ندارند.

در این آزمایش به پیاده سازی یک شبکه عصبی ۲ لایه پرداخته می شود.

ساختار کلی یک شبکه عصبی ۲ لایه در شکل ۱.۳ نمایش داده شده است.

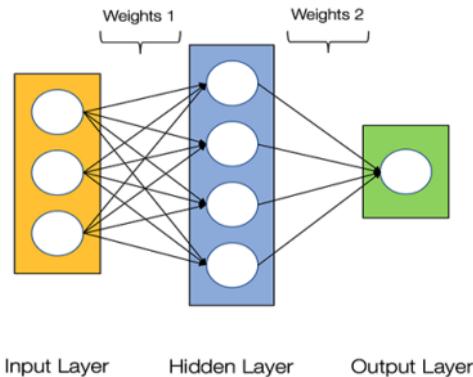
عناصر اصلی تشکیل دهنده شبکه عصبی ۲ لایه عبارتند از:

۱. لایه ورودی

۲. لایه مخفی

۳. لایه خروجی

۴. مجموعه ای از وزن ها و بایاس



شکل ۱.۳: ساختار کلی یک شبکه عصبی ۲ لایه

### ۵. توابع فعال ساز

انتخاب های مختلفی برای توابع فعال ساز یک شبکه عصبی وجود دارد. از توابع فعال ساز پرکاربرد می توان به

اشاره کرد  $tanh$  و  $sigmoid$

خروجی یک شبکه عصبی ۲ لایه به صورت معادلات ۱.۳ متناظر با تابع فعال ساز شبکه و  $W_1$  و  $W_2$  به ترتیب متناظر با وزن های لایه اول و دوم و  $b_1$  و  $b_2$  به ترتیب متناظر با بایاس لایه اول و دوم در نظر گرفته شده است.

$$\hat{y}(x) = \sigma(W_2\sigma(w_1x + b_1) + b_2) \quad (1.3)$$

برای سنجش عملکرد شبکه یک تابع هزینه در نظر گرفته می شود این تابع معمولاً به صورت توان ۲ خطأ است و به صورت زیر تعریف می شود.

$$cost - function = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.3)$$

هدف شبکه عصبی ۲ لایه پیدا کردن مقدار وزن ها و بایاس است به گونه ای که تابع هزینه در نظر گرفته شده به کمترین مقدار خود برسد . به پیدا کردن مقادیر مناسب وزن ها و بایاس های در شبکه عصبی برای بهبود عملکرد آن ، یادگیری شبکه گفته می شود.

هر مرحله یادگیری شامل مراحل زیر است:

۱. محاسبه خروجی شبکه به ازای داده ورودی به این مرحله *forward pass* گفته می شود

۲. تازه سازی وزن ها و بایاس های شبکه بر اساس خطا. به این مرحله پس انتشار خطا گفته می شود

در ادامه به مروری کوتاه بر روی الگوریتم پس انتشار خطا پرداخته می شود.

### ۱.۲.۳ الگوریتم پس انتشار خطا

مطابق با الگوریتم پس انتشار خطا برای پیدا کردن بهترین وزن ها که منجر به کمینه شدن تابع هزینه می شوند باید در خلاف جهت مشتق تابع هزینه بر حسب وزن ها حرکت کرد.

اگر تابع هزینه را برابر با توان ۲ خطا در نظر گرفته شود (معادله ۲.۳) ، به دلیل اینکه وزن ها در تابع هزینه به صورت مستقیم قرار ندارند ، نمی توان به صورت مستقیم از تابع هزینه بر حسب وزن مشتق گرفت . برای مشتق گرفتن از تابع هزینه بر حسب وزن می توان از قانون زنجیره ای استفاده کرد برای مثال برای تازه سازی وزن های لایه اول شبکه می توان به صورت زیر عمل کرد:

$$\begin{aligned} Loss(y, \hat{y}) &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ \frac{\partial Loss(y, \hat{y})}{\partial W} &= \frac{\partial Loss(y, \hat{y})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial W} \quad \text{where } z = Wx + b \\ &= 2(y_i - \hat{y}_i) * \text{derivative of sigmoid function} * x \\ &= 2(y_i - \hat{y}_i) * z(1-z) * x \end{aligned}$$

وزن های لایه دوم نیز به روش مشابه تازه سازی می شوند.

### ۳.۳ شرح آزمایش

در این قسمت به پیاده سازی یک شبکه عصبی ۲ لایه در محیط پایتون پرداخته می شود.

### ۱.۳.۳ پیاده سازی شبکه عصبی ۲ لایه

برای پیاده سازی شبکه عصبی ۲ لایه یک کلاس به اسم *deep network* ایجاد کنید و مراحل زیر را دنبال کنید.

۱. یک تابع در کلاس بنویسید که مقدار دهی اولیه وزن ها و بایاس های لایه اول و دوم را انجام دهد(مقدار دهی اولیه را می توانید در تابع سازنده کلاس نیز انجام دهید)

۲. تابعی بنویسید که خروجی لایه اول شبکه را محاسبه کند(تابع *sigmoid* را به عنوان تابع فعالساز در نظر بگیرید)

۳. یک تابع برای محاسبه *sigmoid* پیاده سازی کنید.
۴. یک تابع برای محاسبه مشتق تابع *sigmoid* پیاده سازی کنید.
۵. یک تابع برای محاسبه ضرب داخلی وزن ها و ورودی شبکه بنویسید.
۶. با استفاده از توابعی که پیاده سازی کرده اید، تابعی بنویسید که وزن های لایه اول و دوم را تازه سازی کند.

### ۲.۳.۳ تست مدل

همان طور که می دانید تابع *xor* را نمی توان با یک پرسپترون ساده تخمین زد. در این قسمت برای ارزیابی مدل خود از تابع *xor* استفاده کنید.

ورودی و خروجی های تابع *xor* در شکل ۲.۳ نمایش داده شده است. قسمتی از داده را برای یادگیری و قسمتی دیگر را برای تست در نظر بگیرید. ورودی و خروجی ها را با استفاده از آرایه ها به مدل پیاده سازی شده خود دهید. دقت مدل پیاده سازی شده بر روی داده تست و یادگیری چه قدر است؟

X1	X2	X3	Y
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

شکل ۲.۳: تابع *xor*

### ۴.۳ تمرین

۱. مدل شبکه عصبی ۲ لایه پیاده سازی شده در محیط پایتون را دوباره در محیط متلب تکرار کنید.
۲. داده های فایل پیوست مربوط به *California – house – prediction* را در نظر بگیرید.
- (آ) دسته بندی مناسب و پارامترهای مناسب برای دسته بندی داده ها را پیدا کنید.
- (ب) سعی کنید با نرم افزار متلب نیز شبکه بیش از ۳ لایه پیاده سازی کرده و نتایج را بررسی کنید. تاثیر اقزایش لایه ها را بررسی کنید.
- (ج) تا چه حد میتوان این داده ها را بهبود داد.

(د) Overfit را بررسی کرده و تغییرات داده های ولیدیشن و تست را نسبت به خطای داده ها به دست آورید.

(ه) افزایش لایه ها در شبکه چه تاثیری میگذارد؟

۳. با استفاده از کتابخانه TensorFlow یک شبکه عصبی چندلایه ایجاد کنید و تاثیر افزایش لایه ها را بررسی کنید.

۴. در مدلی که در پرسشن قبلاً ساخته اید لایه batch normalization را اضافه کنید و تاثیر آنرا بررسی کنید.

۵. محل قرارگیری لایه batch normalization را نسبت به هسته خطی وتابع فعالسازی بررسی کنید و پاسخ خود را توجیه کنید.



# آزمایش چهارم (پیاده سازی الگوریتم $k - mean$ )

## ۱.۴ پیش گزارش

۱. مفهوم الگوریتم با ناظارت و بدون ناظارت را بیان کند.
۲. مزیت های الگوریتم های با ناظارت را نسبت به الگوریتم های بدون ناظارت و بر عکس بیان کنید.
۳. الگوریتم دسته بندی  $k - mean$  را با جزئیات توضیح دهید.
۴. کاربرد های الگوریتم  $k - mean$  را بنویسید.

## ۲.۴ مقدمه

در آزمایشات گذشته با استفاده از شبکه عصبی چند لایه به دسته بندی داده های با برچسب پرداخته شد. در این آزمایش قصد داریم به دسته بندی داده های بدون برچسب بپردازیم. به یادگیری بر روی داده های بدون برچسب و تلاش برای پیدا کردن الگوهای نهفته در آن ها یادگیری بدون ناظارت می گویند. یکی از پرکاربردترین الگوریتم ها در یادگیری بدون ناظارت الگوریتم  $k - mean$  است. در ادامه به بررسی روند اجرای الگوریتم  $k - mean$  پرداخته می شود.

## ۱.۲.۴ الگوریتم $k - mean$

الگوریتم  $k - mean$  یک الگوریتم بازگشتی است که با یک فرض اولیه درباره مراکز دسته ها آغاز می شود. در هر مرحله از اجرای الگوریتم مراحل زیر اجرا می شود:

۱. پیدا کردن دسته متناظر با تمام نقاط

در مرحله اول اجرای الگوریتم فاصله تمام نقاط تا مراکز دسته ها محاسبه می شود و سپس هر داده متعلق به دسته ای که کمترین فاصله با آن را دارد می شود.

بعد از پیدا کردن دسته های متناظر با هر داده ، در مرحله دوم میانگین داده های متعلق به یک دسته به عنوان مرکز دسته در نظر گرفته می شود.

مراحل ۱ و ۲ تا زمانی که مراکز دسته ها تغییر نکند و یا تغییرات خیلی کمی داشته باشند ادامه می یابد.  
بعد از ثابت شدن مراکز دسته ها الگوریتم همگرا می شود.  
روند کلی اجرای الگوریتم در شکل ۱.۴ نمایش داده شده است

### K-MEANS( $P, k$ )

Input: a dataset of points  $P = \{p_1, \dots, p_n\}$ , a number of clusters  $k$   
Output: centers  $\{c_1, \dots, c_k\}$  implicitly dividing  $P$  into  $k$  clusters

- 1 choose  $k$  initial centers  $C = \{c_1, \dots, c_k\}$
- 2 **while** stopping criterion has not been met
- 3     **do** ▷ assignment step:
- 4         **for**  $i = 1, \dots, N$
- 5             **do** find closest center  $c_k \in C$  to instance  $p_i$
- 6                 assign instance  $p_i$  to set  $C_k$
- 7     ▷ update step:
- 8         **for**  $i = 1, \dots, k$
- 9             **do** set  $c_i$  to be the center of mass of all points in  $C_i$

شکل ۱.۴: مراحل اجرای الگوریتم  $k - mean$

در زمان تست دسته ای که به داده ورودی کمترین فاصله را دارد به داده ورودی نسبت داده می شود.

## ۳.۴ شرح آزمایش

در این قسمت ابتدا به پیاده سازی الگوریتم  $k - mean$  پرداخته می شود . بعد از پیاده سازی الگوریتم به تست و بررسی کاربردها و معایب الگوریتم پرداخته می شود.

## ۱.۳.۴ پیاده سازی الگوریتم

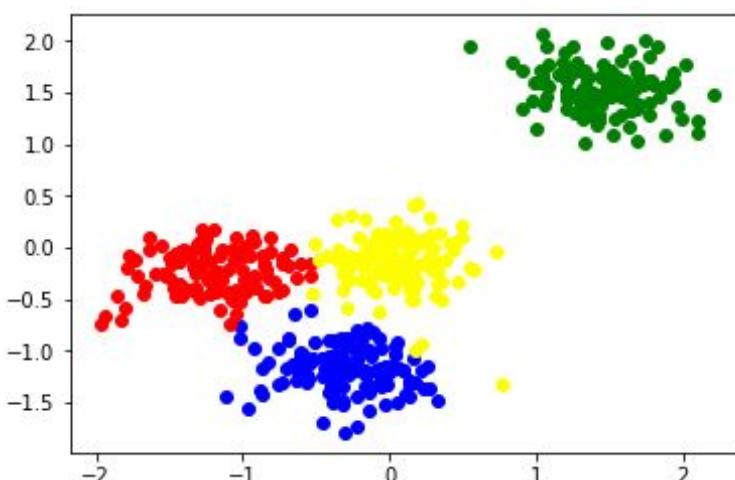
در شکل ۱.۴ مراحل اجرای الگوریتم  $k - mean$  نمایش داده شده است. با توجه به مراحل اجرای الگوریتم به پیاده سازی الگوریتم بپردازید. دقت کنید پیاده سازی باید در محیط پایتون انجام شود و کد پیاده سازی شده باید مستقل از تعداد کلاس و نوع داده ورودی باشد، به بیانی دیگر کد پیاده سازی شده باید به ازای هر داده ورودی و هر تعداد کلاس کارآیی داشته باشد(تعداد کلاس ها و داده را از ورودی دریافت کنید).

برای پیاده سازی الگوریتم مراحل زیر را انجام دهید:

۱. تابعی بنویسید که مراکز دسته ها را با استفاده از اعداد تصادفی مقدار دهی اولیه کند. ورودی این الگوریتم تعداد کلاس ها و خروجی آن مقدار اولیه مراکز دسته ها است.
۲. تابعی بنویسید که یک داده و مراکز دسته ها را به عنوان ورودی بگیرد و اندیس نزدیکترین دسته به داده ورودی در آرایه را برگرداند.
۳. تابعی بنویسید که آرایه ای از داده ها را از ورودی دریافت کند و سپس میانگین داده های ورودی را باز گردداند.
۴. با استفاده از توابع پیاده سازی شده الگوریتم  $k - mean$  را پیاده سازی کنید

## ۲.۳.۴ تست الگوریتم

مجموعه ای از داده ها در اختیاراتان قرار می گیرد. الگوریتم را بر روی مجموعه داده هایی که در اختیاراتان قرار گرفته است با تعداد دسته های برابر با ۲ و ۳ و ۴ اجرا کنید و دسته های خروجی را مشاهده کنید. دسته های خروجی را به رنگ های مختلف مانند (شکل ۲.۴) رنگ آمیزی کنید.



شکل ۲.۴: دسته بندی انجام شده توسط الگوریتم  $k - mean$

### ۳.۳.۴ ارزیابی الگوریتم

الگوریتم  $k - mean$  در دسته الگوریتم های بدون نظارت قرار میگیرد.

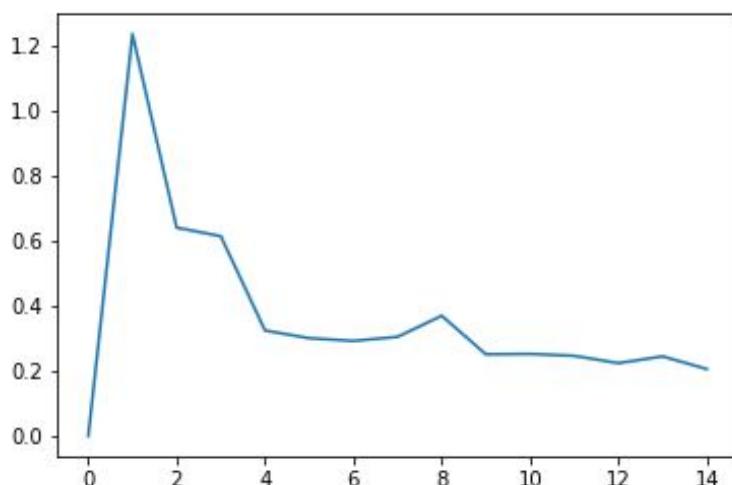
تعداد دسته ها و شرایط اولیه برای مراکز دسته ها در عملکرد این الگوریتم تاثیر فراوانی می گذارد برای ارزیابی الگوریتم  $k - mean$  مراحل زیر را دنبال کنید:

۱. تابعی بنویسید که نقاط و دسته بندی متناظر با آن ها را به از ورودی بگیرد، سپس برای هر دسته میانگین فاصله

نقاط متعلق به دسته مورد نظر را تا مرکز دسته محاسبه کند به این عدد خطای هر دسته گفته می شود.

۲. تابعی بنویسید که نقاط و دسته بندی متناظر با آن ها را به از ورودی بگیرد و میانگین خطای دسته ها را به عنوان خطای الگوریتم محاسبه کند و به عنوان خروجی باز گرداند.

بعد از پیاده سازی تابع بر روی مجموعه داده هایی که در اختیاراتان قرار داده شده است، الگوریتم  $k - mean$  را به ازای دسته های ۱ تا ۱۵ اجرا کنید و در هر مرحله خطای الگوریتم را محاسبه کنید. در انتها نمودار خطای دسته ها بر حسب تعداد دسته ها را مانند شکل ۳.۴ رسم کنید.



شکل ۳.۴: نمودار خطای خروجی الگوریتم بر حسب تعداد دسته (محور افقی تعداد دسته ها و محور عمودی خطای تشان می دهد).

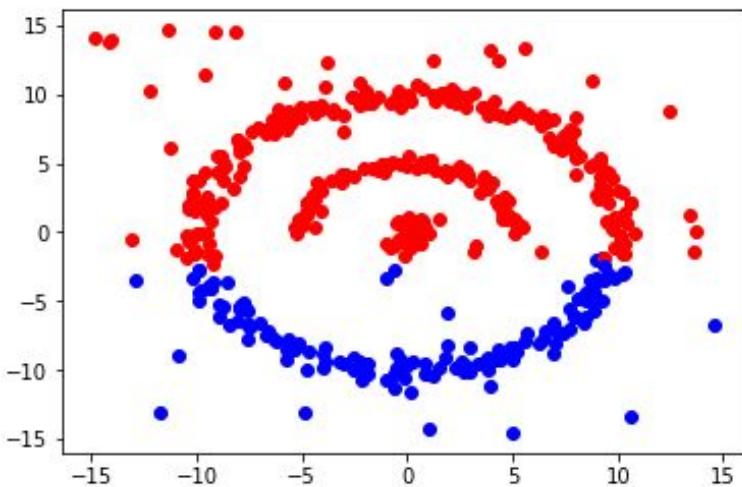
### ۴.۳.۴ محدودیت های $kmean$

در این قسمت به بررسی محدودیت های الگوریتم  $k - mean$  پرداخته می شود.

مجموعه ای از داده ها در اختیاراتان قرار داده می شود. الگوریتم  $k - mean$  پیاده سازی شده را بر روی مجموعه داده ای که در اختیاراتان قرار داده شده با تعداد دسته های ۲، ۳، ۴ اجرا کنید و دسته های مختلف را رنگ آمیزی کنید و

نمودار خطای بر حسب تعداد دسته ها را رسم کنید.

در صورت پیاده سازی صحیح الگوریتم خروجی مشابه تصویر ۴.۴ خواهد بود.



شکل ۴.۴: محدودیت های  $kmean$  (محورها متناظر با ویژگی های داده های ورودی است)

## ۴.۴ استفاده از کتابخانه های آماده

در این قسمت با استفاده از کتابخانه `sklearn` به بررسی های پارامترهای موثر در عملکرد الگوریتم  $k$ -mean می پردازیم.

برای تعریف یک کلاس از نوع `K-mean cluster` از دستور زیر استفاده می کنیم:

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init='warn', max_iter=300,  
tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='lloyd')
```

## ۵.۴ کاهش حجم عکس به وسیله $k - mean$

یکی از کاربردهای الگوریتم  $k - mean$  کم کردن حجم عکس است.

یک عکس در اختیاراتن قرار داده می شود. در این قسمت قصد داریم با استفاده از الگوریتم  $k - mean$  پیاده سازی شده به کم کردن حجم عکس بپردازیم.

برای کاهش حجم عکس مراحل زیر را دنبال کنید:

۱. با استفاده از دستور `imread` عکس (`filename`) را از پوشه مجموعه داده (این پوشه در اختیاراتن)

قرار داده می شود) و ارد محیط پایتون کنید (با بارگذاری عکس متغیر مربوط به آن در محیط پایتون یک آرایه ۳

بعدی با تعداد سطرها و ستون های ۸۰۰ خواهد بود. سطرها و ستون ها متناظر با مکان پیکسل ها خواهد بود و هر بعد آرایه متناظر با مقادیر  $R$  و  $G$  و  $B$  است).

۲. برای استفاده از الگوریتم  $mean - k$  که پیاده سازی کرده اید ابتدا آرایه<sup>۳</sup> بعدی عکس را به یک آرایه ۲ بعدی تبدیل کنید درآرایه دو بعدی هر سطر متناظر با یک پیکسل خواهد بود.

۳. الگوریتم  $mean - k$  را با تعداد کلاس برابر با ۱۶ اجرا کنید سپس مقدار هر پیکسل را با مقدار مرکز دسته متناظر جایگذاری کنید . عکس خروجی را رسم کنید و با عکس اولیه مقایسه کنید

## ۶.۴ استفاده از کتابخانه های آماده

در این قسمت با استفاده از کتابخانه `sklearn` به بررسی های پردازی پارامترهای موثر در عملکرد الگوریتم  $k$ -mean می پردازیم.  
برای تعریف یک کلاس از نوع `K-mean cluster` از دستور زیر استفاده می کنیم:

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init='warn', max_iter=300,  
tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='lloyd')
```

۱. پارامترهای ورودی تابع را بررسی کرده و تاثیر هر کدام را بر عملکرد مدل توضیح دهید.
۲. داده های استفاده شده در قسمت قبل را به کمک تابع معرفی شده نیز دسته بندی کرده و عملکرد این تابع را با کد خود مقایسه کنید.

## ۷.۴ تمرین

۱. با توجه به نمودار خطای بر حسب تعداد کلاس ها که در قسمت ارزیابی الگوریتم رسم کرده اید حالت بهینه الگوریتم در چه تعداد کلاس رخ می دهد؟

۲. به نظر شما چرا الگوریتم توانایی دسته بندی صحیح داده ها را در قسمت محدودیت های  $mean - k$  نداشت؟  
برای بهبود الگوریتم چه پیشنهادی دارید؟

۳. الگوریتم  $mean - k$  پیاده سازی شده را در محیط متلب مجددا پیاده سازی کنید.

۴. دسته بندی داده های قسمت اول آزمایش را با استفاده از دستور  $kmean$  در متلب (دستور آماده متلب) تکرار کنید.

۵. فرض کنید شمادرای یک فروشگاه سوپر مارکت هستید و از طریق کارت های عضویت ، اطلاعات اصلی در مورد مشتریان خود مانند شناسه مشتری ، سن ، جنسیت ، درآمد سالانه آن ها در اختیار دارید. نمره خرید چیزی است که شما بر اساس پارامترهای تعریف شده مانند رفتار مشتری و خرید به مشتری اختصاص می دهید. با استفاده از  $k - mean$  شبکه ای را طراحی کنید تا مشتری هایی را که راحت تر می توان ترغیب به خرید کرد شناسایی کنید.

داده این مساله را می توانید از <https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python> دانلود کنید.

۶. با استفاده از تابع `lstdinlinesklearn.cluster.KMeans` الگوریتم این آزمایش را مورد بررسی قرار دهید و تاثیر های مختلف را بر نحوه عملکرد آن بسنجید.



# آزمایش پنجم(پیاده سازی شبکه عصبی $RBF$ )

## ۱.۵ پیش گزارش

۱. ساختار یک شبکه عصبی  $RBF$  را رسم کنید.

۲. شبکه عصبی چند لایه چه تفاوت هایی با شبکه عصبی  $RBF$  چیست؟

## ۲.۵ مقدمه

یکی از کاربردهای شبکه عصبی در مساله رگرسیون است. رگرسیون کاربرد های زیادی در پیش بینی بورس و پیش بینی توابع ناشناخته شده دارد. در این آزمایش با استفاده از شبکه عصبی  $RBF$  به تخمین یکتابع پرداخته می شود. شبکه های عصبی  $RBF$  از شناخته شده ترین و پرکاربرد ترین شبکه های عصبی برای مساله رگرسیون هستند. اگر تعدادی داده از رفتار یکتابع ناشناخته وجود داشته باشد شبکه عصبی  $RBF$  با استفاده از ترکیب چندتابع گوسین رفتار تابع ناشناخته شده را با خطای کم پیش بینی خواهد کرد.

## ۱.۶.۵ ساختار شبکه

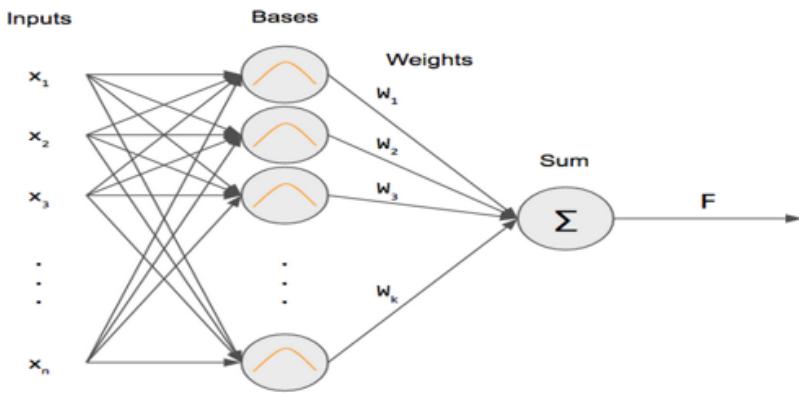
ساختار یک شبکه  $RBF$  تا حد زیادی به شبکه عصبی چندلایه<sup>۱</sup> شباهت دارد. همان طور که در شکل ۱.۵ مشاهده می کنید. شبکه  $RBF$  نیز همانند شبکه تمام پیوسته از<sup>۳</sup> لایه تشکیل شده است.

شبکه عصبی  $RBF$  دارای ساختار زیر است:

۱. لایه اول ، لایه ورودی است

---

<sup>1</sup> fully – connected



شکل ۱.۵: ساختار کلی شبکه RBF

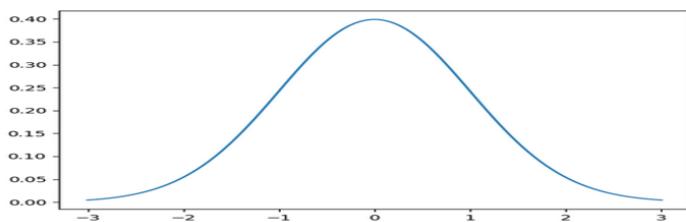
۲. لایه دوم شامل یک یا چند لایه مخفی است

۳. در نهایت لایه آخر که لایه خروجی است.

تفاوت شبکه عصبی RBF با شبکه عصبی چند لایه در توابع استفاده شده در لایه مخفی است. در شبکه عصبی از توابع گوسین به عنوان توابع فعالساز برای لایه های مخفی استفاده می شود. در ادامه مروری کوتاه بر توابع گوسین خواهیم داشت.

## ۲.۵ تابع گوسین

تابع گوسین و یا توابع نرمال یکی از مهم ترین توابع در آمار محاسبه می شوند. شکل کلی یک تابع گوسین در تصویر ۲.۵ نمایش داده شده است



شکل ۲.۵: تابع نرمال

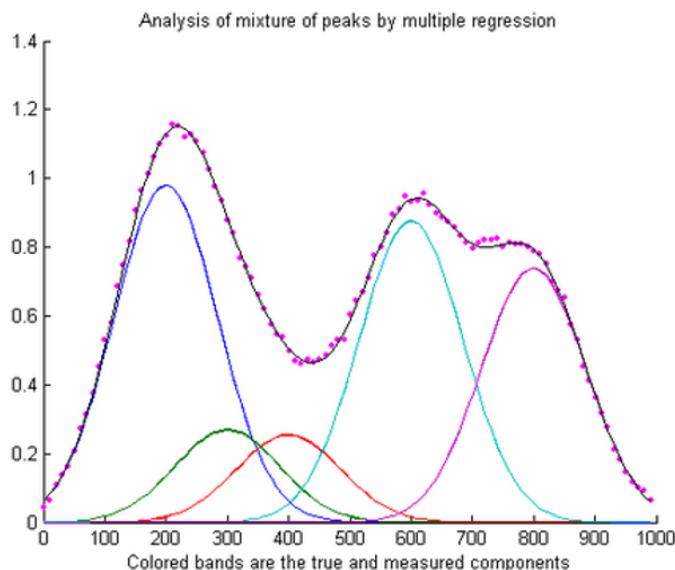
یک تابع گوسین توسط مقدار میانگین و انحراف معیار توصیف می شود. رابطه توصیف کننده تابع گوسین به صورت زیر است:

$$N(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (1.5)$$

در ادامه به بررسی کاربرد توابع گوسین در ساختار شبکه عصبی RBF پرداخته می شود.

### ۳.۲.۵ توابع گوسین و $RBF$

در شبکه عصبی  $RBF$  خروجی به صورت مجموعی از تعدادی تابع گوسین است. در شکل شماره ۳.۵ تعدادی تابع گوسین را مشاهده می کنید. توابع گوسین مختلف به وسیله رنگ های مختلف مشخص شده اند. تفاوت این توابع گوسین در وزن و انحراف معیار و مقدار میانگین آن ها است. اگر مجموع توابع گوسین نمایش داده شده را در شکل شماره ۳.۵ در نظر بگیرید یک تابع پیوسته به دست می آید. اگر مقدار میانگین و انحراف معیار توابع گوسین در نظر گرفته شده تغییر کند، مقدار تابع پیوسته ناشی از مجموع آن ها نیز تغییر خواهد کرد. با توجه به این موضوع در یک شبکه عصبی  $RBF$  با استفاده از تعدادی داده از رفتار یک تابع، مقادیر میانگین و انحراف معیار مناسب برای هریک از توابع گوسین به دست خواهد آمد. مقادیر انحراف معیار و میانگین به گونه ای به دست خواهند آمد که مجموع توابع گوسین توصیف مناسبی از تابع بر حسب داده های در اختیار ارایه دهد. بعد از



شکل ۳.۵: تخمین تابع پیوسته توسط مجموع توابع گوسین

به دست آوردن مقادیر میانگین و انحراف معیار برای هریک از توابع گوسین می توان مقدار تابع را در هر نقطه دلخواه پیش بینی کرد. برای به دست آوردن مقدار میانگین برای توابع گوسین در نظر گرفته شده روش های مختلفی وجود دارد یکی از این روش ها استفاده از الگوریتم  $k$  میانگین بر روی داده های ورودی برای مقداردهی میانگین های توابع گوسین است.

بعد از مشخص کردن میانگین توابع گوسین استفاده شده برای کامل شدن تابع باید مقدار انحراف معیار هر یک از توابع نیز مشخص شود. برای مشخص کردن انحراف معیار نیز روش های مختلفی وجود دارد در یکی از این روش ها انحراف معیار برابر با رابطه زیر در نظر گرفته می شود در این رابطه  $d$  بیشترین فاصله بین دو دسته  $m_1$  و  $m_2$  مشخص کننده

تعداد دسته ها است.

$$\sigma = \frac{d}{\sqrt{2m}} \quad (2.5)$$

مانند دیگر انواع شبکه های عصبی وزن های خروجی شبکه  $RBF$  نیز بر حسب خطا تازه سازی می شوند. در ادامه به بررسی قانون تازه سازی وزن ها در شبکه  $RBF$  پرداخته می شود.

#### ۴.۲.۵ قانون تازه سازی وزن های شبکه

خروجی کلی یک شبکه عصبی  $RBF$  به صورت زیر است:

$$F(x) = \sum_{j=1}^k w_j \phi_j(x, c_j) + b \quad (3.5)$$

در رابطه بالا  $\phi$  نشان دهنده توابع گوسین است و  $w$  وزن متناظر با هریک از توابع گوسین را نمایش می دهد. پارامتر  $b$  نیز به عنوان بایاس شبکه در نظر گرفته می شود.

همان طور که از رابطه بالا مشخص است مانند بقیه شبکه های عصبی خروجی شبکه عصبی  $RBF$  نیز خروجی به صورت مجموع وزن دار تعدادی توابع گوسین خواهد بود. در قسمت قبل روش هایی برای پیدا کردن مقادیر میانگین و انحراف معیار هریک از توابع گوسین بیان شد. برای کامل کردن شبکه باید مقادیر وزن های شبکه نیز تعیین شوند. مانند بقیه شبکه های عصبی برای تازه سازی وزن های هر یک از توابع گوسین از پس انتشار خطا استفاده می شود. برای آشنایی بیشتر با الگوریتم پس انتشار خطا می توانید به *lecture* های درس مراجعه کنید.

### ۳.۵ شرح آزمایش

در این مرحله به پیاده سازی یک شبکه عصبی  $RBF$  در محیط پایتون پرداخته می شود. در انتهای این شبکه عصبی طراحی شده قصد داریم به پیش بینی یکتابع بپردازیم.

## ۱.۳.۵ پیاده سازی شبکه

برای پیاده سازی شبکه  $RBF$  مراحل زیر را دنبال کنید:

۱. برای اولین قدم ابتدا یک تابع برای تولید تابع نمایی بنویسید این تابع یک نقطه و انحراف معیار و میانگین تابع نمایی را از ورودی می گیرد و مقدار تابع نمایی در نقطه ورودی را باز می گرداند.

۲. در قدم بعدی تابع  $k$  میانگین را برای مقدار دهی اولیه مقادیر میانگین توابع گوسین بنویسید . ورودی این تابع داده های ورودی و تعداد دسته ها است. برای این مرحله می توانید از تابع  $k$  میانگین نوشته شده در جلسه گذشته استفاده کنید. در این تابع باید مقادیر میانگین و انحراف معیار هر یک از توابع گوسین نیز مشخص شود.

۳. یک کلاس  $RBF$  ایجاد کنید این کلاس تعداد توابع  $RBF$  و نرخ یادگیری را از ورودی دریافت می کند.در تابع سازنده این کلاس وزن ها و بایاس را با استفاده از اعداد تصادفی مقدار دهی اولیه کنید. در این کلاس توابع زیر را پیاده سازی کنید.

(آ) در این کلاس یک تابع به اسم  $fit$  تعریف کنید . وظیفه این تابع مشخص کردن مقادیر میانگین و انحراف معیار برای توابع گوسین است. با توجه به وظیفه این تابع، برای پیاده سازی این تابع کافی است که تابع  $k$  میانگین را که در مرحله پیش پیاده سازی کرده اید صدا بزنید. با توجه به اینکه مقادیر میانگین و انحراف معیار را در تابع  $fit$  مشخص کرده اید . برای کامل کردن شبکه باید به پیاده سازی پس انتشار خطا برای تازه سازی وزن های خروجی بپردازید.

(ب) یک تابع به اسم  $train$  در کلاس  $RBF$  تعیین کنید . وظیفه این تابع پیاده سازی قانون پس انتشار خطا است. برای پیاده سازی الگوریتم پس انتشار خطا می توانید از توابع نوشته شده در جلسات گذشته نیز استفاده کنید . قسمت اصلی کد این تابع در تصویر شماره ۴.۵ نمایش داده شده است.

```
for i in range(X.shape[0]):  
    # forward pass  
    a = np.array([self.rbf(X[i], c, s) for c, s, in zip(self.centers, self.std  
    F = a.T.dot(self.w) + self.b  
  
    loss = (y[i] - F).flatten() ** 2  
    print('Loss: {:.2f}'.format(loss[0]))  
  
    # backward pass  
    error = -(y[i] - F).flatten()  
  
    # online update  
    self.w = self.w - self.lr * a * error  
    self.b = self.b - self.lr * error
```

شکل ۴.۵: الگوریتم پس انتشار خطا

(ج) در نهایت یک تابع برای پیش بینی کردن خروجی تابع تخمین زده شده به ورودی دلخواه بنویسید. برای نوشتن این تابع کافی است رابطه زیر را در تابع پیاده سازی کنید. دقت کنید مقدار  $k$  به عنوان ورودی در کلاس مشخص شده است. توابع گوسین را نیز در تابع  $fit$  محاسبه کرده اید.

$$F(x) = \sum_{j=1}^k w_j \phi_j(x, c_j) + b \quad (4.5)$$

## ۴.۳.۵ تست شبکه

برای بررسی عملکرد شبکه طراحی شده تعداد ۱۰۰ نمونه از یک تابع سینوس را که به نویز سفید آغشته شده است را به عنوان داده ورودی به شبکه بدھید. چگونگی تولید ۱۰۰ نمونه داده سینوسی و افزودن نویز به این داده ها در شکل شماره نمایش داده شده است.

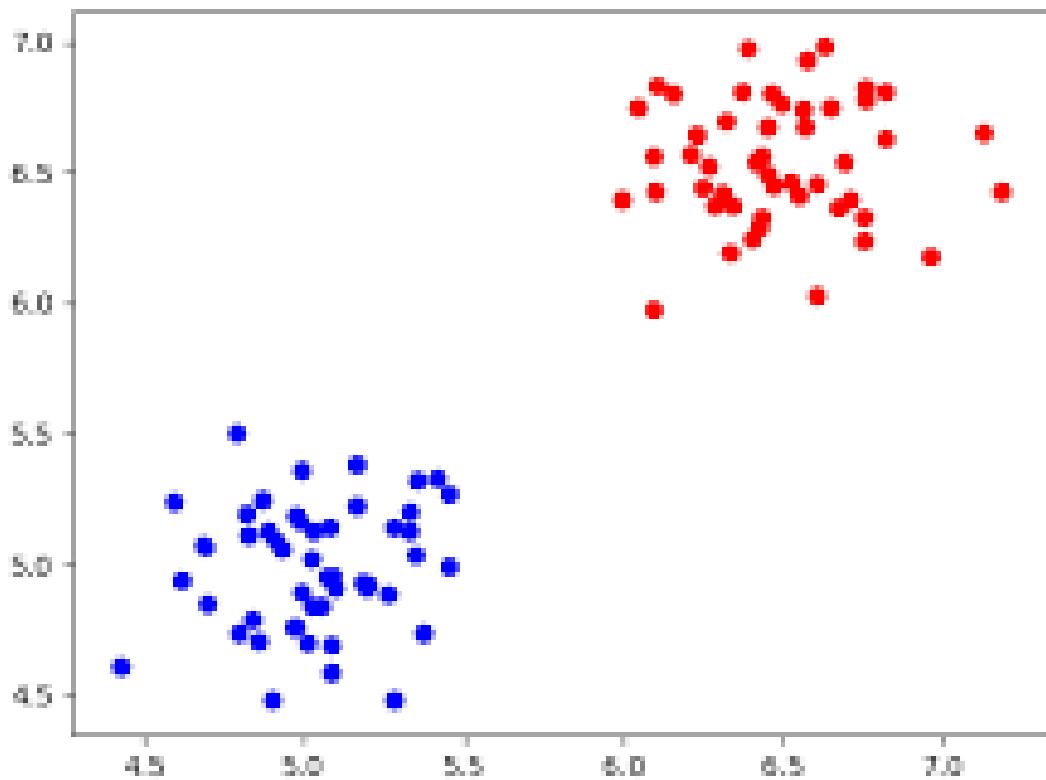
```
NUM_SAMPLES = 100
X = np.random.uniform(0., 1., NUM_SAMPLES)
X = np.sort(X, axis=0)
noise = np.random.uniform(-0.1, 0.1, NUM_SAMPLES)
y = np.sin(2 * np.pi * X) + noise
```

شکل ۴.۵: تولید داده تست

بعد از دادن ورودی به شبکه و صدا زدن تابع متناظر با یادگیری وزن های شبکه، خروجی شبکه را برای ۱۰۰ نمونه بعدی تابع  $\sin$  به دست آورید. نمودار خروجی اصلی و خروجی پیش بینی شبکه را رسم کنید.

## ۴.۵ استفاده از Tensorflow

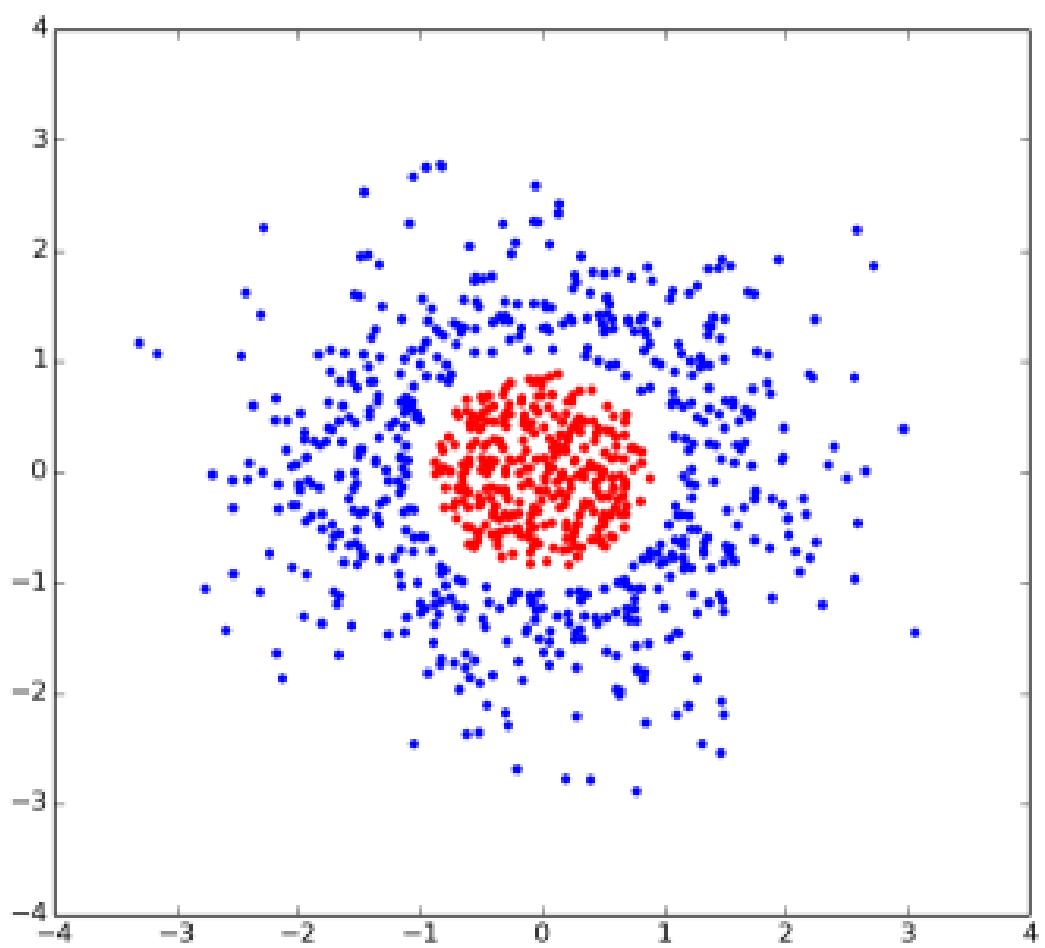
در این قسمت شبکه RBF را به کمک کتابخانه Tensorflow پیاده سازی می کنیم. توجه داریم که شبکه های RBF در حقیقت یک MLP دولایه می باشند که تابع فعال ساز لایه مخفی آنها از توابع Radial Basis می باشند. برای تعریف یک MLP در Tensorflow می توان با استفاده از مدل Sequential یک لایه ورودی با توجه به بعد ورودی ها، یک لایه مخفی از نوع fully connected و یک لایه خروجی تعریف نمود. برای آموزش شبکه از روش gradient descend استفاده کرده و optimizer های مختلف را روی مدل امتحان کنید. پس از بدست آوردن نتایج مورد نیاز، به مدل یک لایه batch normalization نیز اضافه کنید. نتایج بدست آمده از مدل جدید را با حالت قبلی مقایسه کنید.



شکل ۶.۵: داده ها

## ۵.۵ تمرین

۱. شبکه عصبی پیاده سازی شده را در محیط متلب مجدداً پیاده سازی کنید.
۲. دیتاست مشابه شکل های زیر در اختیار شما قرار می گیرد.  
توسط یک شبکه  $RBF$  در محیط پایتون به دسته بندی هر دو دیتاست بپردازید. چه نوع کرنلی را انتخاب می کنید؟ چرا؟ عملکرد این شبکه را با یک شبکه تک لایه و چند لایه مقایسه کنید.
۳. پیاده سازی موجود در [این نمونه کد](#) را بررسی کنید و با استفاده از آن مدل  $RBF$  را در محیط Tensorflow به کار بگیرید و  $hyperparameter$  های مختلف ورودی آن را بررسی کنید.



شکل ۷.۵: داده ها

# آزمایش ششم (پیاده سازی شبکه عصبی (*Hopfield*

## ۱.۶ پیش گزارش

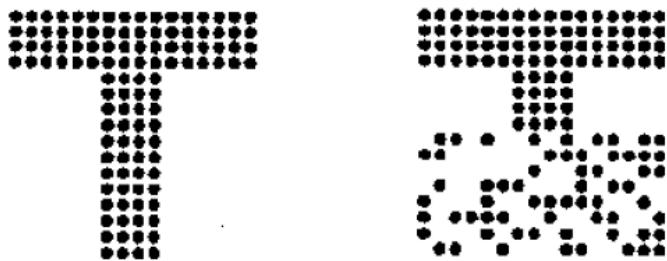
۱. با مراجعه به اینترنت در مورد حافظه‌ی انجمنی در انسان‌ها تحقیق کنید و چکیده‌ی آن را به کلاس ارائه دهید.
۲. با مراجعه به اینترنت در مورد کاربردهای حافظه‌ی انجمنی در علوم مختلف تحقیق کنید و چکیده‌ی آن را به کلاس ارائه دهید
۳. تفاوت حافظه‌ی *Autoassociative* و *Hetroassociative* را با مثالی از حافظه انسان شرح دهید.
۴. درباره‌ی دستور *newhop* در متلب و کاربرد آن تحقیق کنید. ورودی‌ها و خروجی‌های این تابع را مشخص کنید و با مثالی نحوه‌ی استفاده از این دستور را شرح دهید.
۵. در شبکه‌ی هاپفیلد به روز رسانی سنکرون و ناسنکرون الگوها چه پیامدهایی می‌تواند داشته باشد. توضیح دهید.

## ۲.۶ مقدمه

فرایند یادگیری و یا به خاطر سپردن اطلاعات همراه با عوامل و احساسات دیگر صورت می‌گیرد. بطور مثال وقتی ما نام یک شهر را به خاطر می‌سپاریم، اسم شهر با خاطرات مختلفی که از آن شهر وجود دارد ذخیره و یا فراخوانی می‌شود. قطعاً همه ما اسم دوستانمان را با تغییر جزئی در ظاهر وی از خاطر نمی‌بریم. از مهم‌ترین عملکردهای مغز ما، تعیین و فراخوانی حافظه است. حافظه ما در یک حالت انجمنی یا به صورت محتوای آدرس‌پذیر عمل می‌کند. به بیان دیگر حافظه جداگانه موجود نیست و در مجموعه خاصی از نورون‌ها قرار داده شده است. تمامی حافظه‌ها از بعضی جهات، رشتۀ‌ای از حافظه‌ها هستند. برای مثال انسان می‌تواند اشخاص را به طرق

مختلف، از جمله رنگ مو یا چشمها، شکل بینی، قد و صدا به خاطر بیاورد و این امکان وجود دارد که با یادآوری یک یا دو ویژگی فرد، به تمامی حافظه دست پیدا کنیم. شبکه عصبی انجمنی قادر است تا حافظه یا الگوهای خاص را در حالتی مشابه مغز ذخیره نماید. شکل زیر را در نظر بگیرید که در آن کاراکتر  $T$  و همان کاراکتر با مقداری نویز مشاهده می‌شود.

در صورتی که یک شبکه ی انجمنی، کاراکتر  $T$  را به خاطر سپرده باشد، با دیدن کاراکتر  $T$  نویزی و ناقص، با استفاده از

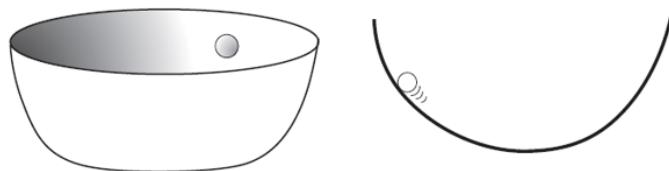


شکل ۱.۶: چپ: کاراکتر  $T$ ، راست: کاراکتر  $T$  مخدوش شده

سرخی که قسمت کامل آن به شبکه می‌دهد، قادر است تمام الگوی کاراکتر  $T$  را فراخوانی نماید.

## ۱.۲.۶ تحلیل فیزیکی حافظه

شبکه های انجمنی حالت خاصی از یک پدیده ی فیزیکی هستند که کاربردهای فراوانی در سایر علوم دارند. شکل زیر یک کاسه به همراه گوی متحرک در آن را نشان می‌دهد:

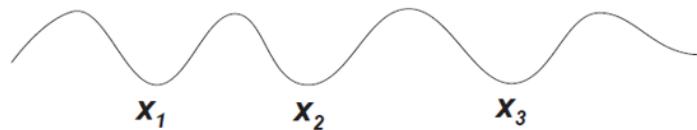


شکل ۲.۶: گوی و کاسه، تعبیر فیزیکی حافظه های انجمنی

هرگاه گوی از هر نقطه‌ی درون کاسه شروع به حرکت کند در نهایت به انتهای کاسه می‌رسد و در آنجا ساکن می‌شود. می‌توان چنین تعبیر نمود که مکان گوی در انتهای کاسه همان داده‌های به خاطر سپرده شده و نقطه‌ی اولیه‌ی گوی، همان اطلاعات اولیه برای بازیابی اطلاعات به خاطر سپرده شده است. مشخصه‌ی خاص نقطه‌ی انتهای کاسه، کمینه بودن انرژی پتانسیل آن است.

در یک شبکه انجمنی نیز به طریق مشابه یکتابع انرژی برای شبکه تعریف می‌گردد که الگوها در نقاط کمینه‌ی تابع انرژی ذخیره می‌گردند. در شکل ۳.۶، الگوهای  $x_1, x_2, x_3$  ذخیره شده است. به راحتی می‌توان با داشتن نقطه‌ی اولیه،

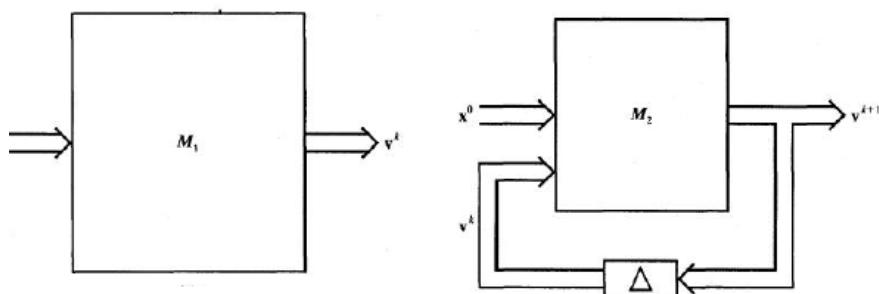
الگوی بازیابی شده را حدس زد.



شکل ۳.۶: ذخیره الگوهای کمینه های محلی

## ۲.۲.۶ شبکه های پافیلد

شبکه های انجمنی به دو دسته‌ی کلی استاتیک و دینامیک تقسیم می‌شوند. در حافظه‌ی استاتیک به محض اعمال ورودی، خروجی آماده می‌گردد و هیچ فیدبکی بین ورودی و خروجی وجود ندارد. اما در شبکه های دینامیکی، بعد از اعمال ورودی، شبکه طی یک فرایند دینامیکی و با استفاده از فیدبک خروجی، الگوها را فراخوانی می‌نماید. شکل زیر نحوه‌ی عملکرد ۲ شبکه‌ی استاتیکی و دینامیکی را نمایش می‌دهد: رابطه‌ی بین ورودی و خروجی شبکه‌ی استاتیکی



شکل ۴.۶: راست: حافظه دینامیکی چپ: حافظه استاتیکی

تصویر رابطه زیر است:

$$\begin{aligned} v^k &= M_1[x^k] \\ v^{k+1} &= M_2[x^k, v^k] \end{aligned} \tag{1.6}$$

## ۳.۶ شرح آزمایش

۱. یک شبکه های پافیلد طراحی کنید تا:

(آ) الگوی تک قطبی  $x = [1, 0, 0]$  را ذخیره نماید

(ب) الگوی دو قطبی  $x = [1, 1, -1]$  را ذخیره نماید.

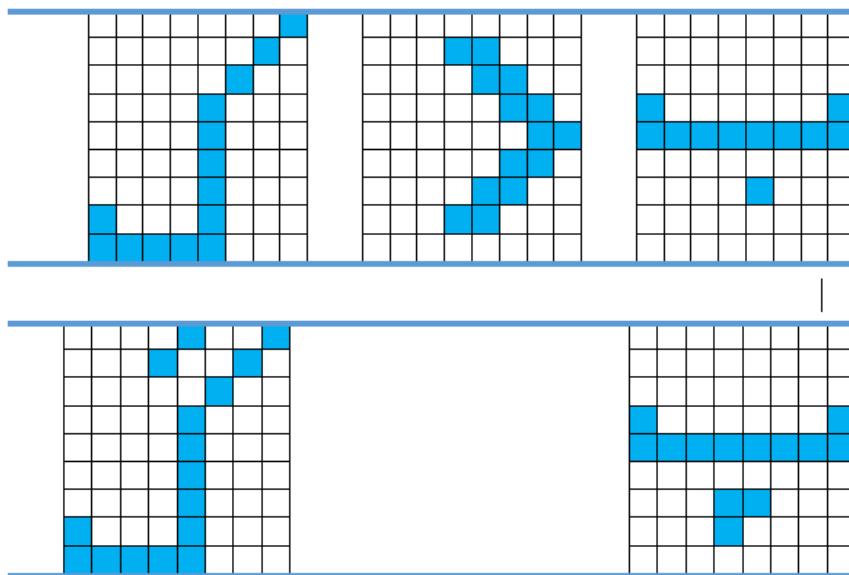
(ج) دو الگوی دو قطبی  $x_1 = [1, -1, 1, -1]$  و  $x_2 = [-1, 1, 1, -1]$  را ذخیره نماید.

(د) ماتریس  $W$  مربوط به ضرایب یک شبکه عصبی به صورت زیر است که یک الگو را در خود ذخیره کرده

است. برنامه ای بنویسید که الگوی ذخیره شده را بدست دهد.

$$w = \begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} \quad (2.6)$$

۲. با استفاده از شبکه عصبی هایپلاید و دستور *newhop* برنامه ای بنویسید که سه حرف "ب"، "د" و "ک" را ذخیره کند. سپس حروف ورودی را با ۲۰ درصد نویز جمع کنید و به عنوان ورودی شبکه تزریق کنید. آیا شبکه قادر به بازیابی حروف ذخیره شده است؟



شکل ۵.۶:

## ۴.۶ تمرین

۱. با استفاده از شبکه عصبی هایپلاید و دستور *newhop* برنامه ای بنویسید که دو حرف "'پ، "گ" را ذخیره کند. سپس حروف ورودی را با ۴۰ درصد نویز جمع کنید و به عنوان ورودی شبکه تزریق کنید. آیا شبکه قادر به بازیابی حروف ذخیره شده است؟

۲. با استفاده از شبکه هاپفیلد و دستور *newhop* در متلب برنامه ای بنویسید که دو حرف (پ) و (گ) را ذخیره کند.  
سپس این تصاویر را با ۴۰ درصد نویز جمع کنید و به عنوان ورودی به شبکه دهید. آیا شبکه قادر به بازیابی تصاویر اصلی می باشد؟

۳. برنامه ای در پایتون برای حل سوال قبل بنویسید.
۴. آیا می توان از شبکه هاپفیلد برای تشخیص ارقام ۰ تا ۹ استفاده کرد؟ چگونه؟ توضیح دهید.
۵. با استفاده از [این نمونه کد](#) حروف بیشتری را به شبکه Hopfield برای یادگیری ارائه دهید و قدرت بازیابی آن را با اعمال نویز به ورودی حرف های مختلف بسنجید.



# آزمایش هفتم(پیاده سازی شبکه عصبی $CNN$ )

## ۱.۷ مقدمه

در این آزمایش هدف پیاده سازی و کامپایل یک شبکه کانولوشنی بر روی مجموعه داده  $MNIST$  است. ابتدا به مروری کوتاه بر شبکه کانولوشنی پرداخته می شود.

شبکه عصبی کانولوشنی از پرکاربردترین شبکه های عصبی برای پردازش تصویر می باشد. عملکرد بسیاری از حملات نیز بر روی این شبکه عصبی بررسی می شوند. یک نمونه از شبکه کانولوشنی در شکل. ۱.۷ نمایش داده شده است.

شبکه کانولوشنی از لایه های اصلی زیر تشکیل شده است:

۱. لایه کانولوشنی

۲. لایه ادغام<sup>۱</sup>

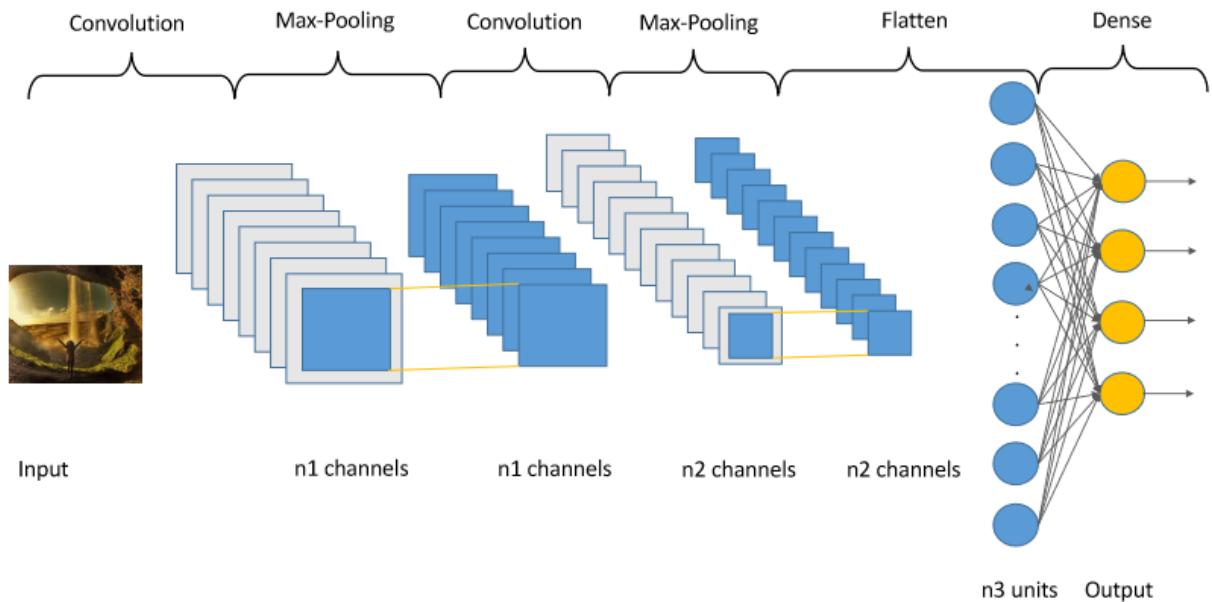
۳. لایه کاملاً متصل

در ادامه به بررسی هر یک از این لایه ها پرداخته می شود.

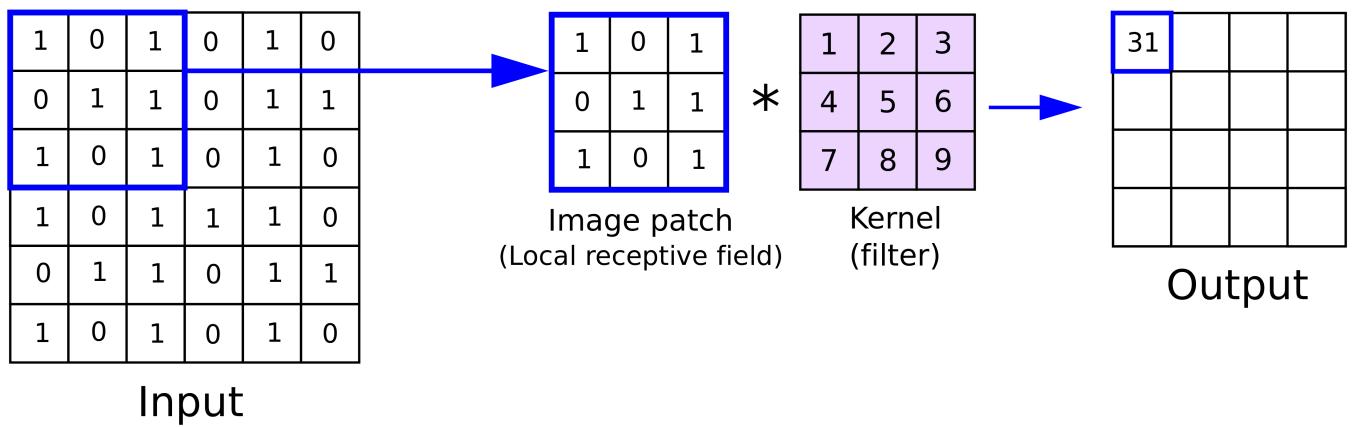
**لایه کانولوشنی** در این لایه عمل کانولوشن بر روی داده ورودی و با استفاده از تعدادی فیلتر انجام می شود. برای محاسبه هر درایه خروجی، ماتریس فیلتر بر روی ماتریس ورودی لغزانده می شود. عمل کانولوشن به این صورت تعریف می شود که ابتدا اولین عنصر فیلتر بر روی اولین عنصر ماتریس ورودی قرار می گیرد. سپس مجموع ضرب درایه های متناظر فیلتر با درایه های متناظر ماتریس ورودی محاسبه می شود. در نهایت فیلتر بر روی تصویر ورودی به اندازه پارامتر از پیش تعیین شده  $s$  یه جلو برد می شود. با تکرار این مراحل ماتریس خروجی محاسبه می شود.

---

<sup>1</sup> Pooling



شکل ۱.۷: نمونه ای از شبکه عصبی کانولوشنی



شکل ۲.۷: عملگر کانولوشن

عمل کانولوشن برای محاسبه یک درایه خروجی در شکل ۲.۷ نمایش داده شده است.

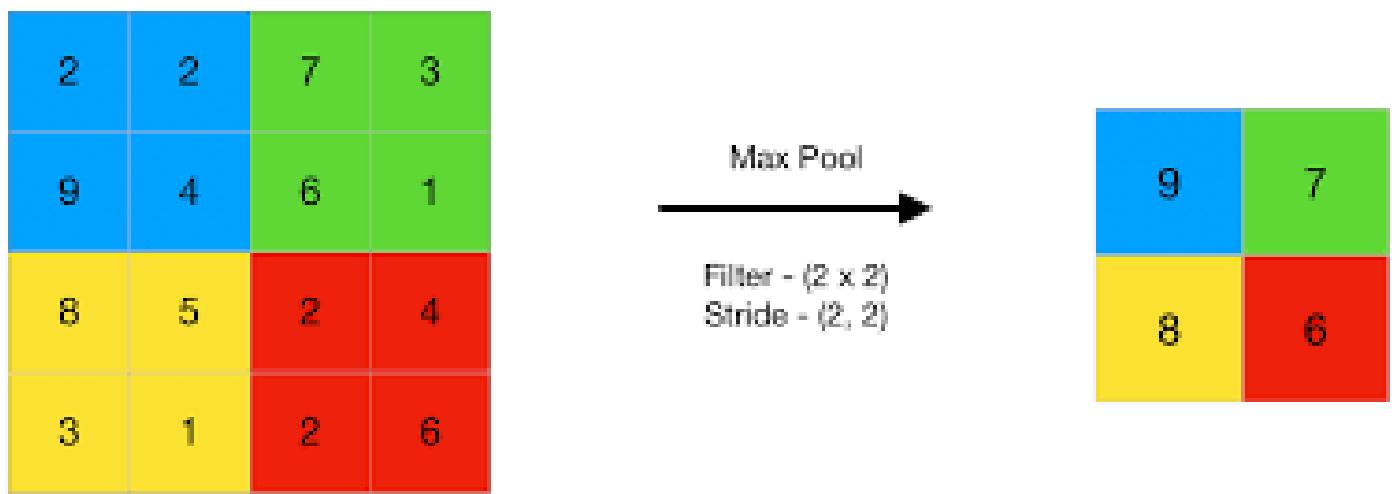
عمل کانولوشن در دو حالت یکسان<sup>۲</sup> و اصلی<sup>۳</sup> انجام می‌پذیرد. تفاوت این دو روش در ابعاد خروجی عملگر کانولوشن است. در حالت یکسان ماتریس خروجی لایه کانولوشن دارای ابعاد برابر با ماتریس ورودی است. در حالی که در حالت دوم ابعاد ماتریس خروجی می‌تواند متفاوت باشد.

پارامترهای فیلتر در شبکه عصبی کانولوشنی با استفاده از پس انتشار خطای<sup>۴</sup> حاصل می‌شود.

*same*<sup>۵</sup>

*valid*<sup>۶</sup>

*Back-propagation*<sup>۷</sup>



شکل ۳.۷: ادغام ماکزیم

**لایه ادغام** یک دیگر از لایه های شبکه کانولوشنی لایه ادغام است. این لایه هیچ پارامتر آموزشی ندارد. هدف این لایه کاهش ابعاد ماتریس ورودی و هم‌زمان حفظ اطلاعات ارزشمند ورودی است. در این لایه ابتدا ماتریس با ابعاد از پیش تعیین شده  $k$  در نظر گرفته می شود. این پارامتر معمولاً برابر با ۲ در نظر گرفته می شود. سپس با لغزاندن فیلتر بر روی ورودی اندازه ماتریس ورودی کاهش می یابد.

یک نمونه از عملگرهایی که برای نمونه برداری در این لایه استفاده می شود، عملگر بیشینه است. در این حالت ماتریس از پیش تعیین شده بر روی داده ورودی لغزانده می شود و تنها بیشینه عناصری که در هر بخش قرار می گیرند را به عنوان خروجی در نظر می گیرد. یک نمونه از عملگرد ادغام بیشینه در شکل ۳.۷ نمایش داده شده است.

**لایه کاملاً متصل** در این لایه یک شبکه عصبی کاملاً متصل قرار گرفته است. در این لایه هدف مرتبط کردن ماتریس نهايی با خروجي های نهايی شبکه است. وزن های شبکه کاملاً متصل از طريق پس انتشار خطأ بدست می آيد.

## ۲.۷ شرح آزمایش

در این آزمایش شبکه کانولوشنی در محیط *Tensorflow* پیاده سازی می شود.

برای پیاده سازی شبکه کانولوشنی در محیط *Tensorflow* گام های زیر را دنبال کنید:

۱. فراخوانی مجموعه داده و کتابخانه های ضروری : برخی از کتابخانه های ضروری برای پیاده سازی عبارتند از *matplotlib.pyplot* و *Tensorflow, numpy, panda* فراخوانی نمایید. با استفاده از دستور *import* از *Tensorflow, numpy, panda* فراخوانی نمایید. با استفاده از دستور *-verion* از بروز بودن نسخه تنسورفلو خود اطمینان حاصل نمایید (آخرین نسخه تنسورفلو نسخه ۲ می باشد).

```
# Run this cell to load the MNIST data
```

```
mnist_data = tf.keras.datasets.mnist  
(train_images, train_labels), (test_images, test_labels) = mnist_data.load_data()
```

شکل ۴.۷: کد لازم برای بارگذاری مجموعه داده

```
def scale_mnist_data(train_images, test_images):  
    """  
    This function takes in the training and test images as loaded in the cell above, and scales them  
    so that they have minimum and maximum values equal to 0 and 1 respectively.  
    Your function should return a tuple (train_images, test_images) of scaled training and test images.  
    """  
  
    | return (train_images, test_images)
```

شکل ۵.۷: فرمت کلی تابع لازم برای پیش پردازش داده ها

در این آزمایش، از مجموعه داده *MNIST* استفاده خواهد کرد. این مجموعه از ۶۰۰۰۰ تصویر رقم دست نویس با برچسب های مربوطه برای آزمایش و یک مجموعه از ۱۰۰۰۰ تصویر رقم دست نویس برای تست شبکه تشکیل شده است. تصاویر این مجموعه نرمال سازی شده اند. این مجموعه داده اغلب در تحقیقات یادگیری ماشین استفاده می شود. هدف شما پیاده سازی یک شبکه کانولوشنی برای تشخیص ارقام ورودی است. ارقام ورودی بین ۰ تا ۹ تغییر می کنند. برای بارگذاری مجموعه داده قطعه کد زیر را اجرا نمایید:

۲. پیش پردازش داده ها: برای یادگیری بهتر شبکه تابعی بنویسید که داده های تست و یادگیری شبکه را مقیاس بندی کند به گونه ای که مقادیر تصاویر خروجی بین صفر و یک قرار گیرد. شکل کلی تابع در تصویر ۵.۷ نمایش داده شده است برای اطمینان از درست کارکردن تابع خود قطعه کد نمایش داده شده در شکل (۶.۷) را اجرا کنید.

۳. ساخت شبکه کانولوشنی: در این قسمت هدف پیاده سازی تابعی است که یک شبکه کانولوشنی را بازگردد. برای پیاده سازی این شبکه از رابط کاربری *Sequential* استفاده می کنیم. قصد ساختن یک شبکه کانولوشنی با ویژگی های زیر را داریم:

(۱) لایه کانولوشن دو بعدی با اندازه فیلتر  $3 \times 3$  و ۸ فیلتر. از پدینگ *same* و تابع *Relu* به عنوان تابع فعالساز

```
# Run your function on the input data  
  
scaled_train_images, scaled_test_images = scale_mnist_data(train_images, test_images)  
  
# Add a dummy channel dimension  
  
scaled_train_images = scaled_train_images[..., np.newaxis]  
scaled_test_images = scaled_test_images[..., np.newaxis]
```

شکل ۶.۷: مقیاس کردن و اضافه کردن ۱ بعد به داده های ورودی

```

def get_model(input_shape):
    """
    This function should build a Sequential model according to the above specification. Ensure the
    weights are initialised by providing the input_shape argument in the first layer, given by the
    function argument.
    Your function should return the model.
    """
    model = tf.keras.Sequential([
        ...
    ])
    return model

```

## شکل ۷.۷: فرم کلی تابع

استفاده نمایید. مطمئن شوید که آرگومان  $input\_shape$  در این لایه ارائه شده است.

- (ب) یک لایه  $Maxpool$  ، با یک پنجره  $2 \times 2$ . (مقدار  $Stride$  را به صورت پیش فرض در نظر بگیرید)
- (ج) یک لایه مسطح، که ورودی را به یک تansor یک بعدی باز می کند.
- (د) دو لایه مخفی متراکم، هر کدام با ۶۴ نورون و تابع فعال ساز  $ReLU$ .
- (ه) یک لایه خروجی متراکم با ۱۰ واحد و تابع فعال سازی  $softmax$

ابعاد تصویر نیز به عنوان ورودی به شبکه کانولوشنی داده می شود. در اولین لایه شبکه کانولوشنی لازم است  $input\_shape$  را برابر با آرگومان دریافتی از ورودی تابع قرار دهید. برای پیاده سازی می توانید از کتابخانه های زیر استفاده نمایید:

`tf.keras.Sequential` •

`tf.keras.layers.Conv2D` •

`tf.keras.layers.MaxPooling2D` •

`tf.keras.layers.Flatten` •

`tf.keras.layers.Dense` •

فرم کلی تابع نیز در شکل ۷.۷ نمایش داده شده است.

برای اطمینان از عملکرد تابع قطعه کد نمایش داده شده در شکل ۸.۷ را اجرا نمایید.

۴. کامپایل کردن مدل: اکنون باید مدل را با استفاده از دستور `compile` کامپایل کنید. برای انجام این کار، باید یک بهینه ساز، یک تابع ضرر و یک متریک برای قضاؤت در مورد عملکرد مدل خود مشخص کنید. مدل را با استفاده از بهینه ساز `Adam` (با تنظیمات پیش فرض)، تابع ضرر `Cross - entropy` و دقت به عنوان تنها معیار اجرا

: # Run your function to get the model

```
model = get_model(scaled_train_images[0].shape)
```

شکل ۷.۸: دریافت یک شبکه کانولوشنی

```
def train_model(model, scaled_train_images, train_labels):
    """
    This function should train the model for 5 epochs on the scaled_train_images and train_labels.
    Your function should return the training history, as returned by model.fit.
    """
    return history
```

شکل ۹.۷: شمای کلی تابع برازش

نمایید. توجه داشته باشید لازم نیست تابع نوشته شده آرگومانی را بازگرداند و صرفا با اجرای دستور *compile* مدل کامپایل می شود.

۵. برازش مدل بر روی داده های ورودی و خروجی: در این قسمت هدف نوشتن تابعی برای برازش بر روی داده های ورودی و خروجی است. مدل و تصاویر مقیاس شده به همراه برچسب های صحیح نیز به عنوان ورودی به تابع داده می شود. اکنون باید با استفاده از روش *fit* مدل، مدل را بر روی مجموعه داده *MNIST* آموزش دهید. آموزش را برای ۵ epoch انجام داده و خروجی تابع *fit* را برای ترسیم نمودارهای دقت به عنوان خروجی تابع بروگردانید. فرم کلی تابع در شکل نمایش داده شده است. برای اطمینان از کارکرد صحیح تابع قطعه کد نمایش داده شده در تصویر ۱۰.۷ زیر را اجرا نمایید: در صورت پیاده سازی صحیح باید خروجی نزدیک به خروجی نمایش داده شده در تصویر ۱۱.۷ دریافت نمایید.

۶. نمایش های نمودارهای دقت: با اجرا کردن قطعه کدهای نمایش داده شده در شکل ۱۲.۷ و ۱۳.۷ نمودارهای دقت و ضرر بر حسب epoch را برای شبکه ترسیم نماید.

۷. ارزیابی مدل: در این قسمت هدف پیاده سازی تابعی برای ارزیابی مدل است. برای این کار می توانید از تابع evaluate استفاده نمایید. ارزیابی را بر روی مجموعه داده تست مقیاس شده و برچسب های متناظر با آن

# Run your function to train the model

```
history = train_model(model, scaled_train_images, train_labels)
```

شکل ۷.۹: برازش تابع

```

Train on 60000 samples
Epoch 1/5
60000/60000 [=====] - 83s 1ms/sample - loss: 0.2034 - sparse_categorical_accuracy: 0.9398
Epoch 2/5
60000/60000 [=====] - 83s 1ms/sample - loss: 0.0653 - sparse_categorical_accuracy: 0.9804
Epoch 3/5
60000/60000 [=====] - 82s 1ms/sample - loss: 0.0455 - sparse_categorical_accuracy: 0.9858
Epoch 4/5
60000/60000 [=====] - 85s 1ms/sample - loss: 0.0330 - sparse_categorical_accuracy: 0.9894
Epoch 5/5
60000/60000 [=====] - 81s 1ms/sample - loss: 0.0260 - sparse_categorical_accuracy: 0.9919

```

شکل ۱۱.۷: برازش تابع

```

frame = pd.DataFrame(history.history)

# Run this cell to make the Accuracy vs Epochs plot

acc_plot = frame.plot(y="accuracy", title="Accuracy vs Epochs", legend=False)
acc_plot.set(xlabel="Epochs", ylabel="Accuracy")

```

شکل ۱۲.۷: ترسیم نمودار دقت بر حسب epoch

انجام دهید. دقت و خسر بر روی مجموعه داده تست را به عنوان خروجی تابع برگردانید. شما کلی تابع در شکل نمایش داده شده است. با اجرا کردن قطعه کد نمایش داده شده در شکل ۱۵.۷ از درست اجرا شدن تابع اطمینان حاصل نمایید.

۸. پیش بینی شبکه: در نهایت با اجرا کردن کد نمایش داده شده در شکل ۱۶.۷ عملکرد شبکه خود را بر روی برخی از تصاویر *MNIST* مشاهده کنید.

## ۳.۷ تمرین

۱. با یک شبکه کانولوشنی با ساختار دلخواه به جداسازی مجموعه داده *Cifar – 10* بپردازید.
۲. بررسی کنید چه شبکه هایی توسعه *Sequential – API* قابل پیاده سازی نیستند. برای پیاده سازی این شبکه اثربخشی تعداد لایه های شبکه بر روی دقت پیش بینی شبکه بر روی داده یادگیری را بررسی نمایید

```

: # Run this cell to make the Loss vs Epochs plot

acc_plot = frame.plot(y="loss", title = "Loss vs Epochs", legend=False)
acc_plot.set(xlabel="Epochs", ylabel="Loss")

```

شکل ۱۳.۷: ترسیم نمودار خسارت بر حسب epoch

```

def evaluate_model(model, scaled_test_images, test_labels):
    """
    This function should evaluate the model on the scaled_test_images and test_labels.
    Your function should return a tuple (test_loss, test_accuracy).
    """
    test_loss, test_accuracy = model.evaluate()

    return (test_loss, test_accuracy)

```

شکل ۱۴.۷: ارزیابی تابع

```
# Run your function to evaluate the model
```

```

test_loss, test_accuracy = evaluate_model(model, scaled_test_images, test_labels)
print(f"Test loss: {test_loss}")
print(f"Test accuracy: {test_accuracy}")

```

شکل ۱۵.۷: ارزیابی تابع

```
# Run this cell to get model predictions on randomly selected test images
```

```

num_test_images = scaled_test_images.shape[0]

random_inx = np.random.choice(num_test_images, 4)
random_test_images = scaled_test_images[random_inx, ...]
random_test_labels = test_labels[random_inx, ...]

predictions = model.predict(random_test_images)

fig, axes = plt.subplots(4, 2, figsize=(16, 12))
fig.subplots_adjust(hspace=0.4, wspace=-0.2)

for i, (prediction, image, label) in enumerate(zip(predictions, random_test_images, random_test_labels)):
    axes[i, 0].imshow(np.squeeze(image))
    axes[i, 0].get_xaxis().set_visible(False)
    axes[i, 0].get_yaxis().set_visible(False)
    axes[i, 0].text(10., -1.5, f'Digit {label}')
    axes[i, 1].bar(np.arange(len(prediction)), prediction)
    axes[i, 1].set_xticks(np.arange(len(prediction)))
    axes[i, 1].set_title(f"Categorical distribution. Model prediction: {np.argmax(prediction)}")

plt.show()

```

شکل ۱۶.۷: پیش بینی تابع

---

ها چه راه حلی را پیشنهاد می دهید.



# آزمایش هشتم (پیاده سازی شبکه عصبی $RNN$ )

## ۱.۸ پیش گزارش

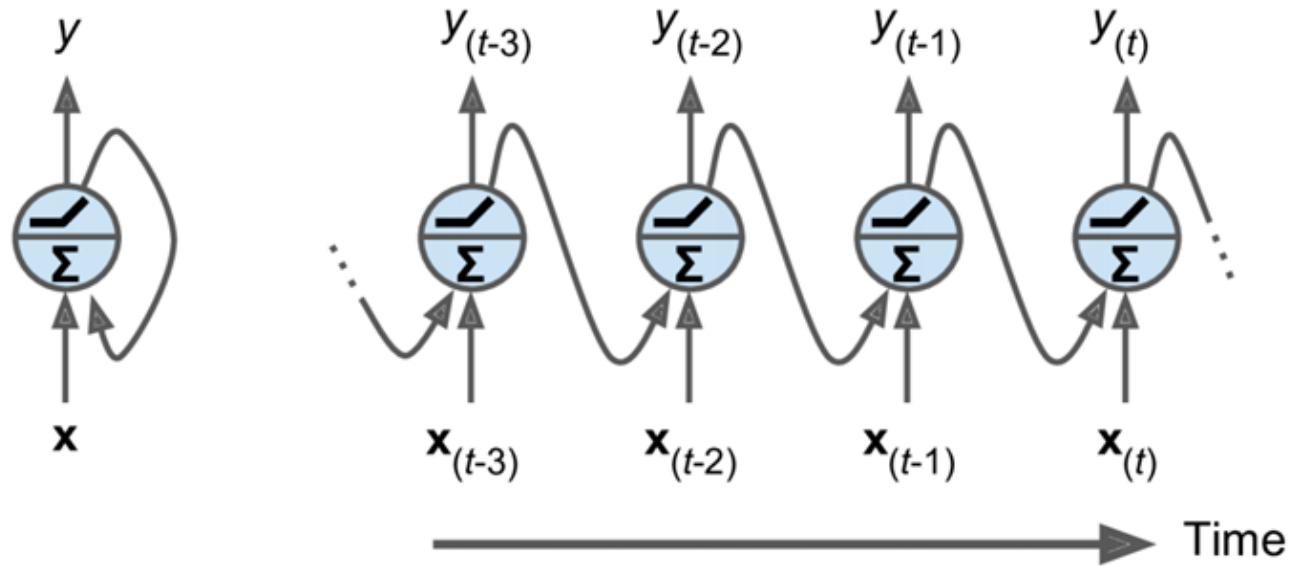
۱. معماری‌های مختلف شبکه عصبی بازگشتی را بیان کنید.
۲. برای هر یک از این معماری‌ها یک مثال و کاربرد بیاورید.
۳. مهم‌ترین چالش‌ها در آموزش شبکه عصبی بازگشتی چه است و چگونه می‌توان آن را مدیریت کرد؟

## ۲.۸ مقدمه

در این آزمایش ابتدا به مرور مفاهیم اولیه شبکه‌های عصبی بازگشتی (Recurrent Neural Network) و سپس به نحوه آموزش آن‌ها با استفاده از الگوریتم پساننتشار در زمان می‌پردازیم. در نهایت این شبکه را برای دیتابست *imdb* به منظور پیش‌بینی اینکه نظرات نوشته مثبت است یا منفی، پیاده سازی می‌کنیم.

## ۱.۲.۸ ساختار شبکه عصبی بازگشتی

یک شبکه عصبی بازگشتی بسیار شبیه به یک شبکه عصبی پیش‌خور (feedforward) به نظر می‌رسد، با این تفاوت که دارای اتصالات معکوس به عقب است. در هر مرحله زمانی  $t$  (که فریم نیز نامیده می‌شود)، یک شبکه بازگشتی ورودی‌های  $x_t$  و همچنین خروجی خود از مرحله زمانی قبل،  $y(t-1)$  را دریافت می‌کند. به این حالت، باز کردن شبکه در طول زمان گفته می‌شود. هر نورون بازگشتی دو دسته وزن دارد: یکی برای ورودی‌های  $x_t$  و دیگری برای خروجی



شکل ۱.۸: شبکه عصبی بازگشتی باز شده در طول زمان

های مرحله زمانی قبلی، ( $y_{t-1} - y_t$ ). خروجی این شبکه را در رابطه زیر مشاهده می‌نمایید.

$$y_t = \sigma(W_x^\top x_t + W_y^\top y_{t-1} + b)$$

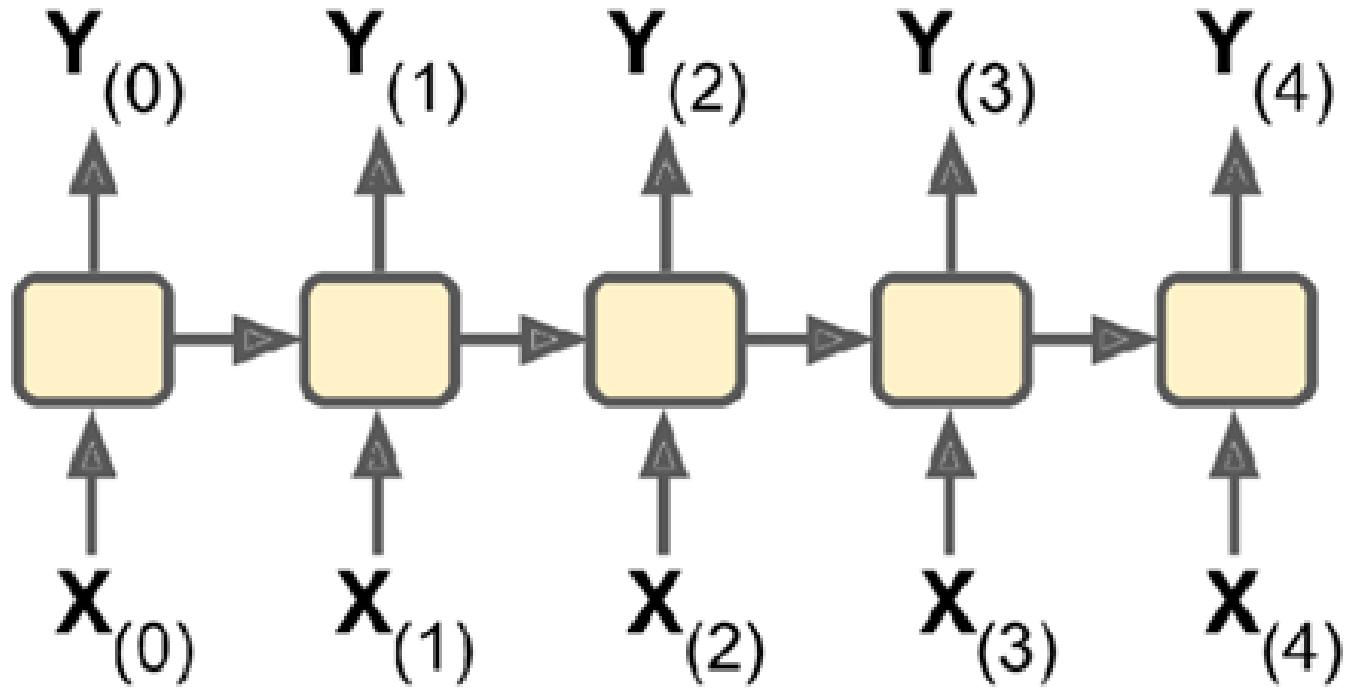
برای آموزش یک RNN شبکه را در طول زمان باز می‌کنیم (شکل ۱.۸) و سپس به سادگی از الگوریتم پس‌انتشار معمولی استفاده می‌کنیم. این استراتژی را پس‌انتشار در طول زمان (BPTT) می‌نامند.

## ۳.۸ شرح آزمایش

در این قسمت به پیاده سازی و آموزش یک شبکه عصبی بازگشتی روی مجموعه دادگان imdb می‌پردازیم. هدف از این آزمایش آموزش مدلی است که بتواند مثبت یا منفی بودن جملات نوشته شده برای imdb را پیش‌بینی کند.

### ۱.۳.۸ داده ورودی

داده استفاده شده در این آزمایش مجموعه دادگان IMDB موجود در Keras یک مجموعه داده IMDB imdb است. مجموعه داده IMDB یک مجموعه داده محبوب برای طبقه بندی احساسات باینری است. این شامل ۵۰۰۰۰ نقد فیلم از پایگاه اینترنتی فیلم‌های اینترنتی (IMDB) است که دارای برچسب مثبت یا منفی هستند. این مجموعه داده به طور گسترده در پردازش زبان طبیعی و وظایف یادگیری ماشین، به ویژه برای کسانی که بر تجزیه و تحلیل احساسات تمرکز می‌کنند، استفاده می‌شود.



شکل ۲.۸: معماری sequence-to-sequence

## ۲.۳.۸ آموزش مدل

با توجه به داده ورودی و خروجی، از یک شبکه عصبی بازگشتی sequence-to-sequence برای آموزش مدل استفاده کنید (شکل ۲.۸). برای پیش‌بینی سری زمانی:

۱. با استفاده از دستور SimpleRNN در keras، مدل شبکه عصبی بازگشتی را تعریف کنید.
۲. مدل شبکه عصبی خود را با تعداد لایه‌ها و تعداد نورون‌های هر لایه به صورت دلخواه طراحی کنید.
۳. با توجه به استفاده از معماری sequence-to-sequence، لایه آخر (خروجی) را یک لایه Dense با تعداد نورون برابر با عرض پنجره، تعریف کنید.
۴. مدل را آموزش داده و خطای داده تست را معین کنید.
۵. برای تعدادی از داده‌های تست، نمودار تخمین‌زده شده و داده واقعی را رسم کرده و با هم مقایسه کنید.

## ۳.۳.۸ بهبود مدل

به منظور بهبود مدل می‌توانید:

- به جای SimpleRNN از لایه LSTM با دستور LSTM در keras استفاده کنید.

• به جای SimpleRNN از لایه GRU با دستور keras GRU استفاده کنید.

• به جای استفاده از لایه خروجی Dense، از لایه با دستور TimeDistributed در keras استفاده کنید.

برای هر یک از حالت‌های بالا، صحبت مدل و زمان آموزش را با هم و با لایه RNN معمولی مقایسه کنید و در نهایت شبکه بهینه را مشخص کنید.

## ۴.۸ تمرین

رای دیتابست Daily Minimum Temperatures in Melbourne نیز همانند آزمایش انجام شده، شبکه‌های RNN مختلف را طراحی کرده و بهترین را گزارش کنید.

# آزمایش نهم (شناسایی به کمک شبکه عصبی)

## ۱.۹ پیش گزارش

۱. تابع تبدیل سیستم زمان گسسته‌ای به صورت زیر است. تابع تبدیل را در محیط سیمولینک مطلب شبیه‌سازی کنید

$$H(z) = \frac{-z^2 + 1.9z + .95}{z^3 - .18z^2 + .08z - .08} \quad (1.9)$$

۲. با مراجعه به اینترنت مزایای کنترل کننده‌های هوشمند نظیر کنترل کننده‌ی عصبی بر سایر کنترل کننده‌ها را بیان نمایید.

۳. با مراجعه به اینترنت در مورد کاربردهای شناسایی سیستم دینامیکی تحقیق نمایید.

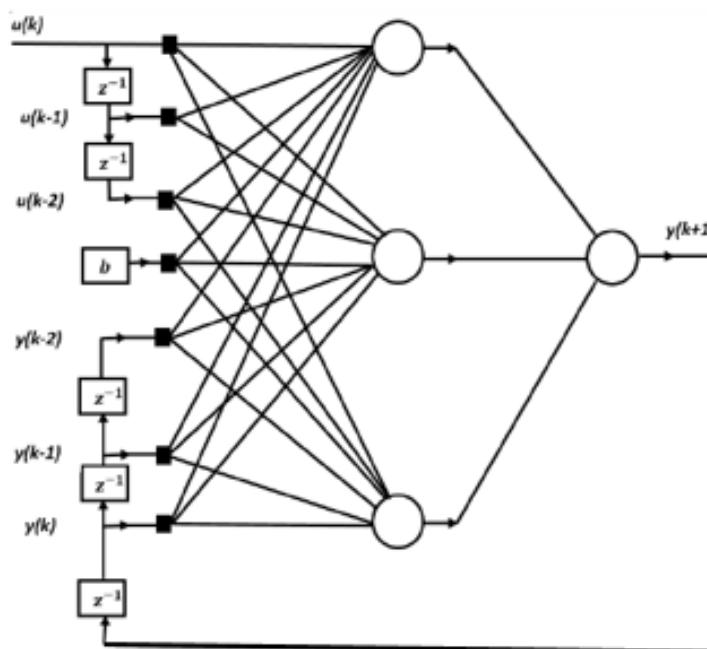
۴. با مراجعه به اسلاید‌های درس قسمت شبکه‌های عصبی (*lecture 7* و اسلاید‌های ۱۶ تا ۱۹) ابعاد تمام پارامترها را بر حسب ابعاد داده ورودی محاسبه کنید.

## ۲.۹ مقدمه

شناسایی سیستم‌ها یکی از پرکاربرد ترین مسائل در علم کنترل است. با استفاده از شبکه‌های عصبی چند لایه‌ی پیاده‌سازی شده در جلسات قبل، می‌توان سیستم‌های ناشناخته را به صورت زمان گسسته یا زمان پیوسته شناسایی کرد. در این آزمایش به شناسایی سیستم‌های گسسته و پیوسته پرداخته می‌شود.

## ۱.۲.۹ شناساگر استاتیکی

یکی از روش‌های شناسایی برای سیستم‌های دینامیکی استفاده از شناساگر استاتیکی است. در این روش ابتدا تعداد داده ورودی و خروجی از سیستم با تاخیرهای مشخص ذخیره شده و به عنوان ورودی به شبکه عصبی اعمال می‌شود. با اعمال این داده‌ها، شبکه عصبی آموزش داده شده و مدل را شناسایی می‌نماید. هدف شبکه عصبی تخمین خروجی است به گونه‌ای که خروجی تخمین زده شده برابر با خروجی اصلی سیستم باشد(یا اختلاف اندکی داشته باشد) در شکل ۱.۷ مدل کلی یک شناساگر استاتیکی نمایش داده شده است . در این شکل ورودی های شبکه عصبی ورودی سیستم و ورودی سیستم تاخیر یافته و خروجی سیستم و خروجی سیستم تاخیر یافته است. خروجی شبکه نیز تخمینی از خروجی سیستم است.



شکل ۱.۹: شناساگر استاتیکی

در ادامه به بررسی ساختار شناساگر پیوسته پرداخته می شود.

## ۲.۲.۹ شناساگر پیوسته

مدل‌های شناسایی سیستم پیوسته به ۲ دسته کلی تقسیم می‌شوند:

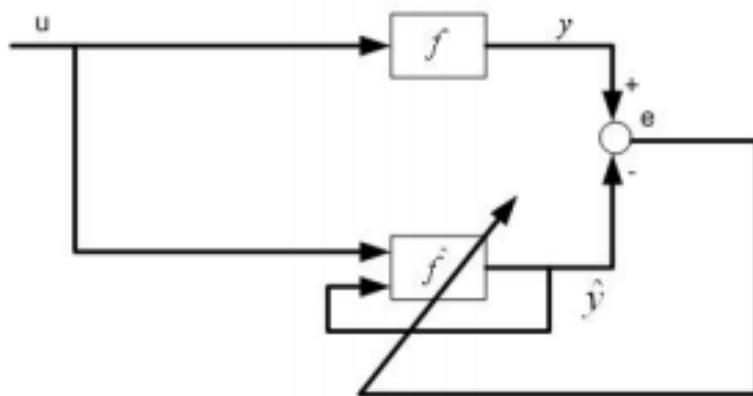
۱. مدل موازی

۲. مدل سری -موازی

در ادامه به مروری کلی بر ساختار این ۲ دسته پرداخته می شود.

### ۳.۲.۹ مدل موازی

مدل کلی شناسایی با استفاده از مدل موازی سیستم در شکل ۳.۲.۷ نمایش داده شده است. در این مدل با استفاده از خطای بین خروجی شبکه عصبی و خروجی اصلی سیستم پارامترهای شبکه تازه سازی می‌شوند. مدل‌های مختلفی را می‌توان برای شبکه عصبی در نظر گرفت برای مثال می‌توان شبکه عصبی را یک شبکه عصبی چند لایه که وزن‌های آن با استفاده از قانون پس انتشار خطا تازه سازی می‌شود در نظر گرفت.



شکل ۳.۹: شناسایی با استفاده از مدل موازی

در مدل موازی فیدبک در نظر گرفته شده در ساختار شبکه عصبی از خروجی خود شبکه است. این فیدبک ممکن است مانع همگرایی شبکه عصبی شود و یا باعث کند شدن سرعت همگرایی شود.

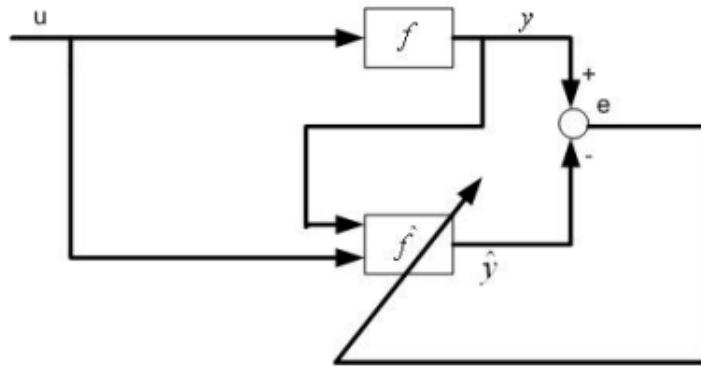
### مدل سری-موازی

با فیدبک گرفتن از خروجی اصلی سامانه به جای خروجی شبکه عصبی می‌توان همگرایی سیستم را نسبت به حالت شناساگر موازی بهبود بخشد.

با تغییر دادن فیدبک در مدل سری (فیدبک موجود در ساختار شبکه عصبی) مدل شناساگر سری-موازی حاصل می‌شود. ساختار کلی مدل در شکل ۳.۷ نمایش داده شده است. در این مدل نیز پارامترهای شبکه با استفاده از خطای بین خروجی شبکه و خروجی اصلی سیستم تازه سازی می‌شوند. ورودی این شبکه ورودی اصلی سامانه و خروجی اصلی سیستم است.

### ۴.۲.۹ ترکیب شناساگر استاتیکی و مدل موازی

در مدل استاتیکی ورودی‌های شبکه شامل خروجی ورودی‌های تاخیر یافته مدل اصلی سیستم بود. تاخیر در سیستم‌های پیوسته با استفاده از انтگرال‌گیر مدل می‌شود. به دلیل ناپایدار بودن انتگرال‌گیر از یک فیلتر برای مدل کردن

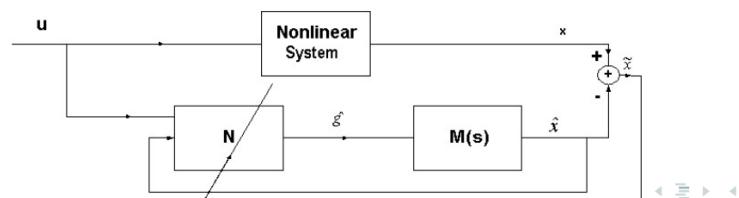


شکل ۳.۹: شناسایی با استفاده از مدل سری-موازی

تاخیر در سیستم های پیوسته استفاده می شود. با در نظر گرفتن این نکته برای عملکرد بهتر شناساگر موازی می توان شناساگر موازی را به گونه ای توسعه داد که مقادیر گذشته خروجی نیز در ورودی شبکه وجود داشته باشد. در این آزمایش به پیاده سازی مدل توسعه یافته شناساگر موازی برای تخمین حالت های یک سیستم پیوسته پرداخته می شود.

ساختار کلی این شبکه در شکل ۴.۷ نمایش داده شده است. در این شکل ورودی شبکه با  $\hat{w}$  نمایش داده شده است.  $\hat{w}$  ورودی و خروجی سیستم بعد از گذشتن از یک فیلتر است. با توجه به اینکه ورودی شبکه  $\hat{w}$  خروجی ها و ورودی های تاخیر یافته نیز در ورودی شبکه تاثیر می گذارند و باعث عملکرد بهتر مدل موازی می شوند. شبکه عصبی را نیز می توان به صورت یک شبکه عصبی چند لایه که با استفاده از قانون پس انتشار خطا وزن های آن تازه سازی می شوند در نظر گرفت.

برای آشنایی بیشتر با این مدل و بررسی دقیق تر آن می توانید به *lecture* های درس مراجعه کنید.<sup>۱</sup>



شکل ۴.۹: شناساگر موازی توسعه یافته

<sup>۱</sup> امکان ترکیب مدل سری-موازی و مدل استاتیکی نیز وجود دارد که در این گزارش به آن پرداخته نشده است.

## ۳.۹ شرح آزمایش

در این قسمت ابتدا به پیاده سازی یک شناساگر استاتیکی می پردازید سپس به تخمین حالت های یک سیستم پیوسته توسط شناساگر موازی (ترکیب شناساگر استاتیکی و شناساگر موازی) می پردازید

### ۱.۳.۹ پیاده سازی شناساگر استاتیکی

در این قسمت به پیاده سازی بک مدل ساده از شناساگر استاتیکی پرداخته می شود.  
برای پیاده سازی یک مدل ساده از شناساگر استاتیکی مراحل زیر را دنبال کنید:

۱. مدل زیر را در محیط سیمولینک مطلب شبیه سازی کنید

$$H(z) = \frac{-z^2 + 1.9z + .95}{z^3 - .18z^2 + .08z - .08} \quad (2.9)$$

۲. نویز سفید را به عنوان ورودی به سیستم شبیه سازی شده اعمال کنید.

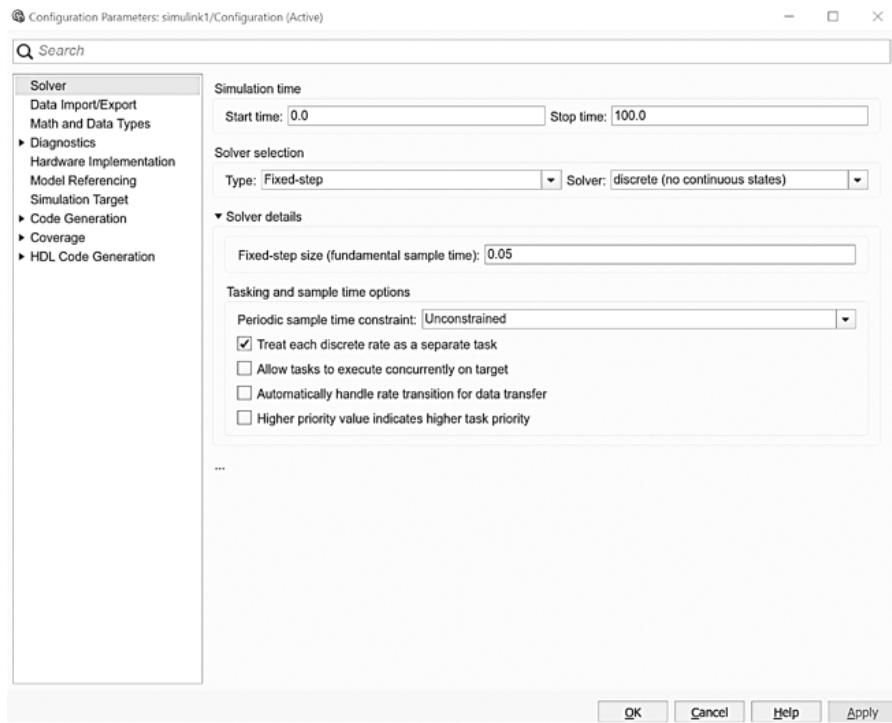
۳. ورودی و خروجی سیستم شبیه سازی شده را در *work space* ذخیره کنید.  
برای ذخیره کردن ورودی و خروجی ها مراحل زیر را دنبال کنید:

(آ) برای ذخیره ورودی و خروجی های مدل طراحی شده، با استفاده از *Model Configuration Parameters*، گزینه *Simulation* شود تا امکان ذخیره سازی فراهم شود. برای این منظور از منوی *Solver*، تنظیمات را مطابق شکل ۷.۵ انجام دهید.

(ب) با قرار دادن بلوک *simout* در ورودی و خروجی سامانه و اجرای سیمولینک داده های ورودی و خروجی را انتخاب کنید. حال در منوی *Solver*، تنظیمات را مطابق شکل ۷.۵ انجام دهید.

۴. در پنل *Workspace* مقادیر ورودی و خروجی های تولید شده قابل مشاهده اند. یک *mfile* مطلب بازکنید تا مقادیر مربوطه را در متغیرهای *input* و *output* ذخیره کنید.

۵. در مرحله‌ی بعد لازم است تا داده های با تاخیر ذخیره شوند. به این ترتیب می‌توان با محاسبه داده های تاخیردار در ورودی و خروجی، آموزش را بهبود بخشید. (تعداد تاخیر ها در ورودی و خروجی برای هر سیستم با توجه به درجه سیستم متفاوت است).



شکل ۵.۹: تنظیمات سیمولینک

داده های با تاخیر را می توان به صورت زیر تولید کرد.

$$inputDelayed1 = [0; input(1 : end - 1)];$$

$$inputDelayed2 = [0; 0; input(1 : end - 2)];$$

$$outputDelayed1 = [0; output(1 : end - 1)];$$

$$outputDelayed2 = [0; 0; output(1 : end - 2)];$$

۶. ورودی شبکه عصبی را به صورت زیر تعریف کنید.

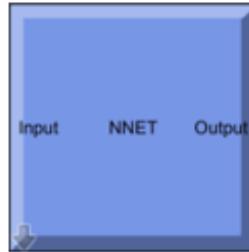
$$x=[input, inputDelayed1, inputDelayed2, outputDelayed2, outputDelayed1];$$

۷. حال با استفاده از ابزار *FunctionFittingNeuralNetwork* داده های جمع آوری شده را مورد آموزش قرار دهید.

آموزش را با داده های موجود انجام داده و در نهایت شبکه عصبی طراحی شده را به محیط سیمولینک *export*

کنید. شبکه عصبی به شکل زیر خواهد بود. که لازم است ورودی هایی مشابه آنچه در *inputDelayedi* ها

( $i = 1, 2$ ) تعریف شد، در سیمولینک تعریف شده و به سیستم اعمال شود



شکل ۶.۹: بلوک شبکه عصبی استاتیک طراحی شده

برای تست مدل طراحی شده، یک ورودی سینوسی به مدل اصلی و مدل استاتیک (مدل تهیه شده به وسیله شبکه عصبی) دهدید و خروجی ها را مقایسه کنید. (دققت کنید ورودی های و خروجی های تا خبر دار نیز باید به عنوان ورودی به بلوک شبکه عصبی استاتیک طراحی شده داده شوند. برای دادن چند ورودی به شبکه می توانید از *demux* استفاده کنید).

### ۲.۳.۹ پیاده سازی شناساگر پیوسته

در این قسمت قصد داریم به تخمین حالت های یک سیستم توسط شناساگر موازی توسعه یافته (ترکیب شناساگر موازی و شناساگر استاتیکی) بپردازیم.

در (شکل ۷.۷) مدل کلی شناساگر آورده شده است. برای پیاده سازی این لایه ها از شبکه عصبی *MLP* که پیشتر طراحی شده بود استفاده می شود. طرح کلی از ۳ بخش اصلی تشکیل شده است:

۱. بخش *sigmoid*

۲. بخش خطی

۳. بخش فیلتر

در این مدل ابتدا قوانین تازه سازی وزن های شبکه مطابق با معادله ۷.۳ پیاده سازی می شود. وزن های شبکه بر اساس  $\tilde{x}$  (خطای شبکه) (تازه سازی می شوند. سپس با استفاده از وزن های شبکه  $\hat{g}$  (بلوک اول شکل ۷.۷ این قسمت را پیاده سازی می کنند) مطابق با رابطه ۷.۴ محاسبه می شود و درنهایت با استفاده از  $\hat{g}$  حالت های سیستم تخمین زده می شوند. (در بلوک سوم شکل ۷.۷ این قسمت پیاده سازی می شود) حالت های تخمین زده شده توسط شبکه عصبی مطابق با رابطه ۷.۵ تازه سازی می شود

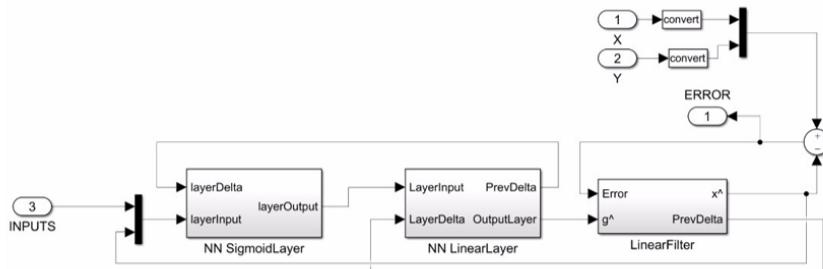
$$\begin{aligned}\dot{\hat{W}} &= -\eta_1(\tilde{x}^T A^{-1})^T (\sigma(\hat{V} \hat{x}))^T - P_1 \| \tilde{x} \| \hat{w} \\ \dot{\hat{V}} &= -\eta_2(\tilde{x}^T A^{-1} \hat{W}(I - \dot{\sigma}(\hat{V} \hat{x})))^T \hat{x}^T - p_2 \| \tilde{x} \| \hat{V}\end{aligned}\quad (۳.۹)$$

$$\tilde{x} = x - \hat{x}$$

$$\hat{g}(\hat{x}, u) = \hat{W} \sigma(\hat{V} \hat{x}) \quad (۴.۹)$$

$$\dot{\hat{x}} = A\hat{x} + \hat{g}(\hat{x}, u) + \epsilon \quad (5.9)$$

در ادامه به بررسی و پیاده سازی هر یک از بلاک ها پرداخته می شود.  
برای پیاده سازی شبکه مراحل زیر را دنبال کنید. (دقت کنید در مدل پیاده سازی شده ابعاد ورودی و تعداد نورون ها



شکل ۷.۹: ساختار کلی شناساگر عصبی

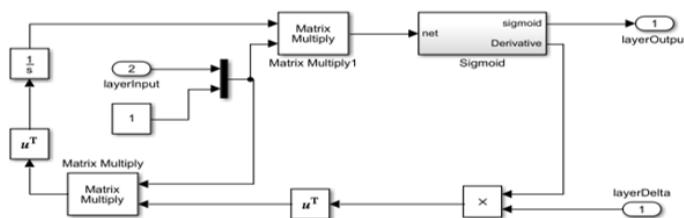
باید به صورت متغیر تعريف شوند تا در صورت نیاز تغییر پیدا کند):

۱. ابتدا بلاک مربوط به *sigmoid* را پیاده سازی کنید.

ورودی بخش *sigmoid* ورودی مطلوب سیستم و خطاب لایه بعد است مدل کلی بخش *sigmoid* در شکل ۷.۸ در شکل آورده شده است. در این قسمت تابع *sigmoid* و مشتق آن نیز که برای قوانین تازه سازی لازم است پیاده سازی می شوند.

در این بلاک قوانین تازه سازی  $\hat{V}$  مطابق با معادله ۳.۷ پیاده سازی کنید برای پیاده سازی فرض کنید  $\hat{x}, W$  را از ورودی دریافت می کنید.  $(W$  در بلاک دوم تولید می شود و سپس به صورت فیدبک به این بلاک منتقل می شود).

بعد از پیاده سازی قوانین تازه سازی  $\hat{V}$  با استفاده از انتگرال گیری سیگنال  $\hat{V}$  را تولید کنید. سپس با پیاده سازی تابع *sigmoid* مقدار  $\sigma(\hat{V}\hat{x})$  را به عنوان خروجی به لایه بعد انتقال دهید. از خروجی این لایه در لایه بعد برای تازه سازی  $\hat{W}$  استفاده می شود.

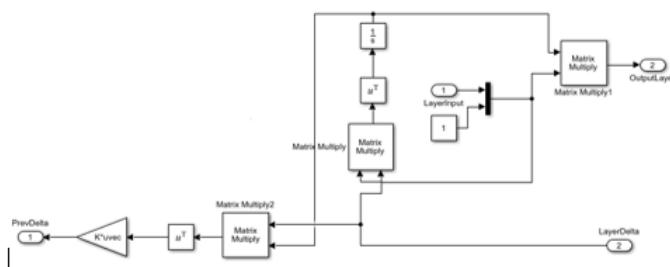


شکل ۸.۹: ساختار کلی بخش *sigmoid*

۲. ساختار کلی این لایه در شکل ۹.۷ نمایش داده شده است.

در این لایه ابتدا با استفاده از خروجی لایه قبلاً  $\hat{W}$  را با استفاده از رابطه ۳.۷ پیاده سازی کنید در این قسمت نیز مانند بخش *sigmoid* فرض کنید.  $\hat{x}$  از ورودی دریافت می‌شود. سپس با انتگرال گیری از قانون تازه سازی  $\hat{W}$ ،  $\hat{W}$  را تولید کنید. بعد از محاسبه  $\hat{W}$  و با استفاده از خروجی لایه قبل (( $\sigma(\hat{V}\hat{x})$ ) را مطابق با رابطه ۴.۷ محاسبه کنید.

با توجه به اینکه  $W$  در رابطه تازه سازی  $\hat{V}$  مورد نیاز است و با توجه به اینکه قانون تازه سازی  $\hat{V}$  در بلوک اول (بخش *sigmoid*) پیاده سازی شده است.  $\hat{W}$  را به صورت فیدبک از این بلوک به بلوک اول منتقل کنید. در نهایت با استفاده از  $\hat{W}$  و  $\sigma(\hat{V}\hat{x})$  (ورودی شبکه) و با توجه به رابطه ۴.۷  $\hat{g}$  را تولید کنید و به عنوان ورودی به لایه بعد انتقال دهید.



شکل ۹.۹: ساختار کلی بخش خطی شناساگر

۳. در این بخش با استفاده از  $\hat{g}$  تولید شده در لایه پیشین و با توجه به رابطه ۷.۵ ابتدا  $\hat{x}$  را تولید کنید و سپس با انتگرال گیری از  $\hat{x}$  سیگنال  $\hat{x}$  را تولید کنید.

سپس با کم کردن حالت های تخمین زده شده شبکه از حالت های اصلی سیستم  $\tilde{x}$  را تولید کنید. با توجه به اینکه این سیگنال را در بلوک های دوم و سوم به عنوان ورودی فرض کردید این سیگنال را به عنوان ورودی به بلوک های دوم و سوم دهید.

### ۳.۳.۹ تست شناساگر شبیه سازی شده

برای تست شناساگر شبیه سازی شده تابع تبدیل نشان داده شده در معادله ۶.۷ را در محیط سیمولینک متلب پیاده سازی کنید و حالت های سیستم را با استفاده از شناساگر پیاده سازی شده تخمین بزنید. ورودی سیستم را  $\sin(t)$  در نظر بگیرید.

نمودار خطای بین حالت های تخمین زده و حالت های اصلی سیستم را رسم کنید.

$$H(s) = \frac{40}{0.02s^2 + s} \quad (6.9)$$

## ۶.۹ تمرین

۱. فرض کنید در بازوی ربات با مدل زیر ( $M$ ) شناخته شده نیست، این سیستم را شناسایی کنید. می خواهیم انتهای بازوی ربات مسیر  $\sin 10t$  را دنبال کند.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= M^{-1}(x)[u - C(x_1, x_2)x_2 - Dx_2 - g(x_1)] \end{aligned} \quad (7.9)$$

که موقعیت انتهای هر لینک  $x_2 = \dot{x}_1$  سرعت انتهای هر لینک  $u$  گشتاور،  $M$  ماتریس اینرسی،  $C$  نیروهای کریولیس و گریز از مرکز،  $D$  دمپینگ ویسکوز و  $g$  نیروی گرانش است و به صورت زیر تعریف شده است:

$$\begin{aligned} M &= \begin{bmatrix} a_1 + 2a_4 \cos q_2 & a_2 + a_4 \cos q_2 \\ a_2 + a_4 \cos q_2 & a_3 \end{bmatrix}, \quad D = 0 \\ C &= a_4 \sin q_2 \begin{bmatrix} -\dot{q}_2 & -(\dot{q}_1 + \dot{q}_2) \\ \dot{q}_1 & 0 \end{bmatrix}, \quad g = \begin{bmatrix} b_1 \cos q_1 + b_2 \cos(q_1 + q_2) \\ b_2 \cos(q_1 + q_2) \end{bmatrix} \end{aligned}$$

$$a_1 = 200.01, \quad a_2 = 23.5, \quad a_3 = 122.5, \quad a_4 = 25, \quad b_1 = 784.8, \quad b_2 = 245.25$$

۲. ابتدا در سیمولینک سیستمی با تابع تبدیل زیر را پیاده سازی کنید.

$$G(z^{-1}) = \frac{(1 - 0.8z^{-1})}{(1 - 0.1z^{-1})(1 - 0.5z^{-1})(1 - 0.94z^{-1})} \quad (8.9)$$

ابتدا چند ورودی مختلف به این سیستم اعمال کنید و داده های مربوطه را ذخیره کنید. با فرض اینکه تنها داده ها را در اختیار دارید و از تابع تبدیل واقعی سیستم اطلاعی ندارید، بهترین سیستم مرتبه ۳ را برای این داده ها تخمین بزنید. از چه روشی (روش هایی) استفاده کردید؟ توضیح دهید. به ازای همان ورودی های بخش قبل، خروجی سیستمی که تخمین زده اید را با خروجی سیستم اصلی در یک نمودار رسم کنید.

با استفاده از داده ها یک سیستم مرتبه ۲ تخمین بزنید و مراحل سوال ارا تکرار کنید.

---

با استفاده از داده ها یک سیستم مرتبه ۱ تخمین بزنید و مراحل سوال ۱ را تکرار کنید.

اگر قرار باشد در یک پروژه واقعی چنین کاری انجام دهید، کدامیک از تخمین های فوق را برای مدل سازی سیستم و داده ها انتخاب می کنید. چرا؟



# آزمایش دهم (کنترل به کمک شبکه عصبی)

## ۱.۱۰ پیش گزارش

۱. روش کنترل به وسیله دینامیک معکوس را با جزییات بیان کنید.

۲. معاویب روش کنترل به وسیله دینامیک معکوس را بیان کنید.

راهکارهایی برای مقابله با معاویب کنترل به وسیله دینامیک معکوس بیان کنید.

## ۲.۱۰ مقدمه

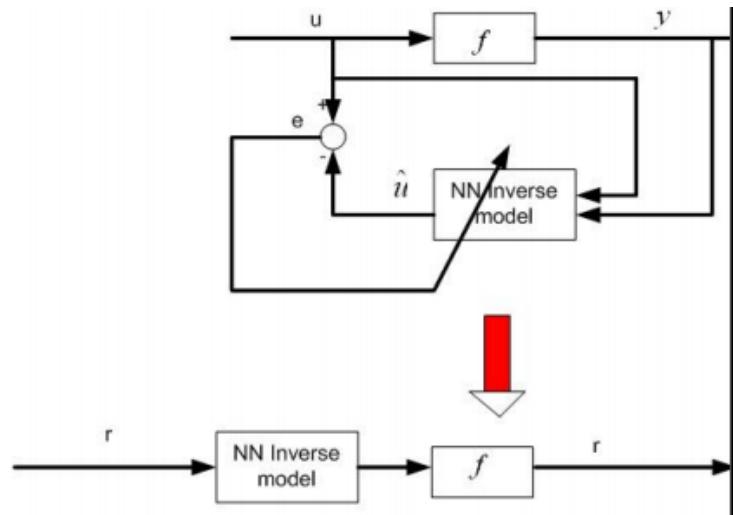
یکی از کاربردهای شبکه عصبی در کنترل یک سیستم ناشناخته است.

در این بخش به کنترل یک سیستم ناشناخته به وسیله شبکه عصبی پرداخته می شود.

یکی از روش های کنترل ، کنترل به وسیله دینامیک معکوس است . در ادامه به مروری کوتاه بر کنترل به وسیله دینامیک معکوس پرداخته می شود.

## ۱.۲.۱۰ کنترل دینامیک معکوس

ساختار کلی این روش در شکل نمایش داده شده است. در این روش ابتدا سیستم معکوس سیستم اصلی در صورت ناشناخته بودن شناسایی می شود.(قسمت بالای تصویر). با استفاده از یک شبکه عصبی می توان سیستم معکوس را شناسایی کرد(آزمایش هفتم) . سپس با توجه به اینکه  $I = f^{-1}f$  با سری کردن مدل معکوس و مدل اصلی خروجی سامانه ، خروجی مرجع را دنبال می کند.(قسمت پایین تصویر)



شکل ۱.۱۰: کنترل دینامیک معکوس

### ۳.۱۰ شرح آزمایش

در این آزمایش به کنترل یک سیستم ناشناخته با استفاده از دینامیک معکوس پرداخته می‌شود.  
مراحل زیر را دنبال کنید:

۱. ابتدا سیستم زیر را در محیط سیمولنک مطلب شبیه سازی کنید. قصد داریم این سیستم را با استفاده از کنترل دینامیک معکوس، کنترل کنیم. فرض می‌شود سیستم ناشناخته است. (از این مدل تنها برای ایجاد داده ورودی و خروجی برای شبکه عصبی استفاده می‌شود).

$$H(s) = \frac{40}{s + 1} \quad (1.10)$$

۲. با توجه با ساختار شناساگر پیاده سازی شده در آزمایش هفتم و با وارد کردن خروجی سامانه به عنوان ورودی شناساگر (به جای ورودی اصلی سامانه)، سیستم معکوس را شناسایی کنید. (توجه کنید برای ایجاد خطای ساختار شناساگر باید از اختلاف بین ورودی سامانه و خروجی شناساگر استفاده کنید)

۳. ساختار پیاده سازی شده در قسمت ۲ (شناسایی سیستم معکوس) را با ساختار سیستم اصلی (تابع تبدیل سیستم) سری کنید.

۴.  $\sin(t)$  را به عنوان ورودی به سیستم پیاده سازی شده (در قسمت ۳) اعمال کنید و خطای خروجی را رسم کنید.

## ۴.۱۰ تمرین

۱. به کمک مدل پیاده سازی شده در آزمایشگاه سعی کنید سیستم زیر را به وسیله دینامیک معکوس به گونه ای کنترل کنید که خرجی  $\sin(2t)$  را دنبال کند.

$$H(s) = \frac{1}{s - 1} \quad (2.10)$$

در صورت عملکرد نامناسب مدل ، علت را توضیح دهید.

۲. تابع  $(s^2 + 5s + 6)/(s + 1)$  را در مطلب پیاده سازی کنید. شبکه عصبی برای کنترل معکوس این آموزش دهید. سعی کنید با تابع پله شبکه را آموزش دهید. پاسخ را بررسی کنید و سعی کنید نویز به سیستم اضافه کنید. سعی کنید با تغییر ورودی سیستم هنگام /اموزش شبکه مودهای مختلف سیستم را تحریک کنید. اکنون پاسخ سیستم چگونه است. حساسیت نسبت به نویز چه تغییری کرده است ؟  
حال سعی کنید تابع  $(s^2 + 5s + 6)/(s - 1)$  را کنترل کنید. چه تغییری در توانایی شبکه به وجود می آید؟ علت چیست ؟

۳. تابع  $(s^2 + 5s + 6)/(s + 1)$  را در مطلب پیاده سازی کنید. شبکه عصبی برای کنترل معکوس این آموزش دهید. سعی کنید با تابع پله شبکه را آموزش دهید. پاسخ را بررسی کنید و سعی کنید نویز به سیستم اضافه کنید. سعی کنید با تغییر ورودی سیستم هنگام /اموزش شبکه مودهای مختلف سیستم را تحریک کنید. اکنون پاسخ سیستم چگونه است. حساسیت نسبت به نویز چه تغییری کرده است ؟  
حال سعی کنید تابع  $(s^2 + 5s + 6)/(s - 1)$  را کنترل کنید. چه تغییری در توانایی شبکه به وجود می آید؟ علت چیست ؟



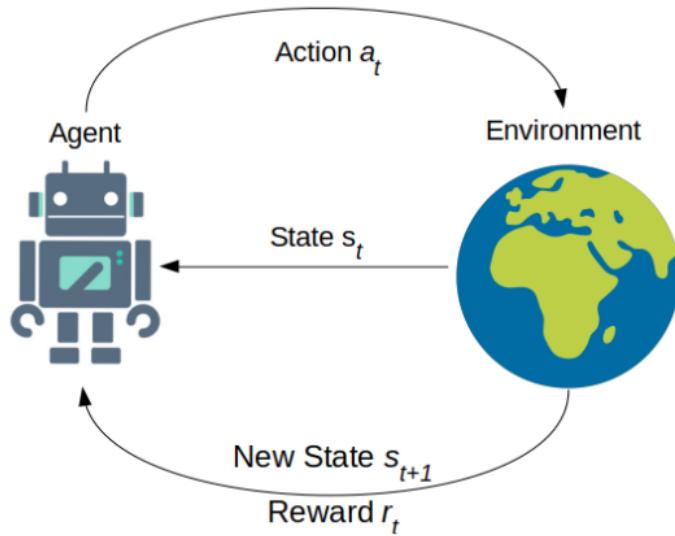
# آزمایش یازدهم (پیاده سازی الگوریتم $DQN$ )

## ۱.۱۱ پیش گزارش

۱. یادگیری تقویتی چه تفاوتی با سایر روش‌های یادگیری ماشین مانند یادگیری با ناظارت و بدون ناظارت دارد؟
۲. دلایل ادغام یادگیری عمیق با یادگیری تقویتی چیست؟ چند مثال ارائه دهید.
۳. بافر بازپخش(Replay Buffer) در Deep Q-Network (DQN) چگونه به فرآیند یادگیری کمک می‌کند و چرا مهم است؟
۴. مفهوم شبکه هدف در DQN و اهمیت آنها در فرآیند یادگیری را توضیح دهید.

## ۲.۱۱ مقدمه

یادگیری تقویتی، یکی از روش‌های یادگیری ماشین، فرآیندی است که در آن یک عامل آموزشی با هدف رسیدن به یک نتیجه مطلوب در محیط خاصی فعالیت می‌کند. در تضاد با یادگیری با ناظارت که در آن مدل‌های یادگیری با نمونه‌هایی از رفتارهای صحیح و نادرست آموزش داده می‌شوند، عامل در یادگیری تقویتی باید از طریق تجربه، آزمایش و کشف در محیط، روش‌های کسب حداکثر پاداش را بیاموزد. این عامل باید وضعیت‌های محیط را تحلیل کند و اقداماتی انجام دهد که به وضعیت‌های جدید منتهی شود. این عملکرد با دریافت پاداشی همراه است که در تکامل رفتارهای بعدی عامل نقش اساسی دارد. در مسائل پیچیده‌تر مانند سیستم‌های خودران، به دلیل وسعت بزرگ فضای حالت‌ها، از تلفیق قابلیت‌های یادگیری عمیق با یادگیری تقویتی، که به آن یادگیری تقویتی عمیق گفته می‌شود، استفاده می‌شود. در ادامه، به بررسی مبانی و اصول پایه‌ای یادگیری تقویتی پرداخته خواهد شد.



شکل ۱.۱۱: فرآیند تصمیم‌گیری مارکوف در یادگیری تقویتی

## ۱.۲.۱۱ مفاهیم اولیه یادگیری تقویتی

### فرآیند تصمیم‌گیری مارکوف

یک مسئله یادگیری تقویتی بصورت کلی به عنوان یک فرآیند تصمیم‌گیری مارکوف از نظر وضعیت، عمل، پاداش و پویایی سامانه توصیف می‌شود. در یک فرآیند تصمیم‌گیری مارکوف، در هر مرحله زمانی، عامل وضعیت فعلی  $s_t$  را مشاهده و عمل  $a_t$  را بر اساس سیاست فعلی‌اش انجام می‌دهد. پس از اجرای عمل، عامل پاداش و حالت بعدی را مشاهده می‌کند. هدف فرآیند تصمیم‌گیری مارکوف یافتن بهترین عمل‌هایی است که پاداش بلندمدت مورد انتظار آن را به حداقل می‌رساند. شکل ۱.۱۱ فرآیند اصلی فرآیند تصمیم‌گیری مارکوف را در یادگیری تقویتی را با یک عامل در تعامل با محیط اطرافش نشان می‌دهد.

### <sup>۱</sup>محیط

محیط یک شبیه‌ساز یا یک راهبرد واقعی است که در آن عامل تعامل و یادگیری دارد. در هر مرحله زمانی، عامل (که توسط سیاست اداره می‌شود) با محیط تعامل می‌کند و در ازای آن پاداشی دریافت می‌کند. محیط به دو دسته قابل مشاهده جزئی<sup>۲</sup> و کاملاً قابل مشاهده تقسیم می‌شود. در مورد یک محیط جزئی قابل مشاهده، عامل تنها قادر است تا حدی محیط را مشاهده کند. در یک محیط کاملاً قابل مشاهده، عامل می‌تواند تمام حالت‌ها را مشاهده کند.

<sup>1</sup>Environment

<sup>2</sup>Partially observable

عمل محركی است که عامل برای تعامل با محیط استفاده می‌کند. عمل‌ها می‌توانند بر اساس محیط و فرمول مسئله گستته یا پیوسته باشند.

### پاداش

پاداش یک مشوق است که با یک مقدار عددی بیان می‌شود که عامل پس از انجام یک عمل دریافت می‌کند. هدف یک عامل به حداقل رساندن پاداش انباسته است. مفهوم عامل تخفیف<sup>۳</sup> اساساً تعیین می‌کند که عوامل یادگیری تقویتی چقدر به پاداش‌ها در آینده دور نسبت به پادash‌های آینده نزدیک اهمیت می‌دهند. معمولاً عامل تخفیف با  $\gamma$  نشان داده می‌شود و می‌تواند هر مقداری از صفر تا یک داشته باشد. بازده<sup>۴</sup> مجموع تمام پادash‌هایی است که عامل انتظار دارد، هنگام پیروی از سیاست از یک حالت اولیه تا پایان اپیزود دریافت کند. از نظر ریاضی، بازده به صورت زیر است:

$$R_t = \sum_{t'=t}^T \gamma^{t'-t} r(s_t) \quad (1.11)$$

### تابع ارزش<sup>۵</sup>

تابع ارزش مقدار یک حالت را مشخص می‌کند. ارزش به عنوان حداقل بازده از یک حالت مشخص تعریف می‌شود. از نظر ریاضی می‌توان آن را به صورت رابطه‌ی (۲.۱۱) محاسبه کرد.

$$V_\pi(s) = E_\pi[R_t | s_t = s] \quad (2.11)$$

که در آن  $\pi$  سیاست،  $R_t$  بازده، و  $s_t$  وضعیت فعلی هستند.

### تابع کیو<sup>۶</sup>

تابع کیو مقدار یک حالت را مشخص می‌کند. مقدار به عنوان حداقل پاداش مورد انتظار تخفیفی که ممکن است یک عامل با انجام یک عمل خاص در یک وضعیت خاص دریافت کند، تعریف می‌شود. از نظر ریاضی می‌توان آن را به

<sup>3</sup>Discount factor

<sup>4</sup>Return

<sup>5</sup>Value function

<sup>6</sup>Q-function

صورت رابطه‌ی (۳.۱۱) محاسبه کرد.

$$Q_\pi(s, a) = E_\pi[R_t | s_t = s, a_t = a] \quad (3.11)$$

که در آن  $\pi$  سیاست،  $R_t$  بازده،  $s_t$  وضعیت فعلی، و  $a_t$  عمل جاری هستند.

## سیاست

سیاست نحوه رفتار عامل را در یک زمان خاص در محیط تعریف می‌کند. به عبارت دیگر، نقشه‌برداری از حالات درک شده محیط به عمل‌های انجام شده در آن شرایط است. یک سیاست زمانی بهینه گفته می‌شود که در هر حالت به حداقل پاداش ممکن دست یابد. سیاست‌ها بیشتر به دو نوع تقسیم می‌شوند: سیاست قطعی<sup>۷</sup> و سیاست تصادفی<sup>۸</sup>. هنگامی که عمل‌های انجام شده توسط عامل قطعی باشند، سیاست قطعی نامیده می‌شود. از سوی دیگر، زمانی که عمل‌ها از توزیع احتمال مشروط اقدامات با توجه به حالت‌ها نمونه‌برداری شوند، سیاست به صورت تصادفی نامیده می‌شود.

## اکتشاف<sup>۹</sup> و بهره‌برداری<sup>۱۰</sup>

اکتشاف فرآیندی است که در آن عامل سعی می‌کند با انجام عمل‌های مختلف موجود در یک وضعیت خاص، محیط اطراف را کشف کند. بهره‌برداری پس از اکتشاف صورت می‌گیرد. عامل از عمل‌های بهینه برای دستیابی به حداقل پاداش بهره‌برداری می‌کند. از یک سیاست اپسیلون-حریصانه<sup>۱۱</sup> برای ایجاد تعادل بین اکتشاف و بهره‌برداری استفاده می‌شود. عامل یک عمل تصادفی با احتمال مشخصی را انتخاب می‌کند در غیر این صورت عملی را انجام می‌دهد که سیاست را دنبال می‌کند. احتمال انجام عمل تصادفی با هر مرحله زمانی کاهش می‌یابد.

## DQN الگوریتم

هدف از یادگیری تقویتی یافتن سیاست بهینه است، یعنی سیاستی که حداقل بازده را بدهد. برای محاسبه سیاست، ابتدا تابع  $Q$  را محاسبه می‌شود. هنگامی که تابع  $Q$  را داریم، با انتخاب یک عمل در هر حالت که دارای حداقل مقدار

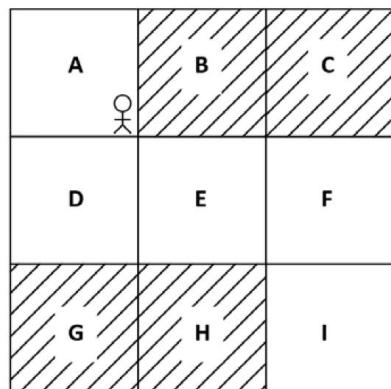
<sup>7</sup>Deterministic policy

<sup>8</sup>Stochastic policy

<sup>9</sup>Exploration

<sup>10</sup>Exploitation

<sup>11</sup> $\epsilon$ -greedy



شکل ۱۲.۱۱: محیط grid

است، سیاست را بدست می‌آوریم. به عنوان مثال، فرض کنید دو حالت  $A$  و  $B$  داریم و فضای عمل ما از دو عمل تشکیل شده است. بگذارید اقدامات بالا و پایین باشد. بنابراین، برای اینکه بفهمیم کدام عمل را در حالت  $A$  و  $B$  انجام دهیم، ابتدا مقدار  $Q$  همه جفت‌های حالت-عمل را محاسبه می‌کنیم، همانطور که جدول ۱.۱۱ نشان می‌دهد:

State	Action	Value
A	up	۱۷
A	down	۱۰
B	up	۱۱
B	down	۲۰

جدول ۱.۱۱: مقدار  $Q$  جفت‌های حالت-عمل

هنگامی که مقدار  $Q$  همه جفت‌های حالت-عمل را داشتیم، در هر حالت، عملی را انتخاب می‌کنیم که حداقل مقدار  $Q$  را دارد. بنابراین، عمل را در حالت  $A$  به بالا و در حالت  $B$  پایین را انتخاب می‌کنیم زیرا حداقل مقدار  $Q$  را دارند. ماتابع  $Q$  را در هر تکرار بهبود می‌دهیم و هنگامی که تابع  $Q$  بهینه را داشتیم، می‌توانیم سیاست بهینه را بدست آوریم. حال، همان‌طور که در شکل ۱۲.۱۱ نشان داده شده است، محیط grid را در نظر می‌گیریم.

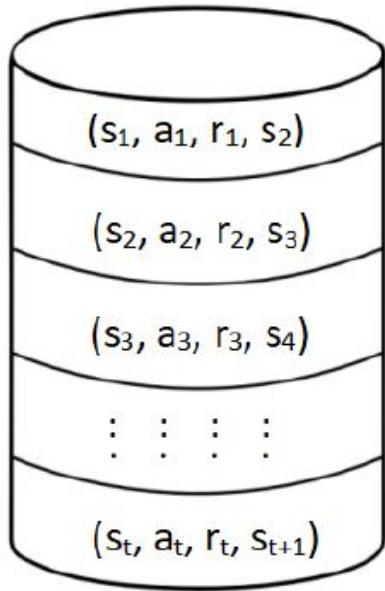
در محیط grid، هدف عامل رسیدن به حالت  $I$  از حالت  $A$  بدون عبور از حالت‌های سایه دار است و در هر حالت، عامل باید یکی از چهار عمل را انجام دهد - بالا، پایین، چپ، درست. برای بدست آوردن سیاست، ابتدا مقادیر  $Q$  همه جفت‌های حالت-عمل را محاسبه می‌کنیم. در اینجا، تعداد حالت‌ها ۹ است ( $A$  تا  $I$ ) و ما ۴ عمل در فضای عمل خود داریم، بنابراین جدول  $Q$  ما شامل  $9 \times 4 = 36$  ردیف است که حاوی مقادیر  $Q$  همه جفت‌های حالت-عمل ممکن است. هنگامی که مقادیر  $Q$  را بدست آوردهیم، با انتخاب عملی در هر حالت که دارای حداقل مقدار  $Q$  است، سیاست را استخراج می‌کنیم. اما آیا این روش خوبی است که مقدار  $Q$  را به طور کامل برای همه جفت‌های حالت-عمل محاسبه کنیم؟ بیایید این را با جزئیات بیشتر بررسی کنیم. فرض کنید محیطی داریم که در آن ۱۰۰۰ حالت و ۵۰ عمل ممکن در هر حالت داریم. در این مورد، جدول  $Q$  ما شامل  $1000 \times 50 = 50000$  سطر است که حاوی مقادیر  $Q$  همه جفت‌های حالت-عمل ممکن است. در مواردی مانند این، که در آن محیط ما از تعداد زیادی حالت و عمل تشکیل شده است،

محاسبه مقادیر  $Q$  تمام جفت‌های حالت-عمل ممکن به صورت جامع بسیار پرهزینه خواهد بود. آیا می‌توان به جای محاسبه مقادیر  $Q$  به این روش، آنها را با استفاده از هر تقریب‌گر تابعی، مانند شبکه عصبی، تقریب زد؟ بله، ما می‌توانیم تابع  $Q$  خود را با یک پارامتر  $\theta$  پارامتر کنیم و مقدار  $Q$  را در جایی که پارامتر  $\theta$  فقط پارامتر شبکه عصبی ما است، محاسبه کنیم. بنابراین، ما فقط حالت محیط را به یک شبکه عصبی داده و مقدار  $Q$  تمام اقدامات ممکن در آن حالت را بر می‌گرداند. هنگامی که مقادیر  $Q$  را به دست آورдیم، می‌توانیم بهترین عمل را به عنوان عملکردی که حداقل مقدار  $Q$  را دارد انتخاب کنیم. از آنجایی که ما از یک شبکه عصبی برای تقریب مقدار  $Q$  استفاده می‌کنیم، شبکه عصبی را شبکه  $Q$  می‌نامند و اگر از یک شبکه عصبی عمیق برای تقریب مقدار  $Q$  استفاده کنیم، شبکه عصبی عمیق را شبکه  $Q$  عمیق (DQN) می‌گویند. می‌توانیم تابع  $Q$  خود را با مشخص کنیم که پارامتر  $\theta$  در زیرنویس نشان می‌دهد که تابع  $Q$  ما با  $\theta$  پارامتر شده است و  $\theta$  فقط پارامتر شبکه عصبی ما است. پارامتر شبکه  $\theta$  را با مقادیر تصادفی مقداردهی اولیه می‌کنیم و تابع  $Q$  را تقریبی می‌زنیم (مقادیر  $Q$ )، اما از آنجایی که  $\theta$  را با مقادیر تصادفی مقداردهی اولیه کردیم، تابع  $Q$  تقریبی بهینه نخواهد بود. بنابراین، با یافتن پارامتر بهینه  $\theta$ ، شبکه را برای چندین تکرار آموزش می‌دهیم. هنگامی که  $\theta$  بهینه را پیدا کردیم، تابع  $Q$  بهینه را خواهیم داشت. سپس می‌توانیم سیاست بهینه را از تابع  $Q$  بهینه استخراج کنیم.

## ۱۲ بافر بازپخش

داده‌های آموزشی شبکه عصبی از بافری به نام Replay buffer که برای جمع آوری تجربیات عامل استفاده می‌شود، بدست می‌آیند و بر اساس این تجربه، شبکه خود را آموزش می‌دهیم. عامل با انجام یک عمل  $a$  از حالت  $s$  به حالت بعدی  $s'$  انتقال می‌دهد و سپس یک پاداش  $r$  دریافت می‌کند. ما می‌توانیم این اطلاعات انتقال  $(s, a, r, s')$  را در بافری به نام بافر تکرار یا بازپخش تجربه ذخیره کنیم. بافر بازپخش معمولاً با  $\mathcal{D}$  نشان داده می‌شود. این اطلاعات انتقال اساساً تجربه عامل است. ما تجربه عامل را در چندین قسمت در بافر پخش ذخیره می‌کنیم. ایده کلیدی استفاده از بافر replay برای ذخیره تجربه عامل این است که ما می‌توانیم DQN خود را با تجربه (انتقال) نمونه برداری شده از بافر آموزش دهیم. یک بافر پخش مجدد در شکل ۳.۱۱ نشان داده شده است.

<sup>12</sup>Replay buffer



شکل ۳.۱۱: Replay buffer

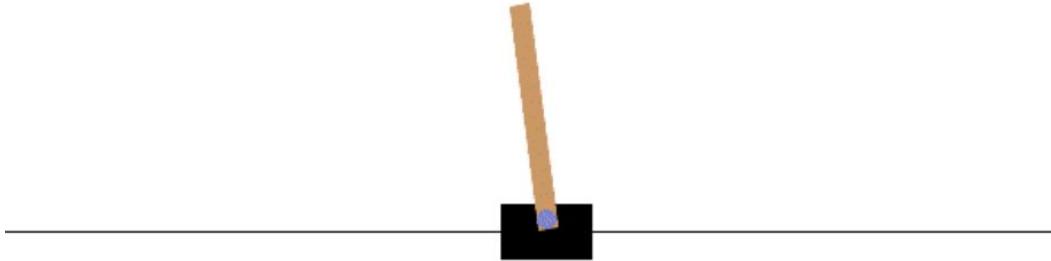
### تابع هزینه<sup>۱۳</sup>

ما یاد گرفتیم که در DQN هدف ما پیش‌بینی مقدار  $Q$  است که فقط یک مقدار پیوسته است. بنابراین، در DQN ما اساساً یک کار رگرسیون را انجام می‌دهیم. ما معمولاً از میانگین مربعات خطأ (MSE) به عنوان تابع زیان برای کار رگرسیون استفاده می‌کنیم. MSE را می‌توان به عنوان میانگین اختلاف مجذور بین مقدار هدف و مقدار پیش‌بینی شده تعریف کرد، همان‌طور که در اینجا نشان داده شده است:

$$MSE = \frac{1}{K} \sum_{i=1}^K (y_i - \hat{y}_i)^2 \quad (4.11)$$

جایی که  $y$  مقدار هدف است،  $\hat{y}$  مقدار پیش‌بینی شده و  $K$  تعداد نمونه‌های آموزشی است. شبکه کد الگوریتم DQN در الگوریتم ۱ آمده است.

<sup>13</sup>Loss function



شکل ۴.۱۱: Cart-pole

---

#### **Algorithm 1** Deep Q-learning with Experience Replay

---

```

Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1 to  $M$  do
    Initialize sequence  $s_1 = \{x_1\}$  and preprocess  $\phi_1 = \phi(s_1)$ 
    for  $t = 1$  to  $T$  do
        With probability  $\epsilon$ , select a random action  $a_t$ 
        Otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
        Set  $s_{t+1} = s_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
        Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $\mathcal{D}$ 
        Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
        Set  $y_j = \begin{cases} r_j & \text{terminal for } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{non-terminal for } \phi_{j+1} \end{cases}$ 
        Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$ 
    end for
end for

```

---

## ۳.۱۱ شرح آزمایش

در این مرحله به پیاده‌سازی DQN برای cart-pole در محیط gym با استفاده از زبان برنامه‌نویسی پایتون پرداخته می‌شود. همان‌طور که در شکل ۴.۱۱ مشاهده می‌شود، آونگ به صورت عمودی روی گاری قرار می‌گیرد و هدف این است که با اعمال نیرو در جهت چپ و راست بر روی گاری، میله را متعادل کند. همچنین مشخصات آن نیز در جدول ۲.۱۱ آمده است.

از آنجایی که هدف این است که میله را تا زمانی که ممکن است عمودی نگه دارد، برای هر قدم برداشته شده، از جمله مرحله پایان، پاداش  $+1$  در نظر گرفته می‌شود. آستانه پاداش برای ۴۷۵ است. برای پیاده‌سازی DQN برای

<b>Action Space</b>	<i>Discrete(2)</i>
<b>Observation Shape</b>	$(4, )$
<b>Observation High</b>	$[4.8, inf, 0.42, inf]$
<b>Observation Low</b>	$[-4.8, -inf, -0.42, -inf]$

جدول ۲.۱۱: جدول مشخصات عمل و حالات

مسئله episode از شبه کد الگوریتم DQN در استفاده کرده و بعد از هر episode ، شماره epsilon را چاپ کنید. همچنین نمودار return برحسب episode را نیز رسم کنید.

## ۴.۱۱ تمرین

۱. مسئله cart-pole را با استفاده از الگوریتم DQN در محیط متلب پیاده‌سازی کنید.
۲. یک بار دیگر مسئله cart-pole را با استفاده از الگوریتم (DDPG) که در آزمایش انجام شد بهتر است یا الگوریتم DQN ؟ چرا؟



## پروژه بخش عصبی

### ۱.۱۲ پیش گزارش

۱. با مراجعه به اینترنت معادلات دینامیک مربوط به خودرو بدون سرنشین چهار چرخ را بیان کنید.
۲. با مراجعه به اینترنت معادلات دینامیک مربوط به ربات نقاش ۲ بعدی را بیان کنید.

### ۲.۱۲ مقدمه

در آزمایش های گذشته به بررسی انواع مختلف شبکه عصبی و طراحی کنترلر و شناساگر با استفاده از شبکه عصبی پرداخته شد . در این آزمایش قصد داریم به پیاده سازی سخت افزاری الگوریتم های بیان شده بپردازیم. برای پیاده سازی سخت افزاری از خودرو زمینی چهار چرخ بدون سرنشین<sup>۱</sup> و یا ربات نقاش ۲ بعدی استفاده می شود. در ادامه به معرفی کوتاه ربات نقاش ۲ بعدی و خودرو زمینی بدون سرنشین پرداخته می شود.

### ۱.۲.۱۲ خودرو زمینی بدون سرنشین

خودرو زمینی بدون سرنشین یک ربات ۴ چرخ با مکانیزم حرکت تانکی و خودکار بوده که قابلیت طی مسیر به صورت خودکار و ارتباط با عامل های دیگر از طریق بی سیم را نیز دارد. این ربات مجهز به حلقه ۶ تایی سونار برای مانع یابی و همچنین ۲ عدد انکوادر جهت مکان یابی بوده و قابلیت نصب سنسورهای مختلفی در آن در نظر گرفته شده است. توضیحات مفصل این ربات در (پیوست ۱) انجام شده است. (حتما پیش از آزمایش، پیوست مطالعه شود). تصویر این ربات در شکل نمایش داده شده است.

---

<sup>1</sup> UGV(unmanned – ground – veichle)



شکل ۱.۱۲: خودروی زمینی بدون سرنشین

## ۲.۲.۱۲ ربات نقاش ۲ بعدی

ربات نقاش یک ربات دو بعدی با دو درجه آزادی ، همانطور که در شکل مشاهده می شود، این ربات از چهار لینک (بازو) که از یک طرف به دو محرک (موتور)  $DC$  و از طرف دیگر به یکدیگر متصل شده اند تشکیل شده است. نقطه اتصال لینک ها به هم سر موثر نماید که با اعمال فرمان مناسب به موتورها ، سر موثر به نقاط مطلوب در صفحه  $x - y$  حرکت داده می شود. سر موثر شامل یک قلم میباشد که این قلم جهت بر جای گذاشتن مسیر حرکت سر موثر به کار رفته است.



شکل ۲.۱۲: ربات نقاش ۲ بعدی

## ۳.۱۲ شرح آزمایش

در این بخش ۳ آزمایش برای پیاده سازی بر روی ربات نقاش ۲ بعدی و خودرو بدون سرنشین در نظر گرفته شده است . از بین این ۳ آزمایش ۱ آزمایش را انتخاب کرده و به پیاده سازی سخت افزاری آن بپردازید.

## ۱.۳.۱۲ آزمایش ۱ - شناسایی خودرو زمینی بدون سرنشین

در این قسمت قصد داریم با استفاده از ساختار شناساگر معرفی شده در آزمایش ۷ به شناسایی خودروی زمینی بدون سرنشین بپردازیم.

برای شناسایی خودرو زمینی بدون سرنشین مراحل زیر را دنبال کنید:

۱. برای پیاده سازی شناساگر لازم است ابتدا از صحت شناساگر طراحی شده در آزمایش ۷ اطمینان حاصل کنید.  
تابع تبدیل ساده شده خودرو بدون سرنشین به صورت رابطه ۱.۹ است. ابتدا شناساگر طراحی شده در آزمایش ۷ را بر روی این مدل شبیه سازی کنید و از صفر شدن خطای بین خروجی شناساگر و خروجی اصلی سامانه اطمینان حاصل کنید.

$$H(s) = \frac{40}{.02s^2 + s} \quad (1.12)$$

۲. فایل سیمولینک مربوط به خودرو بدون سرنشین در اختیارتان قرار داده می شود. فایل را در محیط سیمولینک باز کنید.

۳.  $pwm_1$  و  $pwm_2$  را برابر با تابع  $\sin(t)$  قرار دهید.

۴. بعد از اطمینان از صحت شناساگر پیاده سازی شده در آزمایش ۷ با استفاده از شناساگر طراحی شده و در نظر گرفتن  $pwm_1$  و  $pwm_2$  خودروی بدون سرنشینی به عنوان ورودی شناساگر و موقعیت خودروی بدون سرنشین به عنوان خروجی (موقعیت در راستای  $x$  و یا  $y$ ) سعی کنید سیستم را شناسایی کنید.

۵. خطای بین خروجی شناساگر (موقعیت تخمین زده خودرو بدون سرنشین) و موقعیت خودرو بدون سرنشین را رسم کنید.

## ۲.۳.۱۲ آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی

برای شناسایی ربات نقاش ۲ بعدی مراحل زیر را دنبال کنید:

۱. ابتدا فایل ربات را که در اختیارتان قرار گرفته است را اجرا کنید. با اجرا کردن این فایل ربات شروع به نقاشی یک دایره خواهد کرد.

۲. این ربات ۲ سیگنال را از ورودی دریافت می کند(سیگنال های  $u_1, u_2$  در فایل سیمولینک). با در نظر گرفتن این ۲ سیگنال به عنوان ورودی شناساگر و زاویه بازوها به عنوان خروجی شناساگر ، سیستم را شناسایی کنید.خطای بین خروجی شناساگر و خروجی اصلی سیستم را رسم کنید.

### ۳.۳.۱۲ آزمایش ۳ - کنترل خودروی زمینی بدون سرنوشت

در این قسمت به پیاده سازی سخت افزاری الگوریتم کنترل شبیه سازی شده (در آزمایش ۸) بر روی خودرو زمینی بدون سرنوشت پرداخته می شود.

قصد داریم خودرو بدون سرنوشت را به گونه ای کنترل کنیم که بر روی خط  $x = 10$  (یا  $y = 10$ ) حرکت کند. برای کنترل خودروی بدون سرنوشت مراحل زیر را دنبال کنید:

۱. ابتدا از صحت کنترلر پیاده سازی شده در آزمایش ۸ اطمینان حاصل کنید.

مدل ساده شده خودرو بدون سرنوشت به صورت رابطه ۱.۹ است. ابتدا کنترلر طراحی شده در آزمایش ۸ را برروی این مدل شبیه سازی کنید. ورودی مرجع را برابر با یک عدد ثابت در نظر بگیرید. در صورت عملکرد صحیح کنترلر خطای بین خروجی سامانه و ورودی مرجع به سمت صفر میل می کند.

۲. فایل سیمولینک مربوط به کنترل خودرو بدون سرنوشت در اختیارتان قرار داده می شود. فایل را در محیط سیمولینک باز کنید.

۳. ورودی مرجع کنترلر را برابر با  $10$  در نظر بگیرید . مقدار  $pwm_1(pwm_2)$  را برابر با خروجی کنترلر عصبی قرار دهید. موقعیت خودرو بدون سرنوشت را نیز به عنوان ورودی کنترلر در نظر بگیرید. خطای بین ورودی مرجع و موقعیت خودرو بدون سرنوشت را رسم کنید.

### ۴.۱۲ تمرین

۱. با توجه به خروجی های حاصل شده در بخش عملی ، ضعف های عملکرد الگوریتم پس انتشار خطا را در حالت گسسته بیان کنید.

## آزمایش دوازدهم (آشنایی با مفاهیم فازی)

### ۱.۱۳ پیش گزارش

۱. تفاوت دسته بندی ترد<sup>۱</sup> را با دسته بندی فازی بیان کنید.

۲. مفهوم  $S - Norm$  در منطق فازی چیست؟

۳. مفهوم  $T - Norm$  در منطق فازی چیست؟

### ۲.۱۳ مقدمه

برای دسته بندی برخی از خصوصیات و اشیا نمی توان یک مرز مشخص در نظر گرفت ، فرض کنید میخواهیم جوان یا پیر بودن انسان ها را مشخص کنیم . تمام انسان ها را نمی توان در ۲ دسته پیر و جوان قرار داد برای مثال یک انسان میانسال را در نظر بگیرید چنانی انسانی ۵۰ درصد جوان و ۵۰ درصد پیر محسوب می شود به همین ترتیب ممکن است یک انسان ۳۰ درصد جوان و ۷۰ درصد پیر محسوب شود. این گونه دسته بندی که در آن مرز مشخصی بین دسته ها در نظر گرفته نمی شود دسته بندی و یا منطق فازی گفته می شود.

در این آزمایش قصد داریم به پیاده سازی مفاهیم اولیه فازی در محیط متلب بپردازیم. در ابتدا به مروری کوتاه بر مفاهیم اولیه فازی پرداخته می شود.

<sup>1</sup> crisp

## ۱.۲.۱۳ تابع عضویت

تابع عضویت یک تابع پیوسته است. خروجی تابع عضویت در بازه  $[0, 1]$  قرار دارد. خروجی تابع عضویت نشان دهنده احتمال متعلق بودن ورودی به کلاس در نظر گرفته شده است.

## ۲.۲.۱۳ نقطه عبور

نقطه عبور<sup>۲</sup> به نقطه‌ای گفته می‌شود که مقدار تابع عضویت در آن نقطه برابر با ۰.۵ است.

## ۳.۲.۱۳ عملگرها در منطق فازی

برای کار با مجموعه‌ها در منطق فازی لازم است عملیات‌های اصلی مانند متمم، اشتراک و اجتماع تعریف شوند. در ادامه به بررسی برخی از عملگرهای اصلی در منطق فازی پرداخته می‌شود.

### مکمل

عملگر مکمل عملگری است که  $\mu_A(\text{تابع عضویت } A)$  را به تابع عضویت مکمل آن  $\mu_{\bar{A}}$  ارتباط می‌دهد. در منطق فازی رابطه‌های گوناگونی برای مکمل یک مجموعه تعریف شده است، از معروف ترین روابط تعریف شده برای مکمل می‌توان به رابطه Sugeno که در معادلات زیر توصیف شده است اشاره کرد. (در رابطه ۱.۱۰)  $\lambda \in (-1, \inf)$  در نظر گرفته شده است

$$c_{\lambda_a} = \frac{1 - \mu_A(x)}{1 + \lambda} \quad (1.13)$$

## ۴.۲.۱۳ اجتماع

اجتماع ۲ مجموعه در منطق فازی با  $S - Norm$  دو مجموعه مشخص می‌شود. مشابه مکمل گیری برای اجتماع ۲ مجموعه در منطق فازی نیز تعاریف متفاوتی وجود دارد. یکی از معروف ترین تعاریف  $S - Norm$  در منطق فازی Dombi تعریف که مطابق با رابطه ۱۰.۲ تعریف می‌شود است. (در رابطه ۲.۱۰)  $\lambda \in (0, \inf)$  در نظر گرفته شده است

$$S_\lambda(\mu_A(x), \mu_B(x)) = \frac{1}{1 + ((\frac{1}{\mu_A(x)} - 1)^{-\lambda} + (\frac{1}{\mu_B(x)} - 1)^{-\lambda})^{-\frac{1}{\lambda}}} \quad (2.13)$$

Cross – over<sup>۲</sup>

اشتراك ۲ مجموعه در منطق فازی با  $T - Norm$  و  $S - Norm$  برای  $Dombi$  نیز در منطق فازی تعاریف متفاوتی وجوددارد. یکی از معروف ترین تعاریف  $S - Norm$  در منطق فازی تعریف که مطابق با رابطه ۳.۱۰ تعریف می شود است.(در رابطه  $\lambda \in (0, \inf)$  در نظر گرفته شده است)

$$T_\lambda(\mu_A(x), \mu_B(x)) = \frac{1}{1 + ((\frac{1}{\mu_A(x)} - 1)^\lambda + (\frac{1}{\mu_B(x)} - 1)^\lambda)^{\frac{1}{\lambda}}} \quad (3.13)$$

### ۳.۱۳ شرح آزمایش

در این آزمایش قصد داریم ابتدا ۲ مجموعه فازی را در محیط متلب پیاده سازی کنیم ، بعد از پیاده سازی ۲ مجموعه به پیاده سازی عملگرهای فازی مانند مکمل ،  $T - Norm$  و  $S - Norm$  پرداخته می شود. برای ایجاد توابع عضویت فازی مراحل زیر را دنبال کنید:

۱. مجموعه جهانی را برابر با  $U = 0 : 0.01 : 1$  در نظر بگیرید.

۲. دستور  $gaussmf$  در متلب یک تابع عضویت فازی براساس تابع عضویت گوسی تولید می کند. ورودی های دستور  $gaussmf$  داده های ورودی و میانگین و انحراف معیار در نظر گرفته شده برای تابع گوسین است. با استفاده از دستور  $gaussmf$  تابع عضویت  $A$  را به صورت زیر تعریف کنید:

$$A = gaussmf(U, [0.350]);$$

۳. با استفاده از دستور  $gaussmf$  تابع عضویت  $B$  را به صورت زیر تعریف کنید:

$$B = gaussmf(U, [0.350]);$$

در ادامه قصد داریم عملیات های مکمل ( $Sugno$ ) و  $(Makhl)$  را پیاده سازی کنیم.

برای پیاده سازی و تست مکمل  $Sugno$  مراحل زیر را دنبال کنید:

۱. با در نظر گرفتن رابطه ۱.۱۰ تابعی بنویسید که یک تابع عضویت و یک مقدار در بازه  $(-\infty, 1)$  (مقدار  $\lambda$  در معادله ۱.۱۰) را از ورودی دریافت کند و سپس مقدار مکمل آن را(با توجه به رابطه ۱.۱۰) بازگرداند.

۲. تابع عضویت  $A$  را که در قسمت قبل پیاده سازی کردید به عنوان ورودی به تابع مکمل  $Sugno$  دهید و خروجی را رسم کنید.

۳. تابع عضویت  $B$  را که در قسمت قبل پیاده سازی کردید به عنوان ورودی به تابع مکمل  $Sugno$  دهید و خروجی را رسم کنید.

برای پیاده سازی و تست  $S - Norm Dombi$  مراحل زیر را دنبال کنید.

۱. با در نظر گرفتن رابطه ۳.۱۰ تابعی بنویسید که یک دو تابع عضویت ( $\mu_A$  و  $\mu_B$  در رابطه ۳.۱۰) و یک مقدار در بازه  $(0, \infty)$  (مقدار  $\lambda$  در معادله ۳.۱۰) را از ورودی دریافت کند و سپس مقدار  $S - Norm Dombi$  آن را (با توجه به رابطه ۳.۱۰) بازگردداند.

۲. توابع عضویت  $A$  و  $B$  را که در قسمت های قبل پیاده سازی کردید به عنوان ورودی به تابع  $S - Norm Dombi$  دهید و خروجی را رسم کنید.

برای پیاده سازی و تست  $T - Norm Dombi$  مراحل زیر را دنبال کنید:

۱. با در نظر گرفتن رابطه ۲.۱۰ تابعی بنویسید که یک دو تابع عضویت ( $\mu_A$  و  $\mu_B$  در رابطه ۲.۱۰) و یک مقدار در بازه  $(0, \infty)$  (مقدار  $\lambda$  در معادله ۲.۱۰) را از ورودی دریافت کند و سپس مقدار  $T - Norm Dombi$  آن را (با توجه به رابطه ۲.۱۰) بازگردداند.

۲. توابع عضویت  $A$  و  $B$  را که در قسمت های قبل پیاده سازی کردید به عنوان ورودی به تابع  $T - Norm Dombi$  دهید و خروجی را رسم کنید.

## ۴.۱۳ تمرین

۱. همان طور که می دانید انواع مختلفی از  $S - Norm$  در منطق فازی وجود دارد . یکی از معروف ترین انواع توصیف شده است. در رابطه ۴.۱۰  $Einstein - sum$  است. در رابطه ۴.۱۰  $Einstein - sum$ ,  $S - Norm$  را در محیط متلب پیاده سازی کنید. توابع عضویت  $A$  و  $B$  را به صورت زیر تعریف کنید.

$$U = 0 : 0.01 : 1;$$

$$A = gaussmf(U, [0.350]);$$

$$B = gaussmf(U, [0.350]);$$

توابع عضویت  $A$  و  $B$  را به عنوان ورودی به تابع  $Einstein - sum$  دهید و خروجی را رسم کنید.

$$S_{es}(\mu_A, \mu_B) = \frac{\mu_A + \mu_B}{1 + \mu_A \mu_B} \quad (4.13)$$

۲. همان طور که می دانید انواع مختلفی از  $T - Norm$  از  $T - Norm$  در منطق فازی وجود دارد . یکی از معروف ترین انواع  $T - Norm$  توصیف شده است. در رابطه  $5.10$   $Einstein - product$   $Einstein - product$  را در محیط مطلب پیاده سازی کنید. توابع عضویت  $A$  و  $B$  را به صورت زیر تعریف کنید.
- $$U = 0 : 0.01 : 1;$$
- $$A = gaussmf(U, [0.350]);$$
- $$B = gaussmf(U, [0.350]);$$
- توابع عضویت  $A$  و  $B$  را به عنوان ورودی به تابع  $Einstein - product$  دهید و خروجی را رسم کنید.

$$S_{es}(\mu_A, \mu_B) = \frac{\mu_A \mu_B}{2 - (\mu_A + \mu_B - \mu_A \mu_B)} \quad (5.13)$$



# آزمایش سیزدهم(کنترلر فازی)

## ۱.۱۴ پیش گزارش

۱. برای ایجاد یک سیستم فازی به مجموعه ای از قوانین اگر-آن گاه احتیاج است. به نظر شما چگونه می توان قوانین فازی مناسب برای کنترل یک سیستم را ایجاد کرد؟
۲. در یک سیستم فازی می توان چند قانون مختلف اگر-آنگاه را با یکدیگر ترکیب کرد. به نظر شما چه منطقی برای ترکیب کردن قوانین مختلف در سیستم فازی باید در نظر گرفته شود؟

## ۲.۱۴ مقدمه

در این آزمایش قصد داریم با استفاده از جعبه ابزار فازی<sup>۲</sup> مطلب به پیاده سازی یک کنترلر فازی بپردازیم. در ابتدا مروری بر چگونگی کارکرد جعبه ابزار فازی مطلب خواهیم داشت.

## ۱.۲.۱۴ جعبه ابزار فازی مطلب

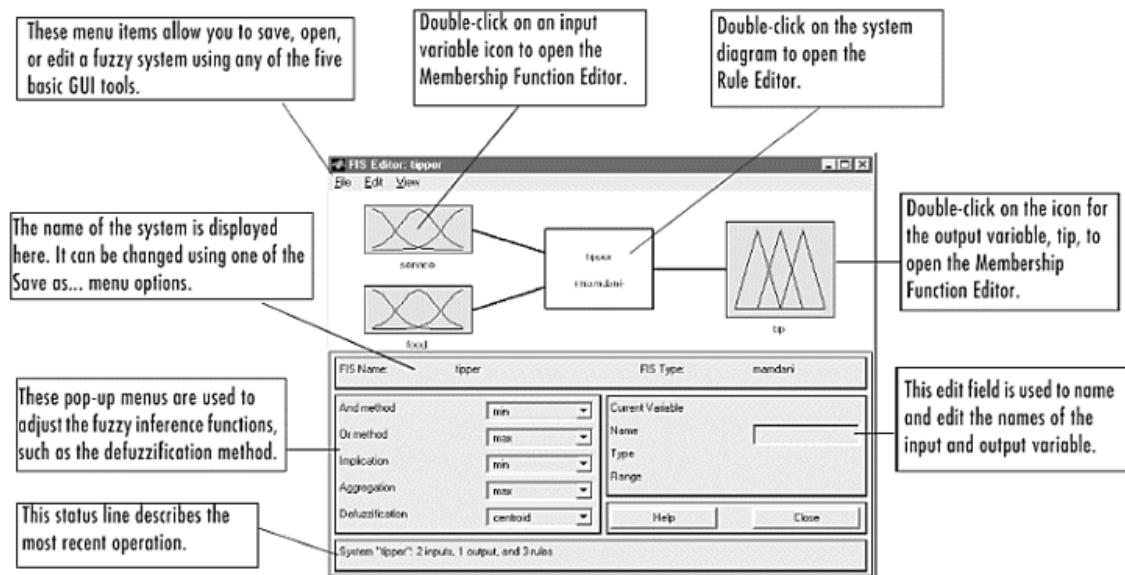
در *fuzzy Logic Designer* مطلب دستور *Command-Windows* را تایپ کنید. پنجره‌ای مشابه شکل ۱.۱۱ باز می‌شود. از منوی *Edit* می‌توانید تعداد ورودی و خروجی‌ها و نوع آن‌ها را مشخص کنید. با کلیک بر روی توابع عضویت ورودی و خروجی می‌توانید تابع عضویت ورودی‌ها و خروجی‌ها را مشخص کنید.

در قسمت مربع شکل پایین صفحه همان طور که در شکل ۱.۱۱ می‌بینید می‌توان توابع مورد نیاز (*fuzzy-interface*) را مشخص کرد. در قسمت مربع شکل پایین سمت راست می‌توان نام ورودی و خروجی‌هارا تغییر داد.

---

<sup>1</sup>*If – Then*

<sup>2</sup>*fuzzy – toolbox*

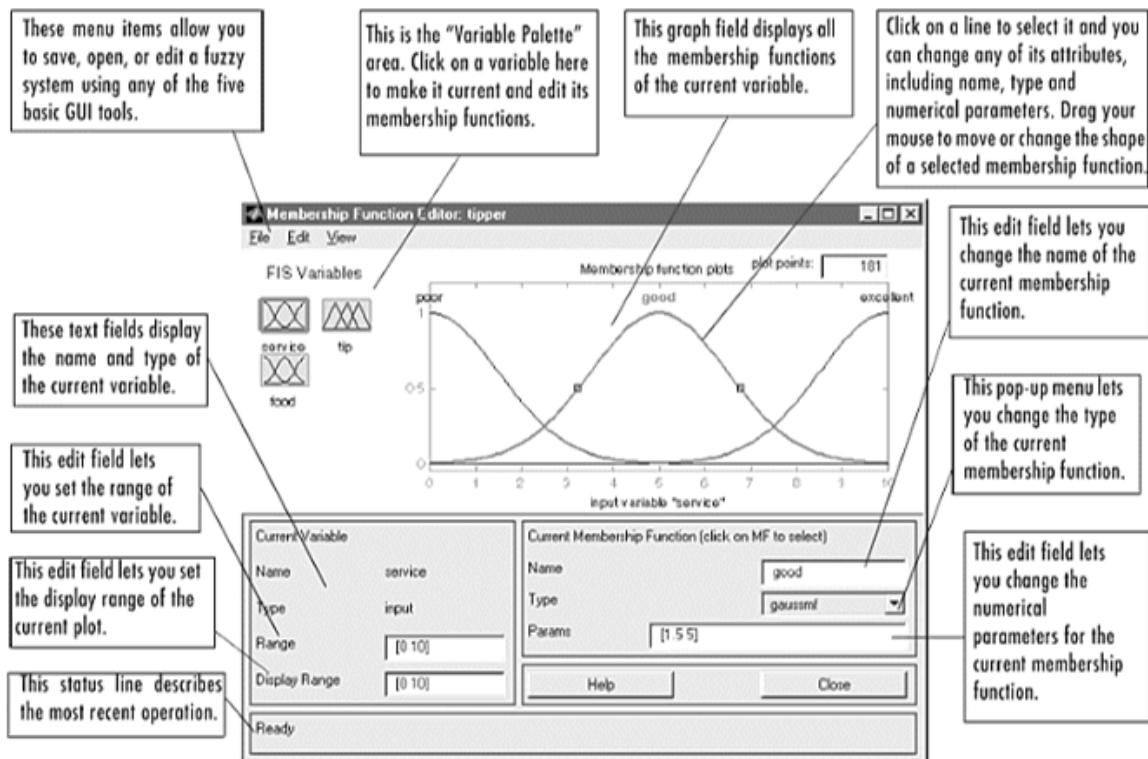


شکل ۲.۱۱۴: محیط اصلی ابزار طراحی فازی

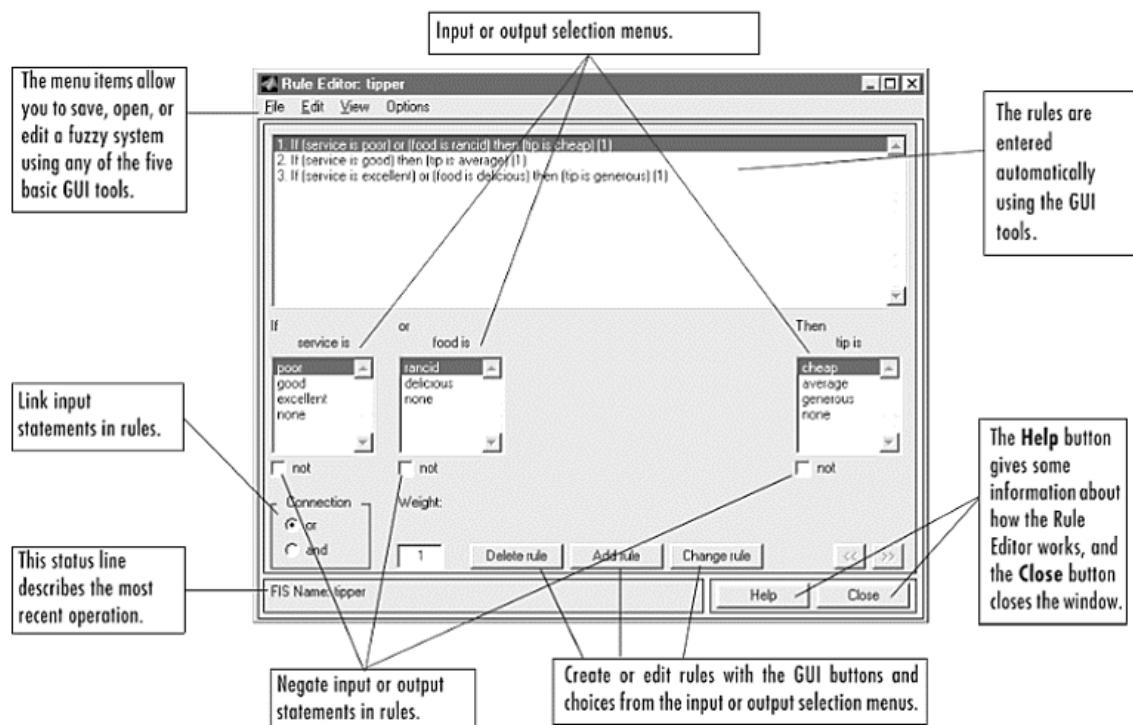
قسمت توابع عضویت پنچره‌ای همانند شکل ۲.۱۱ دارد. در مربع پایین همانطور که در شکل ۲.۱۱ مشخص است می‌توان بازه ورودی و خروجی را تعیین کرد. برای تغییر مشخصات هر کدام از خطها ورودی یا خروجی کافی است بر روی آن کلیک کنید.

در گراف بالای صفحه نیز می‌توان تمام توابع عضویت را مشاهده کرد. در مربع پایین سمت راست نیز قابلیت تصحیح نام تابع عضویت انتخاب شده را دارد. در این قسمت همچنین می‌توان مقادیر عددی تابع عضویت را تغییر داد. پنچره تنظیم قوانین به صورت شکل ۳.۱۱ است با استفاده از مربعات پایین صفحه به راحتی می‌توان قوانین فازی را به صورت اگر-آنگاه پیاده‌سازی کرد در مربعات سمت راست حالت ورودی با توجه به تابع عضویت تعریف شده و در مربع سمت چپ همان طور که در شکل ۳.۱۱ نشان داده شده است حالت خروجی با توجه به تابع عضویت مشخص می‌شود.

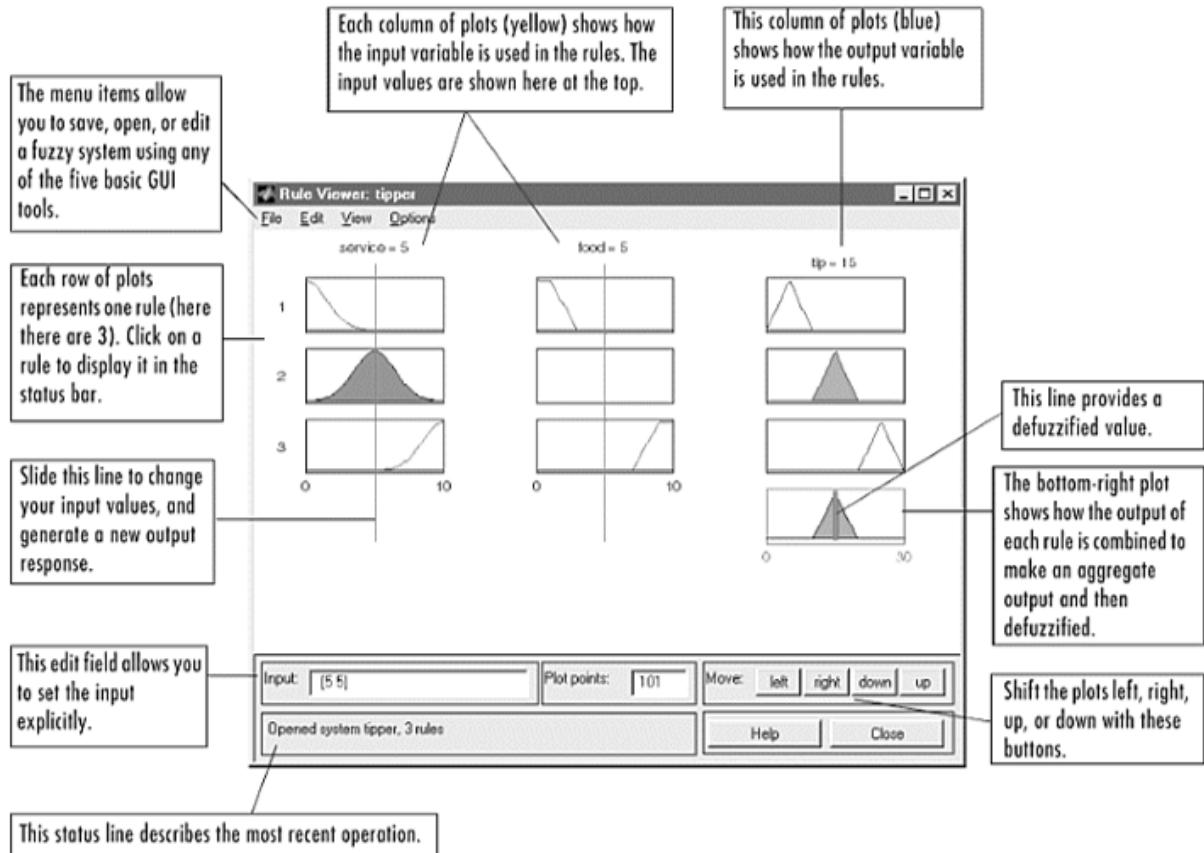
در قسمت *rule – viewer* می‌توان غیرفازی ساز را مانند شکل ۴.۱۱ در ستون سوم مشخص کرد.



شكل ۲.۱۴: تعریف توابع عضویت در ابزار فازی متلب



شكل ۳.۱۴: تعریف قوانین فازی در ابزار فازی متلب



شکل ۴.۱۴: مشاهده کردن قوانین در ابزار فازی متلب

## ۳.۱۴ شرح آزمایش

در این قسمت هدف، طراحی یک کنترلر فازی برای سرعت خودرویی است که دینامیک در حال حرکت آن به صورت زیر تعریف می‌شود.

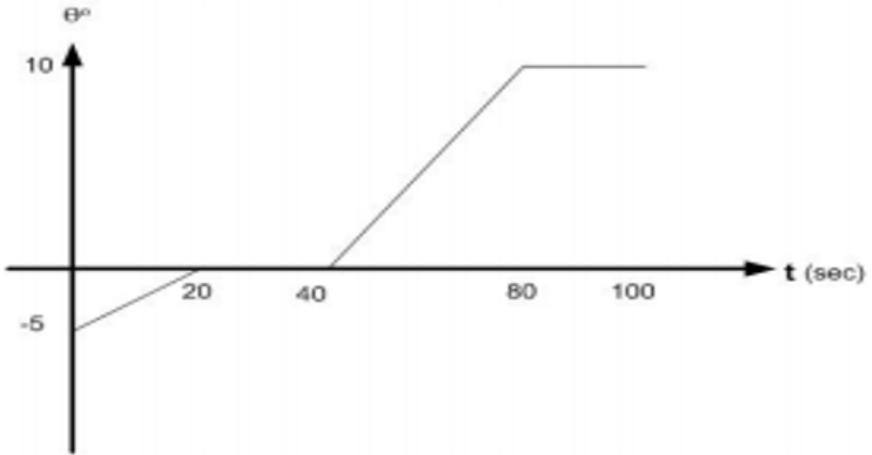
$$\dot{v} = -.1v + [-1, .1][T, \theta]^T \quad (1.14)$$

$T$  و ورودی‌های سیستم بیانگر زاویه پدال گاز و شیب جاده است. هدف طراحی یک کنترلر فازی برای سیستم توصیف شده است به طوری که با گرفتن فیدبک از سرعت کنونی خودرو ( $v$ ) و با توجه به شیب جاده بتواند سرعت خودرو را در  $50\text{ km/h}$  نگه دارد. (ورودی کنترل کننده فازی را خطأ و مشتق خطای خروجی سیستم در نظر بگیرید.) شیب جاده نیز در شکل ۵.۱۱ توصیف شده است.

ساختار کلی کنترلر فازی در شکل نمایش ۶.۱۱ داده شده است.

برای طراحی کنترلر فازی مراحل زیر را دنبال کنید:

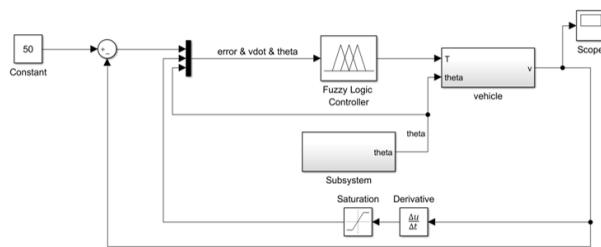
- تابع عضویت ورودی را تعریف کنید. توجه کنید ورودی دارای ۳ حالت آرام و متوسط و سریع است.



شکل ۵.۱۴: نمودار شیب جاده بر حسب زمان

۲. با توجه به ورودی قوانین فازی را تعریف کنید. پیشنهاد می شود برای خروجی فازی ساز ۳ حالت آرام و عادی و سریع در نظر بگیرید.

۳. توابع عضویت خروجی را تعریف کنید



شکل ۶.۱۴: ساختار کلی کنترلر فازی

## ۶.۱۴ تمرین

۱. عملکرد کنترلر فازی طراحی شده در آزمایشگاه در حالتی که خطای یک سیگنال نویزی است شبیه سازی کنید. (نویز را به صورت نویز سفید در نظر بگیرید).

۲. کنترلر فازی پیشنهاد شده در آزمایش خطای مشتق خطا را از ورودی دریافت می کرد. به نظر شما لزوم استفاده از مشتق خطای چیست؟ آیا می توان تنها با فیدبک گرفتن از خطای خودرو توصیف شده در قسمت شرح آزمایش را کنترل کرد؟

۳. کنترلر فازی طراحی شده در آزمایشگاه را به گونه ای تغییر دهید که تنها خطای شیب جاده را از ورودی دریافت کند. هدف کنترلر ثابت نگاه داشتن سرعت خودرو بر روی  $50 \text{ km/h}$  است

۴. ترکیب دو رابطه فازی زیر را براساس قاعده  $\max - \min$  به دست آورید:

$$R_2 = \begin{bmatrix} .9 & .1 \\ .2 & .3 \\ .5 & .6 \\ .7 & .2 \end{bmatrix} \quad (۲.۱۴)$$

$$R_1 = \begin{bmatrix} .1 & .3 & .5 & .7 \\ .4 & .2 & .8 & .9 \\ .6 & .8 & .3 & .2 \end{bmatrix}$$

۵. مجموعه های فازی فشار و حجم دارای توابع عضویت زیر می باشند ارزش درستی گزاره اگر فشار ۴ باشد پس حجم برابر با ۳ خواهد بود را پیدا کنید.

$$A(\mu) = \begin{cases} 1 & \mu > 5 \\ 1 - (\frac{5-\mu}{4})^2 & 1 < \mu < 5 \\ 0 & O.W \end{cases} \quad (۳.۱۴)$$

$$B(\mu) = \begin{cases} 1 & \mu < 1 \\ 1 - (\frac{v-1}{4})^2 & 1 < v < 5 \\ 0 & O.W \end{cases}$$

۶. مجموعه های فازی  $F, G, H$  که به وسیله توابع تعلق زیر در بازه  $0 \leq x \leq 10$  تعریف می شوند را در نظر بگیرید.

$$\mu_F(x) = \frac{x}{x+2}$$

$$\mu_G(x) = 2^{-x} \quad (۴.۱۴)$$

$$\mu_H(x) = \frac{1}{1 + 10(x-2)^2}$$

فرمولهای زیر را برای توابع فازی تعریف شده به دست آورید و رسم کنید:

$\bar{H}, \bar{G}, \bar{F} \bullet$

$H \cup G, F \cup H, F \cup G \bullet$

$H \cap G, F \cap H, F \cap G \bullet$

$F \cup G \cup H, F \cap G \cap H \bullet$

$\bar{G}, F \cap \bar{H} \bullet$

۷. پارامتر کنترلی که به صورت کمی می‌توان توصیف کرد را به دلخواه انتخاب کنید. ( سطح آب یک مخزن یا سرعت یک موتور، قیمت یا مساحت خانه یا ... )

حال سعی کنید حداقل ۳ پارامتر کیفی برای این کمیت ارائه دهید. (کم و زیاد و متوسط و ....)  
با کمک زبان برنامه نویسی پایتون عضویت تمام اعداد در بازه‌ای منطقی برای آن کمیت را به دست بیاورید  
حال پارامتر دیگری به عنوان ورودی سیستم اضافه کنید و برای خروجی نیز تابع عضویت تعریف کنید ( برای سادگی برای خروجی فقط دو حالت در نظر بگیرید )

بعد از طراحی قوانین بین حالات ورودی و خروجی با استفاده از استنتاج لوكاشویتز و غیرفازی ساز – *center* – *of gravity* مقدار خروجی را به ازای مقادیر مختلف ورودی به دست آورده و بررسی کنید.



# آزمایش چهاردهم(کنترلر $PID$ فازی)

## ۱.۱۵ پیش گزارش

فرض کنید قصد دارید کنترلر  $PID$  را توسط یک کنترلر فازی پیاده سازی کنید.

۱. چه سیگنال هایی را به عنوان ورودی کنترلر فازی در نظر می گیرید.
۲. قوانینی را که به صورت اگر-آنگاه برای پیاده سازی کنترلر فازی در نظر می گیرید را بیان کنید.

## ۲.۱۵ مقدمه

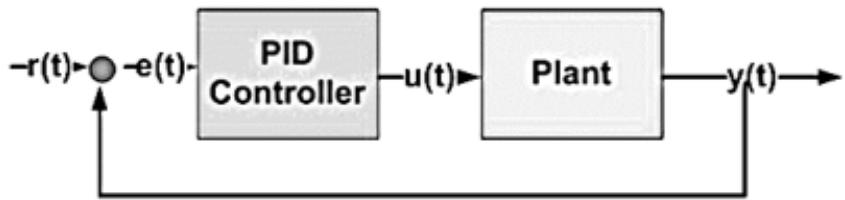
در این آزمایش قصد داریم عملکرد کنترلر  $PID$  را توسط یک کنترلر فازی شبیه سازی کنیم. در این قسمت ابتدا به مروری کوتاه بر ساختار کنترلر  $PID$  پرداخته می شود. سپس ساختار کنترلر فازی برای شبیه سازی عملکرد کنترلر  $PID$  بررسی می شود.

## ۱.۲.۱۵ کنترلر $PID$

کنترل کننده  $PID$  یک کنترل کننده قدرتمند و در عین حال ساده، برای سیستم های کنترلی می باشد که تابع تبدیل و خروجی آن به صورت معادلات ۱.۱۲ است. ساختار کلی کنترلر در شکل ۱.۱۲ نمایش داده شده است. ورودی این کنترلر

خطا و خروجی آن ورودی سامانه است.

$$\begin{aligned}
 G(s) &= K_p + \frac{K_i}{s} + K_d s \\
 u(t) &= K_p[e(t) + \frac{1}{T_i} \int_0^t e(r) dr + T_d \dot{e}] \\
 T_i &= \frac{K_p}{K_i} \\
 T_d &= \frac{K_d}{K_p}
 \end{aligned} \tag{1.15}$$



شکل ۱.۱۵: ساختار کلی کنترلر PID

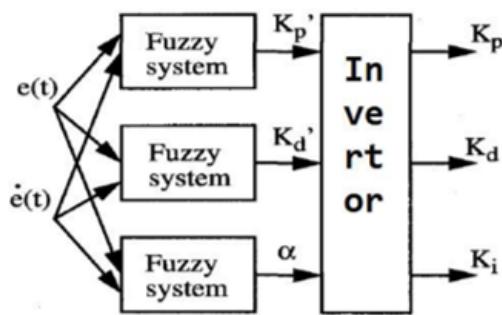
### کنترلر PID فازی

برای طراحی کنترلر PID فازی ورودی های کنترلر به صورت خطأ و مشتق خطأ در نظر گرفته می شود. خروجی کنترلر نیز ضرایب  $K_{P'}$  و  $K_{d'}$  و  $\alpha$  که به صوت زیر تعریف می شوند در نظر گرفته می شود. در نهایت بعد از به دست آوردن ضرایب توسط یک بلوک معکوس کننده ضرایب  $K_{P'}$  و  $K_{d'}$  و  $\alpha$  به ضرایب  $K_p$ ،  $K_d$  و  $K_i$  تبدیل می شوند. ساختار ورودی و خروجی کنترلر فازی PID در شکل ۲.۱۲ نمایش داده شده است.

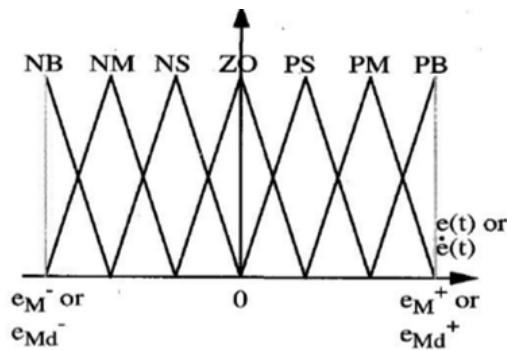
$$\begin{aligned}
 K_{P'} &= \frac{K_p - K_{pmin}}{K_{pmax} - K_{pmin}} \\
 K_{d'} &= \frac{K_d - K_{dmin}}{K_{dmax} - K_{dmin}}
 \end{aligned} \tag{2.15}$$

$$T_i = \alpha T_d$$

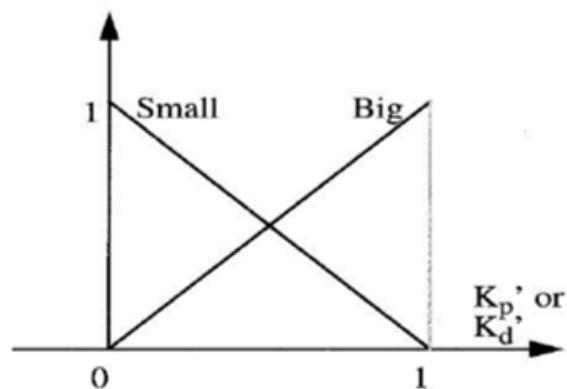
توابع عضویت مختلفی را می توان برای ورودی های کنترلر PID فازی در نظر گرفت. یک نمونه از توابع عضویت که می توان برای خطأ و مشتق خطأ در نظر گرفت در شکل ۳.۱۲ نمایش داده شده است. برای  $K_{d'}$ ،  $K_{P'}$  و  $\alpha$  نیز انتخاب های متفاوتی برای توابع عضویت وجود دارد. یکی نمونه از توابع عضویت برای  $K_{P'}$ ،  $K_{d'}$  در شکل و یک نمونه تابع عضویت برای  $\alpha$  در شکل نمایش داد شده است. بعد از مشخص شدن توابع عضویت برای متغیرهای مختلف سیستم فازی



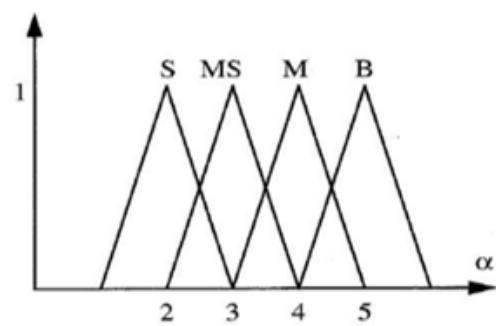
شکل ۱۵: رابطه بین ورودی و خروجی های کنترلر PID فازی



شکل ۱۶: تابع عضویت خطأ و مشتق خطأ



شکل ۱۷: تابع عضویت گین ها



شکل ۱۸: تابع عضویت  $\alpha$

برای کامل شدن ساختار باید به تعریف قوانین پرداخته شود. قوانین سیستم فازی را می توان به گونه های مختلفی تعریف کرد. یک نمونه از قوانین برای  $K_P'$ ،  $K_{d'}$  و  $\alpha$  به ترتیب در شکل های ۶.۱۲، ۷.۱۲ و ۸.۱۲ نمایش داده است.

		$\dot{e}(t)$						
		NB	NM	NS	ZO	PS	PM	PB
e(t)	NB	B	B	B	B	B	B	B
	NM	S	B	B	B	B	B	S
	NS	S	S	B	B	B	S	S
	ZO	S	S	S	B	S	S	S
	PS	S	S	B	B	B	S	S
	PM	S	B	B	B	B	B	S
	PB	B	B	B	B	B	B	B

شکل ۶.۱۵: قوانین  $K_{P'}$

		$\dot{e}(t)$						
		NB	NM	NS	ZO	PS	PM	PB
e(t)	NB	S	S	S	S	S	S	S
	NM	B	B	S	S	S	B	B
	NS	B	B	B	S	B	B	B
	ZO	B	B	B	B	B	B	B
	PS	B	B	B	S	B	B	B
	PM	B	B	S	S	S	B	B
	PB	S	S	S	S	S	S	S

شکل ۷.۱۵: قوانین  $K_{d'}$

با تعریف شدن قوانین و مشخص شدن توابع عضویت با انتخاب یک فازی ساز و یک ضد فازی ساز<sup>۱</sup> طراحی کنترلر فازی

		$\dot{e}(t)$						
		NB	NM	NS	ZO	PS	PM	PB
e(t)	NB	2	2	2	2	2	2	2
	NM	3	3	2	2	2	3	3
	NS	4	3	3	2	3	3	4
	ZO	5	4	3	3	3	4	5
	PS	4	3	3	2	3	3	4
	PM	3	3	2	2	2	3	3
	PB	2	2	2	2	2	2	2

شکل ۸.۱۵: قوانین  $a$

پایان می پذیرد. برای انتخاب فازی ساز و ضد فازی ساز نیز گزینه های مختلفی وجود دارد برای مثال می توان از  $PID$  فازی ساز<sup>۲</sup> و ضد فازی ساز  $singleton$   $average - center$  (با در نظر گرفتن  $product - inference - engine$ ) استفاده

<sup>1</sup> defuzzifier

## ۳.۱۵ شرح آزمایش

هدف از این آزمایش، طراحی و پیاده‌سازی یک سیستم فازی  $PID$  برای هدایت ربات از موقعیت اولیه به موقعیت دلخواه است. ورودی‌های سیستم فازی، مکان هدف نسبت به مکان کنونی ربات و خروجی آن فرمان کنترلی موتورها می‌باشد. با انتخاب قوانین مناسب، کنترل صورت خواهد گرفت.

برای پیاده‌سازی کنترل فازی مراحل زیر را دنبال کنید:

۱. با توجه به شکل‌های ۳.۱۲ تا ۵.۱۲ (با استفاده از جعبه ابزار فازی متلب) توابع عضویت متغیرهای ورودی و خروجی را مشخص کنید(خطا و مشتق خطای عنوان ورودی سیستم  $K_{P'}$  ،  $K_{d'}$  و  $\alpha$  به عنوان خروجی سیستم در نظر گرفته می‌شود)
۲. با استفاده از قسمت انتخاب قوانین در جعبه ابزار متلب قوانین سیستم را مطابق با شکل‌های ۶.۱۲ ، ۷.۱۲ و ۸.۱۲ تعریف کنید.
۳. فازی ساز را به صورت *center – average singleton* در نظر بگیرید.
۴. با توجه به معادلات ۲.۱۲ و با در نظر گرفتن  $K_p$  در محدوده  $K_i$  در محدوده یک بلوک در سیمولینک طراحی کنید که  $K_{P'}$  ،  $K_{d'}$  و  $\alpha$  را از ورودی دریافت کند و  $K_p$  و  $K_i$  و  $T_i$  را به عنوان خروجی باز گرداند.
۵. خروجی‌های سیستم فازی را به عنوان ورودی به بلوک طراحی شده در مرحله ۳ دهید.

بعد از طراحی کردن کنترلر  $PID$  فازی کنترلر را بر روی سامانه توصیف شده در معادلات ۳.۱۲ و با در نظر گرفتن  $\sin(t)$  به عنوان ورودی مرجع امتحان کنید و نمودار خطای خروجی (تفاوت بین خروجی سامانه و ورودی مرجع) را رسم کنید.

$$H(s) = \frac{40}{s + 1} \quad (3.15)$$

## ۴.۱۵ تمرین

۱. با فرض وجود نویز در خروجی سامانه عملکرد کنترلر فازی طراحی شده را شبیه سازی و بررسی کنید.

---

۲. تعداد قوانین کنترل فاری را افزایش دهید و تاثیر آن را در عملکرد کنترلر بررسی کنید.

## آزمایش پانزدهم ( $c - mean$ )

### ۱.۱۶ پیش گزارش

۱. ضعف های الگوریتم  $K$  میانگین را بیان کنید. این ضعف ها چگونه توسط الگوریتم  $c - mean$  فازی بهبود می یابد. و  $k - mean$  را بیان کنید.

### ۲.۱۶ مقدمه

در آزمایش های قبل با الگوریتم  $k - mean$  برای دسته بندی داده ها آشنا شدید. ممکن است مرز مشخصی برای دسته های داده ها وجود نداشته باشد. در چنین شرایطی کارآیی الگوریتم  $k - mean$  کاهش می یابد. الگوریتم  $c - mean$  فازی مشابه الگوریتم  $k - mean$  است با این تفاوت که خروجی این الگوریتم احتمال متعلق بودن داده ها به دسته های مشخص شده است.

تابع هزینه در نظر گرفته شده برای الگوریتم  $c - mean$  به صورت معدله ۱.۱۳ است. در معادله زیر  $n$  مشخص کننده تعداد داده ها و  $i$  مشخص کننده تعداد دسته ها و  $V_i$  مشخص کننده مرکز دسته  $i$  است. هدف الگوریتم  $c - mean$  کمینه کردن تابع هزینه در نظر گرفته شده در معادله ۱.۱۳ است. این تابع هزینه تلاش می کند تراکم داده ها درون یک دسته را افزایش دهد با زیاد شدن تراکم داده ها درون یک دسته هم زمان تراکم داده ها در خارج از دسته ها کاهش می یابد.

$$\sum_{k=1}^n \sum_{c=1}^i \mu_{ik} \| x_{ik} - v_i \| \quad (1.16)$$

در شکل ۱.۱۳ مراحل الگوریتم  $c - mean$  نمایش داده شده است. در این آزمایش داده ها قصد داریم به پیاده سازی الگوریتم

- Suppose there are  $n$  data points,  $X = \{x_1, \dots, x_n\}$ . Fix  $c$ ,  $2 \leq c < n$ , and initialize  $U^{(0)} \in M_c$ .
- At iteration  $l$ ,  $l = 0, 1, 2, \dots$  compute the  $c$ -mean vectors  

$$v_i^{(l)} = \frac{\sum_{k=1}^n \mu_{ik}^{(l)} x_k}{\sum_{k=1}^n \mu_{ik}^{(l)}}$$
where  $[\mu_{ik}^{(l)}] = U^{(l)}$ , and  $i = 1, 2, \dots, c$ .
- Update  $U^{(l)}$  to  $U^{(l+1)}$  using  

$$\mu_{ik}^{(l+1)} = \begin{cases} 1 & \|x_k - v_i^{(l)}\| = \min_{1 \leq j \leq c} (\|x_k - v_j^{(l)}\|) \\ 0 & \text{otherwise} \end{cases}$$
- Compare  $U^{(l)}$  with  $U^{(l+1)}$ : if  $\|U^{(l+1)} - U^{(l)}\| < \epsilon$  for a small constant  $\epsilon$  stop; otherwise, set  $l = l + 1$  and go to Step 2.

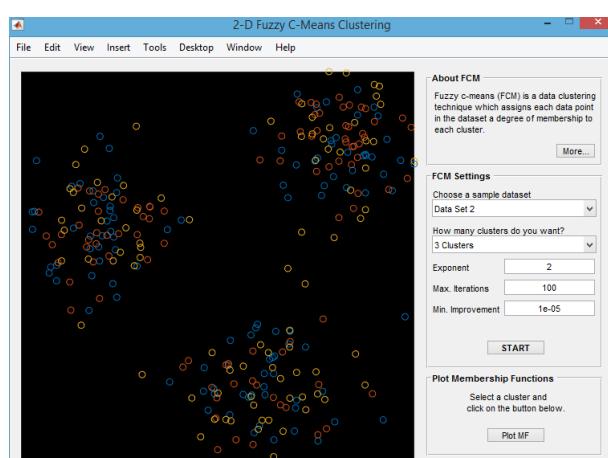
شکل ۱.۱۶: مراحل الگوریتم  $c - mean$

فازی با استفاده از ابزار متلب بپردازیم.

## ۱.۲.۱۶ آشنایی با جعبه ابزار $c - mean$ فازی متلب

- برای این نوع طبقه بندی، ابزار مخصوصی در متلب تعبیه شده که برای مرکز دسته‌ها یک فرض اولیه در نظر می‌گیرد. مرکز دسته‌ها یعنی موقعیت میانگین هر دسته‌بندی. از آنجایی که حدس اولیه عموماً اشتباه است، ابزار متلب به هر نقطه،تابع عضویت نسبت می‌دهد که قادر است مراکز دسته‌هارا به محل اصلی آنها هدایت کند. این کار با کمینه کردن تابع هدف رخ می‌دهد که مشخص کننده فاصله بین هر نقطه با مرکز دسته‌ای است که توسط تابع عضویت آن نقطه، وزن پیدا کرده است.

با تایپ کردن `fcmdemo` در `command – window` پنچره ای به شکل ۲.۱۳ باز می‌شود. پنچره باز شده محیط `fcm – setting` (مربع سمت راست بالای پنچره باز شده) می‌توانید داده `fcmdemo` را نشان می‌دهد. در قسمت `FCM Settings` (با کلیک بر روی *Start*) ورودی و تعداد دسته‌های مورد نظر را مشخص کرد. بعد از مشخص کردن تنظیمات با کلیک بر روی *Start* الگوریتم اجرا می‌شود.



شکل ۲.۱۶: محیط  $fcm$

## ۲.۲.۱۶ ابزار فازی در محیط *command – line* متلب

برای استفاده از ابزار فازی می‌توان از *command – line* متلب نیز استفاده کرد. در *command – line* با استفاده از دستور  $[C, U] = fcm(dataset, N, options)$  می‌توان الگوریتم دسته بندی فازی را اجرا کرد. ورودی‌های این دستور داده‌های در نظر گرفته شده برای دسته بندی و یک بردار *options* به صورت *options = [exponentmaxIterations, minImprovement, displayObjectiveFunction];* داده های در نظر گرفته شده برای دسته بندی و یک بردار *options* به صورت *options = [exponentmaxIterations, minImprovement, displayObjectiveFunction];* دسته بندی بیشینه مجاز تعداد تکرار الگوریتم است. المان دوم در بردار *options* مشخص کننده کمترین تغییرات مراکز دسته‌ها در هر تکرار نسبت به تکرار قبل (برای مشخص شدن همگرا شدن و یا نشدن الگوریتم) است. در صورت *True* بودن المان سوم در بردار *options* مقدار تابع هزینه بعد از هر تکرار نمایش داده می‌شود و در غیر اینصورت این مقدار نمایش داده نمی‌شود. خروجی‌های دستور *fcm* مراکز داده‌ها و ماتریس احتمال متعلق بودن هر داده به هر دسته است.

## ۳.۱۶ شرح آزمایش

در این آزمایش ابتدا قصد داریم با استفاده از جعبه ابزار دسته بندی فازی متلب به دسته بندی داده‌ها بپردازیم سپس با استفاده از *command – line* به دسته بندی داده‌های چند بعدی پرداخته می‌شود.

### ۱.۳.۱۶ دسته بندی با استفاده از جعبه ابزار *fcm*

در این قسمت قصد داریم به دسته بندی داده‌های ۲ بعدی با استفاده از جعبه ابزار *fcm* بپردازیم. مراحل زیر را دنبال کنید:

۱. در *Windows Command – Windows* دستور *fcmdemo* را تایپ کنید.

۲. نمونه مجموعه داده‌ی شماره ۵ را از منوی *Choose a sample dataset* انتخاب کنید.

۳. در منوی بعدی، تعداد دسته‌ها را برابر با ۵ دسته قرار دهید.

۴. *START* را انتخاب کنید و نتیجه را مشاهده نمایید.

همانطور که مشاهده می‌شود، حدس مرکز اولیه مرحله به مرحله توسط نرم افزار مشخص شده و به مرکز بهینه نزدیک تر می‌گردد.

۵. یکی از دسته‌ها را انتخاب کنید. با کلیک بر روی *PLOTFM* سطح تابع عضویت آن را مشاهده کنید.

### ۲.۳.۱۶ دسته بندی *fcm* در *Command – line*

فرض کنید یک شرکت تبلیغاتی شبکه اجتماعی کاریابی، برای گسترش عملکرد شبکه‌ی خود تصمیم دارد کاربران خود را با داشتن اطلاعات زیر به ۷ دسته تقسیم کند. بدین صورت که با توجه به اطلاعات تقسیم بندی شده، تبلیغات منحصر به فردی برای گروه‌های متفاوت ارسال شود. برای مثال، گروه شامل افراد جوان تر، تبلیغات مربوط به فناوری‌های روز شبکه‌را دریافت می‌کنند. درحالی که افراد مسن، تبلیغاتی درباره ساده سازی شبکه و کاربری کم دردسرتر مشاهده خواهند کرد.

داده شرکت در اختیارتان قرار داده می‌شود. قصد داریم با استفاده از *fcm* داده‌های شرکت را به ۷ دسته مناسب تقسیم کنیم.

برای دسته بندی داده‌ها مراحل زیر را دنبال کنید:

۱. در متلب به آدرس *CILab* مراجعه کنید. مجموعه داده‌ی *SocialNetworkAds.csv* را انتخاب کنید تا اطلاعات را وارد متلب کنید.

۲. داده‌های مورد نیاز را انتخاب کرده و *Numeric – Matrix* را برابر با *Output – Type* قرار دهید و بر روی *ImportSelection* کلیک کنید تا اطلاعات در یک متغیر در محیط متلب ذخیره شوند.

۳. بر روی متغیر ایجاد شده در *Workspace* راست کلیک کرده و گزینه *Save – As* را انتخاب کنید. حال مجموعه داده را با نام *SNdataset.m* ذخیره کنید.

۴. فایل ذخیره شده را باز کنید و در انتهای آن دستور زیر را اضافه کنید.

$$\text{numberofClusters} = 7;$$

$$[\text{center}, U] = fcm(\text{SocialNetworkAds}, \text{numberofClusters});$$

با استفاده از این دستور، مشاهده می‌شود که تمام مراحل تقسیم بندی اجرا شده است. در نهایت متغیرهای *U*، *center* و *Workplace* ذخیره می‌شوند.

## ۴.۱۶ تمرین

۱. مراحل دسته بندی انجام شده در بخش آشنایی با جعبه ابزار  $c - mean$  فازی متلب را برای مجموعه داده سوم به جای مجموعه داده پنجم(از مجموعه داده آماده متلب در بخش انتخاب مجموعه داده) تکرار کنید.
۲. مراحل دسته بندی انجام شده در بخش آشنایی با جعبه ابزار  $c - mean$  فازی متلب را باز دیگر برای مجموعه داده پنجم(مجموعه داده آزمایش شده در آزمایشگاه ) تکرار کنید آیا خروجی به دست آماه با خروجی به دست آماده در آزمایشگاه در قسمت دسته بندی توسط جعبه ابزار  $fcm$  یکسان است ؟ پاسخ خود را توجیه کنید.



# آزمایش شانزدهم (*anfis*)

## ۱.۱۷ پیش گزارش

۱. یک شبکه *anfis* چه مزایایی نسبت به شبکه عصبی و فازی دارد؟

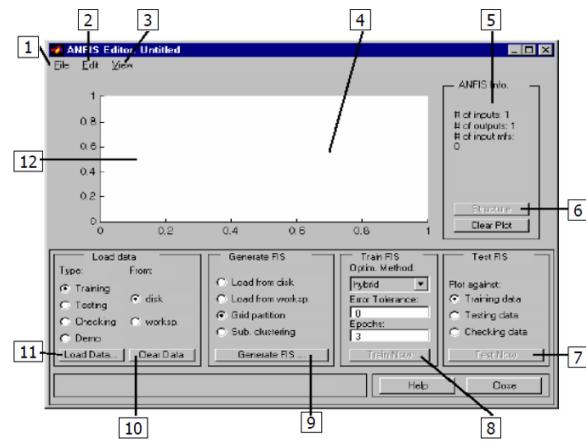
۲. کاربردهای شبکه *anfis* را بیان کنید

## ۲.۱۷ مقدمه

سیستم عصبی فازی *ANFIS(Adaptive – Neural – Fuzzy – Inference – System)* در واقع ترکیب یک سیستم فازی با یک شبکه عصبی است. در شبکه فازی تعین توابع عضویت، ویژگی آن ها و ... توسط دانش انسانی صورت می گرفت. در سیستم عصبی-فازی از یک شبکه عصبی برای انتخاب پارامترهای سیستم فازی استفاده می گردد. در این آزمایش قصد داریم با استفاده از جعبه ابزار *anfis* متلب به پیاده سازی یک سیستم *anfis* بپردازیم. در ابتدا به چگونگی استفاده از جعبه ابزار *anfis* متلب پرداخته می شود.

## ۱.۲.۱۷ جعبه ابزار *anfis* متلب

با تایپ کردن *anfisedit* در *command – line* نمایش داده می شود. در مربع شماره ۱ می توانید سیستم *anfis* ایجاد شده را ذخیره کنید و یا یک سیستم *anfis* جدید ایجاد کنید. در مربع شماره ۲ با کلیک بر روی *Edit* می توانید پارامترهای سیستم فازی را مشخص کنید. در صورت در اختیار داشتن یک فایل *.fis* می توانید در قسمت مشخص شده توسط مربع ۳ آن را بارگزاری کنید. مربع شماره ۴ محل رسم را نمایش می دهد. در مربع شماره ۵ مشخصات سیستم *anfis* مانند تعداد ورودی ها و خرجی ها نمایش داده می شود. بعد از ایجاد یا باز کردن یک فایل *FIS* توسط گزینه نمایش داده شده با عدد ۶ می توان ساختار شبکه فازی را مشاهده نمود با کلیک



شکل ۱.۱۷: جعبه ابزار *anfistool*

کردن بر روی مربع ۷ می توانید داده های تست را رسم کنید. بعد از مشخص کردن مشخصات سیستم فازی با انتخاب کردن مربع ۸ می توان شبکه را آموزش داد. در مربع شماره ۹ می توانید یک فایل *Fis* را به روش های مختلف تولید کنید. در مربع شماره ۱۰ می توانید داده های بارگزاری شده را پاک کنید. در مربع شماره ۱۱ می توانید داده های ورودی و خروجی را بارگزاری کنید. برای رسم نمودار در ناحیه مشخص شده توسط عدد ۱۲ داده های تست با  $00$ ، "داده های آموزش با +نمایش داده می شود

### ۳.۱۷ شرح آزمایش

در این آزمایش قصد داریم توسط یک شبکه *anfis* به تخمین تابع  $\cos(t)$  بپردازیم.  
برای تخمین تابع  $\cos(t)$  توسط شبکه *anfis* مراحل زیر را دنبال کنید.

۱. داده های ورودی و خروجی تابع  $\cos(t)$  را در یک بازه تولید کنید. داده های ورودی و خروجی را در *workspace* ذخیره کنید.

۲. در *command-line* *anfisedit* را در مطلب تایپ کنید تا جعبه ابزار *anfis* نمایش داده شود.

۳. در قسمت *loaddata* در جعبه ابزار *anfis* داده های ذخیره شده در *workspace* را در محیط *anfis* وارد کنید.

۴. گزینه *Generate - Fis* را انتخاب کنید و مشخصات سیستم فازی و تعداد توابع عضویت را وارد کنید.

۵. با انتخاب گزینه *Train - Fis* عملیات یادگیری شبکه *Fis* را شروع کنید و نمودار خطای مشاهده کنید.

## ۴.۱۷ تمرین

۱. سعی کنید تابع تبدیل  $\frac{1}{s+2}$

را توسط یک شبکه *anfis* تخمین بزنید.

۲. ابتدا در سیمولینک سیستمی با تابع تبدیل زیر را پیاده سازی کنید.

$$G(z^{-1}) = \frac{(1 - 0.8z^{-1})}{(1 - 0.1z^{-1})(1 - 0.5z^{-1})(1 - 0.94z^{-1})} \quad (1.17)$$

ابتدا چند ورودی مختلف به این سیستم اعمال کنید و داده های مربوطه را ذخیره کنید.

به کمک ابزار *anfis* در متلب به مدلسازی (شناسایی) این سیستم بپردازید.

حال سیستم جدید خود را به ازای ورودی های جدید تست کنید و نتایج را با سیستم اصلی مقایسه کنید.



# پروژه بخش فازی

## ۱.۱۸ پیش گزارش

۱. با مراجعه به اینترنت معادلات دینامیک مربوط به خودرو بدون سرنشین چهار چرخ را بیان کنید.
۲. با مراجعه به اینترنت معادلات دینامیک مربوط به ربات نقاش ۲ بعدی را بیان کنید.

## ۲.۱۸ مقدمه

در آزمایش های گذشته به طراحی کنترلر توسط سیستم فازی پرداخته شد. در این آزمایش قصد داریم به پیاده سازی سخت افزاری الگوریتم های بیان شده بپردازیم. برای پیاده سازی سخت افزاری از خودرو زمینی چهار چرخ بدون سرنشین<sup>۱</sup> و یا ربات نقاش ۲ بعدی استفاده می شود. در ادامه به معرفی کوتاه ربات نقاش ۲ بعدی و خودرو زمینی بدون سرنشین پرداخته می شود.

## ۱.۲.۱۸ خودرو زمینی بدون سرنشین

خودرو زمینی بدون سرنشین یک ربات ۴ چرخ با مکانیزم حرکت تانکی و خودکار بوده که قابلیت طی مسیر به صورت خودکار و ارتباط با عامل های دیگر از طریق بی سیم را نیز دارد. این ربات مجهز به حلقه ۶ تایی سونار برای مانع یابی و همچنین ۲ عدد انکوادر جهت مکان یابی بوده و قابلیت نصب سنسورهای مختلفی در آن در نظر گرفته شده است. توضیحات مفصل این ربات در (پیوست ۱) انجام شده است. (حتما پیش از آزمایش، پیوست مطالعه شود). تصویر این ربات در شکل زیر نمایش داده شده است.

---

<sup>۱</sup>UGV(unmanned – ground – veichle)



شکل ۱.۱۸: خودروی زمینی بدون سرنشین

## ۲.۲.۱۸ ربات نقاش ۲ بعدی

ربات نقاش یک ربات دو بعدی با دو درجه آزادی ، همانطور که در شکل زیر مشاهده می شود، این ربات از چهار لینک (بازو) که از یک طرف به دو محرک (موتور)  $DC$  و از طرف دیگر به یکدیگر متصل شده اند تشکیل شده است. نقطه اتصال لینک ها به هم سر موثر نماید که با اعمال فرمان مناسب به موتورها ، سر موثر به نقاط مطلوب در صفحه  $y - x$  حرکت داده می شود. سر موثر شامل یک قلم میباشد که این قلم جهت بر جای گذاشتن مسیر حرکت سر موثر به کار رفته است. سر موثر ربات به یک



شکل ۲.۱۸: ربات نقاش ۲ بعدی

## ۳.۱۸ شرح آزمایش

در این بخش ۲ آزمایش برای پیاده سازی بر روی ربات نقاش ۲ بعدی و خودرو بدون سرنشین در نظر گرفته شده است . از بین این ۲ آزمایش ۱ آزمایش را انتخاب کرده و به پیاده سازی سخت افزاری آن بپردازید.

## ۱.۳.۱۸ آزمایش ۱ - کنترل خودروی زمینی بدون سرنوشت

در این قسمت به پیاده سازی سخت افزاری کنترل  $PID$  فازی شبیه سازی شده، بر روی خودرو زمینی بدون سرنوشت می شود.

قصد داریم خودرو بدون سرنوشت را به گونه ای کنترل کنیم که بر روی خط  $x = 10$  (یا  $y = 10$ ) حرکت کند. برای کنترل خودروی بدون سرنوشت مراحل زیر را دنبال کنید:

۱. ابتدا از صحت کنترلر پیاده سازی شده در آزمایش ۱۱ اطمینان حاصل کنید.

مدل ساده خودرو بدون سرنوشت به صورت رابطه ۱.۱۵ است. ابتدا کنترلر طراحی شده در آزمایش ۱۱ را برروی این مدل شبیه سازی کنید. ورودی مرجع را برابر با یک عدد ثابت در نظر بگیرید. در صورت عملکرد صحیح کنترلر خطای بین خروجی سامانه و ورودی مرجع به سمت صفر میل می کند.

$$H(s) = \frac{40}{0.02s^2 + s} \quad (1.18)$$

۲. فایل سیمولینک مربوط به کنترل خودرو بدون سرنوشت در اختیارتان قرار داده می شود. فایل را در محیط سیمولینک باز کنید.

۳. ورودی مرجع کنترل را برابر با ۱۰ در نظر بگیرید. مقدار  $pwm_1(pwm_2)$  را برابر با خروجی کنترلر فازی قرار دهید. موقعیت خودرو بدون سرنوشت را نیز به عنوان ورودی کنترل در نظر بگیرید. خطای بین ورودی مرجع و موقعیت خودرو بدون سرنوشت را رسم کنید.

## ۲.۳.۱۸ آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی

در این قسمت قصد داریم با استفاده از یک شبکه *anfis* به شناسایی ربات نقاش ۲ بعدی بپردازیم. برای شناسایی ربات نقاش ۲ بعدی مراحل زیر را دنبال کنید:

۱. ابتدا فایل ربات را که در اختیارتان قرار گرفته است را اجرا کنید. با اجرا کردن این فایل ربات شروع به نقاشی یک دایره خواهد کرد.

۲. این ربات ۲ سیگنال را از ورودی دریافت می کند (سیگنال های  $u_1, u_2$  در فایل سیمولینک). این ۲ سیگنال را به عنوان ورودی در *workspace* مطلب ذخیره کنید و زاویه بازو ها را به عنوان خروجی در *workspace* ذخیره کنید.

- 
۳. داده های ورودی و خروجی را به شبکه *anfis* طراحی شده در آزمایش ۱۳ دهید و شبکه را شناسایی کنید.
۴. خروجی شبکه *anfis* را به ازای  $u_1, u_2$  برابر با ۴ تخمین بزنید.
۵. در فایل سیمولینک مقادیر  $u_1, u_2$  را برابر با ۴ قرار دهید و برنامه را بر روی ربات اجرا کنید.
۶. مقدار زاویه تخمین زده شده توسط شبکه *anfis* را با مقدار زاویه بازوهای ربات نقاش ۲ بعدی مقایسه کنید.
۱. خروجی های حاصل شده از بخش عملی را بخش تئوری مقایسه کنید. ضعف های الگوریتم های استفاده شده در پیاده سازی سخت افزاری چیست؟

# پیوست

## آ ربات نقاش ۲ بعدی

ربات نقاش دو بعدی در واقع یک ربات با دو بازو است که بازوها از یک طرف به یکدیگر متصل شده اند و نقش گیرنده قلم یا هر وسیله ای که قرار است با آن نقاشی رسم شود را دارند. هر یک از بازوهای این ربات دارای دو درجه آزادی است که امکان حرکت آزادانه را به سادگی به آن می دهد.

برای حرکت کامل در صفحه ۲ بعدی وجود ۳ درجه آزادی کافیست. یعنی تنها کافیست با دو درجه آزادی در مختصات مطلوب قرار گرفت و با درجه آزادی سوم با زاویه (Orientation) دلخواه در آن نقطه قرار گرفت. وجود ۴ درجه آزادی در این ربات باعث می شود با تعداد حالت های بیشتری برای مفاصل (بازوها) بتوان در موقعیت مطلوب قرار گرفت. وجود درجه آزادی چهارم در این ربات اصطلاحاً باعث ایجاد Redundancy در آن می شود. وجود درجات آزادی بالاتر از حد نیاز در ربات ها بخصوص برای مقابله با موانع کاربرد دارد.

با توضیحات داده شده به نظر شما برای حرکت کامل در فضای ۳ بعدی حداقل چند درجه آزادی لازم است؟ اعمال نیرو به بازوهای ربات نقاش از طریق دو عدد سرو موتور در مفصل دو بازو انجام می شود. همچنین برای فیدبک گرفتن از محل سر بازوها که به هم متصل است از انکودر بر روی موتورها استفاده می شود. به عبارتی دیگر با دانستن محل موتورها می توان محل سر بازوها را تنظیم نمود.

جهت کار با ربات باید با داشتن مختصات مطلوب قلم، موقعیت لازم هر یک از بازو ها و سپس موتور ها را بدست آوریم. این مسئله در رباتیک، مسئله سینماتیک معکوس نامیده می شود. برای کشیدن هر طرحی با ربات لازم است از معادلات سینماتیک معکوس ربات استفاده کرده و موقعیت لازم برای هر یک از بازو ها و موتور ها را بدست آورد و موتور ها و بازو ها را به آن موقعیت هدایت کرد تا نقطه انتهایی ربات (End Effector) که در این ربات همان قلم است، در موقعیت مطلوب قرار بگیرد.

مسئله سینماتیک مستقیم در ربات ها، عکس مطلب فوق می باشد. یعنی در سینماتیک مستقیم ربات، با فرض

داشتن موقعیت موتور ها و بازو ها، هدف بدبست آوردن نقطه انتهایی ربات (قلم) می باشد.

معمولًا مسئله سینماتیک معکوس در ربات ها، مسئله پیچیده تری است تا مسئله سینماتیک مستقیم؛ چرا که در سینماتیک معکوس معمولاً باید به حل دستگاهی از معادلات غیر خطی (و غالباً شامل عبارت های مثلثاتی) بپردازیم. این گونه معادلات اکثراً پاسخی به فرم بسته ندارند و برای حل آن ها باید به روش های عددی پرداخت.

اکنون معادلات مربوط به مسئله سینماتیک مستقیم و سینماتیک معکوس ربات نقاش دو بعدی را بررسی می کنیم. کدام معادلات مربوط به سینماتیک مستقیم هستند؟ کدام مربوط به سینماتیک معکوس؟ چرا؟

## ۱.۱ ترسیم با ربات های نقاش

هدف نهایی ربات های نقاش، ترسیم طرح های دلخواه می باشد. یکی از راه های ترسیم طرح با این ربات ها داشتن معادلات طرح مورد نظر می باشد. با داشتن این معادلات می توان به کمک معادلات سینماتیک معکوس ربات، طرح مورد نظر را رسم کرد. بطور مثال برای ترسیم یک دایره کافیست هر یک از بازو ها یک حرکت سینوسی با دامنه و فرکانس یکسان و اختلاف فاز  $90^\circ$  درجه نسبت به هم داشته باشند. اما واضح است که برای اکثر تصاویر موجود (تصاویر گرفته شده با دوربین و ...) بدست آوردن این گونه معادلات بسیار پیچیده و عملاً غیر ممکن است. در این گونه موارد می توان با استفاده از الگوریتم های مرسوم پردازش تصویر، لبه ها و کانتور های تصاویر را یافت و باز هم به کمک معادلات سینماتیک معکوس، به ترسیم این نقاط پرداخت. باید این نکته را هم در نظر گرفت که اکثر ربات های نقاش موجود تنها از یک رنگ استفاده می کنند. به علاوه این که شدت این رنگ نیز معمولاً قابل تنظیم نیست. در نتیجه تصویر رسم شده با این ربات ها معمولاً تنها از یک رنگ بجز پس زمینه تشکیل شده است. در نتیجه این گونه ربات ها برای ترسیم تصاویری با جزئیات رنگی بالا مناسب نمی باشند. برای وضوح بیشتر بهتر است تصویر ورودی به گونه ای پردازش شود تا ضمن حفظ کیفیت و جزئیات تنها از دو رنگ (ممولاً سفید و غیرسفید) تشکیل شده باشد. بدین منظور می توان از الگوریتم هایی نظیر Adaptive Thresholding و OpenCV از توابع همچون cv2.findContours ، cv2.adaptiveThreshold ، cv2.threshold و ... در مازول cv2.CV\_MORPH\_CLOSE استفاده کنند.

در شکل ?? نمونه ای از تصویر ورودی، و خروجی ترسیم شده توسط ربات های نقاش را می بینید.



(ب) تصویر ترسیم شده



(آ) تصویر ورودی

شکل ۱.۱۹: نمونه ای از ورودی و خروجی های ربات نقاش

## ب روبات زمینی ( $UGV$ )

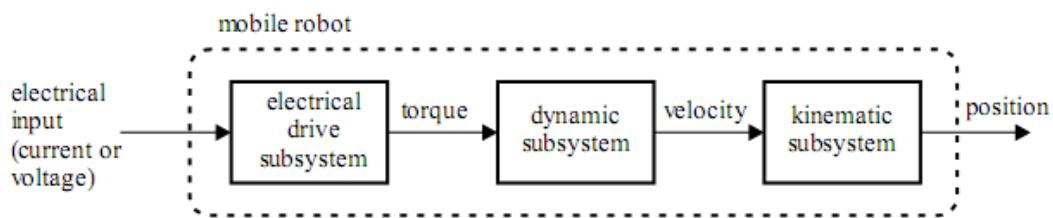
بات زمینی طراحی شده برای این آزمایشگاهیک ربات ۴ چرخ با مکانیزم حرکت تانکی و خودکار بوده که قابلیت طی مسیر به صورت خودکار و ارتباط با عاملهای دیگر از طریق بی‌سیم را نیز دارد. این ربات مجهز به حلقه ۶ تایی سونار برای مانع‌یابی و همچنین ۲ عدد انکودر جهت مکان‌یابی بوده و قابلیت نصب سنسورهای مختلفی در آن در نظر گرفته شده است. هرچند ابزارهای فازی و عصبی، امکان مدلسازی و طراحی کنترل کننده را به گونه‌ای فراهم می‌آورند که دیگر نیازی به دانستن مدل ریاضی ربات وجود ندارد، با این حال، برای اطمینان از مدلسازی و کنترل خودکار این ربات، در ابتدا مدل دینامیک آن تشریح می‌شود. در ادامه برای موقع اضطراری، آموزش نحوه کنترل روبات به صورت دستی و بدون برنامه نویسی توضیح داده شده. لطفاً برای حفظ ایمنی خود، روبات و دیگران، در صورت مشاهده هرگونه رفتار پیش‌بینی نشده یا خطرناک در برنامه‌ی حرکت خودکار، به سرعت روبات را خاموش کرده و با راهنمایی استاد خود و مطالب بخش دوم این پیوست، روبات را به صورت دستی به موقعیت امن منتقل نمایید.

### ب.1 مدلسازی

برای کنترل هر ربات بر روی آن یک کامپیوتر محلی تعییه شده است که نرم‌افزارهای مسیریابی و پردازش تصویر و همچنین کنترل رفتاری ربات را بر عهده دارد. برای مدلسازی، ربات را به سه بخش تقسیم می‌کنیم که شامل سینماتیک، دینامیک و بخش درایور آن است



شکل ۲.۱۹: ربات زمینی



شکل ۳.۱۹: سه بخش یک ربات زمینی شامل سینماتیک، دینامیک و درایور

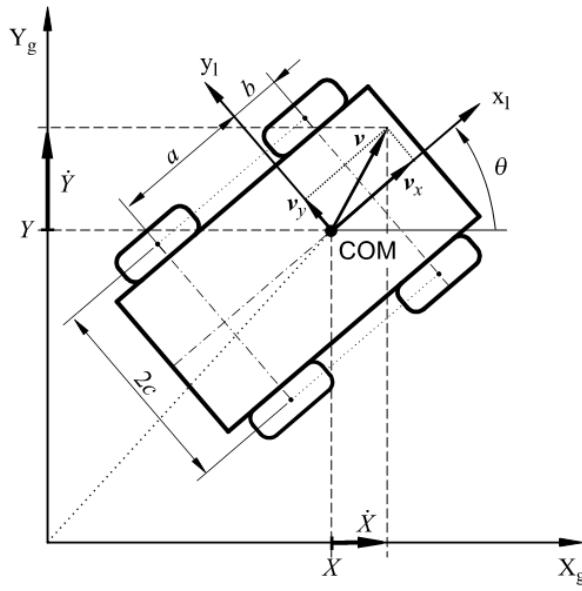
### مدل سینماتیکی

برای درنظر گرفتن مدل سینماتیک ربات، یک دستگاه مختصات مرجع در نظر می‌گیریم و آن را با سه‌تایی  $(X_g, Y_g, Z_g)$  نمایش می‌دهیم. یک دستگاه مختصات محلی هم که مرکز آن روی مرکز جرم ربات هست، را با  $(x_l, y_l, z_l)$  نمایش می‌دهیم. با فرض آنکه ربات روی صفحه با سرعت خطی  $(v = [v_x, v_y, 0])$  بیان شده در دستگاه محلی، حرکت می‌کند و اینکه بردار  $(q = [X, Y, \theta])$  مختصات ربات را در دستگاه مرجع بیان می‌کند، می‌توانیم بردار  $\dot{q} = [\dot{X}, \dot{Y}, \dot{\theta}]^T$  را بردار سرعت‌های تعمیم یافته بدانیم. با توجه به شکل زیر می‌توان، رابطه بین  $(\dot{x}, \dot{y})$  را با  $(v_x, v_y)$  به صورت زیر بیان نماییم.

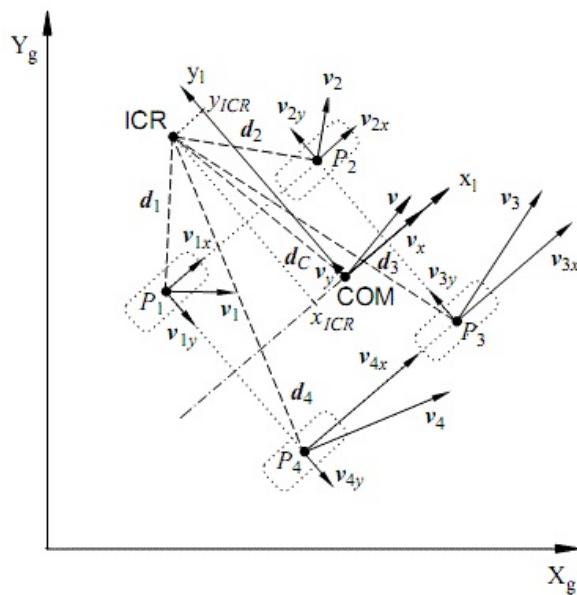
$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

این ربات، برخلاف ربات‌هایی با محور دیفرانسیلی، می‌توانند سرعتی در راستای عمود بر صفحه چرخ داشته باشند، و این تنها به دلیل لغزشی است که دارند، بنابراین فرض عدم لغزش برای این ربات‌ها نامعقول می‌باشد. اگر فاصله مرکز چرخش لحظه‌ای را با  $d_C = [d_{Cx}, d_{Cy}]^T$  نمایش دهیم و فاصله چرخ  $(i)$  ام را از مرکز چرخش لحظه‌ای را با  $d_i = [d_{ix}, d_{iy}]^T$  نمایش دهیم، در این صورت می‌توان روابط زیر را بنویسیم:

$$\frac{\|v_i\|}{\|d_i\|} = \frac{\|v\|}{\|d_C\|} = |w|$$



شکل ۴.۱۹: دیاگرام جسم آزاد



شکل ۵.۱۹: محورهای مختصات روی قسمت‌های مختلف ربات

با تعریف نمودن مختصات مرکز چرخش لحظه‌ای در دستگاه محلی به صورت  $ICR = (x_{ICR}, y_{ICR}) = (-d_{xC}, -d_{yC})$  داشت:

$$\frac{v_x}{y_{ICR}} = -\frac{v_y}{x_{ICR}} = w$$

بعد از جاگذاری نمودن روابط برای مولفه‌های سرعت هر چرخ می‌توانیم بنویسیم:

$$v_L = v_{1x} = v_{2x}$$

$$v_R = v_{3x} = v_{4x}$$

$$v_F = v_{1y} = v_{3y}$$

$$v_B = v_{2y} = v_{4y}$$

اگر  $\eta$  را به صورت  $\eta = [v_x, w]^T$  تعریف نماییم، می‌توانیم روابط زیر را نتیجه بگیریم (در آنها  $a, b, c$  پارامترهای ربات هستند که در شکل ۵.۱۹ نمایش داده شده است).

$$\begin{bmatrix} v_L \\ v_R \\ v_F \\ v_B \end{bmatrix} = \begin{bmatrix} 1 & -c \\ 1 & c \\ 0 & b - x_{ICR} \\ 0 & -a - x_{ICR} \end{bmatrix} \begin{bmatrix} v_x \\ w \end{bmatrix}$$

اگر فرض کنیم که شعاع موثر چرخ  $i$  ام، برابر با شعاع چرخ  $r$  می‌باشد، در این صورت داریم:

$$w_w = \begin{bmatrix} w_L \\ w_R \end{bmatrix} = \frac{1}{r} \begin{bmatrix} v_L \\ v_R \end{bmatrix}$$

که  $w_L, w_R$  به ترتیب سرعت زاویه‌ای چرخ‌های راست و چپ می‌باشد. با دو رابطه فوق می‌توان به رابطه زیر رسید:

$$\begin{bmatrix} v_x \\ w \end{bmatrix} = r \begin{bmatrix} \frac{w_L + w_R}{2} \\ \frac{-w_L + w_R}{2c} \end{bmatrix}$$

با در نظر گرفتن این ربات، محدودیت غیرهولونومیک را می‌توان به صورت  $v_y + x_{ICR}\dot{\theta} = 0$  بنویسیم. بنابراین مدل سینماتیکی ربات به صورت زیر نوشته می‌شود:

$$\begin{aligned} \dot{q} &= S(q)\eta \\ S(q) &= \begin{bmatrix} \cos \theta & x_{ICR} \sin \theta \\ \sin \theta & -x_{ICR} \cos \theta \\ 0 & 1 \end{bmatrix} \end{aligned}$$

در این بخش مدل دینامیکی ربات را بیان می کنیم. اگر  $F_i$  بیانگر نیروی فعال وارد شده بر چرخ  $i$  ام باشد و  $N_i$  نشان دهنده نیروی جاذبه باشد، واضح است که نیروی فعال به طور خطی با گشتاور وارد شده بر چرخ  $i$  ام است یعنی؛  $F_i = \frac{\tau_i}{r}$ ، از طرفی می دانیم که نیروهای مقاوم هم چه در راستای طولی و چه در راستای عرضی وجود دارد. با تقریب می توانیم این نیروها را با اصطکاک ساده تخمین بزنیم.

$$F_f(\sigma) = \mu_c N \operatorname{sgn}(\sigma) + \mu_v \sigma$$

بنابراین نیروی های اصطکاک وارد شده بر چرخ  $i$  ام را می توانیم به صورت زیر بنویسیم:

$$F_{li} = \mu_{ci} m g \operatorname{sgn}(v_{yi})$$

$$F_{si} = \mu_{sci} m g \operatorname{sgn}(v_{xi})$$

که  $\mu_{ci}, \mu_{sci}$  ضرایب پارامترهای نیروهای اصطکاک عرضی و طولی می باشند. با استفاده از روابط لاغرانژین می توانیم به رابطه زیر برای دینامیک ربات برسیم:

$$M(q)\ddot{q} + R(q) = B(q)\tau$$

اما رابطه فوق برای مدل سازی رباتی است که هیچ محدودیتی ندارد اما با توجه به محدودیت غیرهولونومیک باید آن را بهبود بدهیم. لذا خواهیم داشت:

$$M(q)\ddot{q} + R(q) = B(q)\tau + A^T(q)\lambda$$

با این حال از آنجاییکه برای کنترل، بیان کردن معادله بالا بر حسب  $\eta$  مناسب تر می باشد، بنابراین با ضرب کردن طرفین رابطه در  $S^T(q)$  خواهیم داشت:

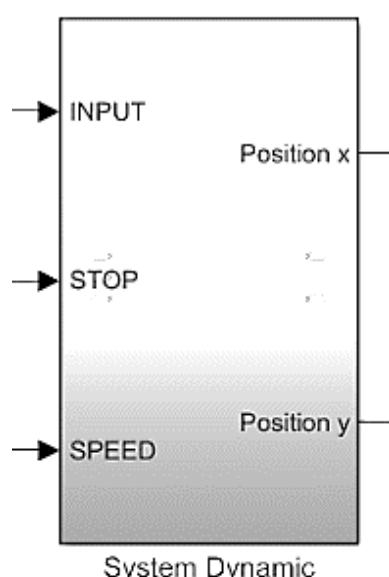
$$\bar{M}(q)\dot{\eta} + \bar{C}\eta + \bar{R} = \bar{B}\tau$$

$$\bar{C} = \begin{bmatrix} 0 & \dot{\theta} \\ \dot{\theta} & \dot{x}_{ICR} \end{bmatrix}, \bar{M} = \begin{bmatrix} m & 0 \\ 0 & mx^2_{ICR} + I \end{bmatrix}$$

$$\bar{B} = \frac{1}{r} \begin{bmatrix} 1 & 1 \\ -c & c \end{bmatrix}, \bar{R} = \begin{bmatrix} F_{rx} \\ x_{ICR}F_{ry} + M_r \end{bmatrix}$$

## مدل نهایی

مدل نهایی به صورت شکل ۶.۱۹ است. ورودی‌های سیستم سرعت خودرو (مقداری بین ۱۰۰- و ۱۰۰+)، جهت آن (۱ چپ، ۲ مستقیم و ۳ راست) و توقف (۰ یا ۱) هستند. همچنین خروجی‌های اصلی سیستم نیز موقعیت‌های  $X, Y$ , ربات هستند. (مقادیر منفی به معنی حرکت به سمت عقب می‌باشد). این مدل با پیاده سازی روابط مذکور در دو بخش قبل، ارتباط بین ورودی و خروجی‌های ربات را برقرار می‌کند. این مدل در نرم افزار متلب به صورت یک جعبه سیاه قرار داده شده و با اعمال انواع ورودی و خروجی، می‌توان مدلسازی یا طراحی کنترل کننده خودکار ربات را انجام داد. همچنین امکان کنترل دستی ربات زمینی با استفاده از همین مدل برقرار است. در ادامه به رابط گرافیکی طراحی شده در نرم افزار متلب، برای تسهیل ارتباط با این ربات به صورت دستی و خودکار معرفی می‌شود.



شکل ۶.۱۹: ورودی خروجی‌های  $UGV$

جدول ۱.۱۹: دستورات اعمالی به *UGV* برای کنترل دستی

<i>E</i>	<i>Q</i>	<i>D</i>	<i>A</i>	<i>S</i>	<i>W</i>
راست رو به جلو	چپ رو به عقب	چپ رو به عقب	راست رو به جلو	عقب	جلو

## ب.۲ ارتباط با *UGV*

برای کار با *UGV* هاب *USB* ارتباط با *NRF* را به کامپیوتر وصل نمایید. پیش از انجام هرکاری، پوشه *CILab* را از استاد خود گرفته یا از روی کامپیوتر آزمایشگاه بردارید و در لپ تاپ خود یا آدرسی متفاوت ذخیره نمایید (در غیراین صورت محتوای برنامه‌ی خودرا از دست خواهید داد). برای باز کردن *GUI* ارتباط با *UGV* بر روی فولدر جدید *CILab* و سپس *Debug* کلیک کرده و *CMVS-UGV.exe* را باز نمایید. حال شاهدیک محیط گرافیکی خواهید بود که در صورت اتصال هاب *USB* و روشن بودن *UGV*، با فشردن دکمه *Open*، خروجی سنسورهای *UGV* را نمایش می‌دهد. با فشردن دکمه سبز رنگ بر روی ربات، امکان روشن و خاموش کردن آن وجود دارد. در صورت اضطرار، با فشردن دکمه‌های کیبورد امکان کنترل ربات به صورت دستی از طریق این صفحه مطابق جدول وجود دارد. همچنین در صورت بروز مشکل می‌توانید دکمه قرمز رنگ در جلوی ربات (فیوز ربات) را فشار دهید و ربات را متوقف کنید. برای روشن کردن مجدد ربات پس از فشردن دکمه قرمز، لازم است تا دکمه‌را چرخانده و به سمت بیرون هدایت کنید. فراموش نکنید که ابتدا ربات را با استفاده از دکمه سبز رنگ خاموش کرده و سپس فیوز آن را وصل کنید.

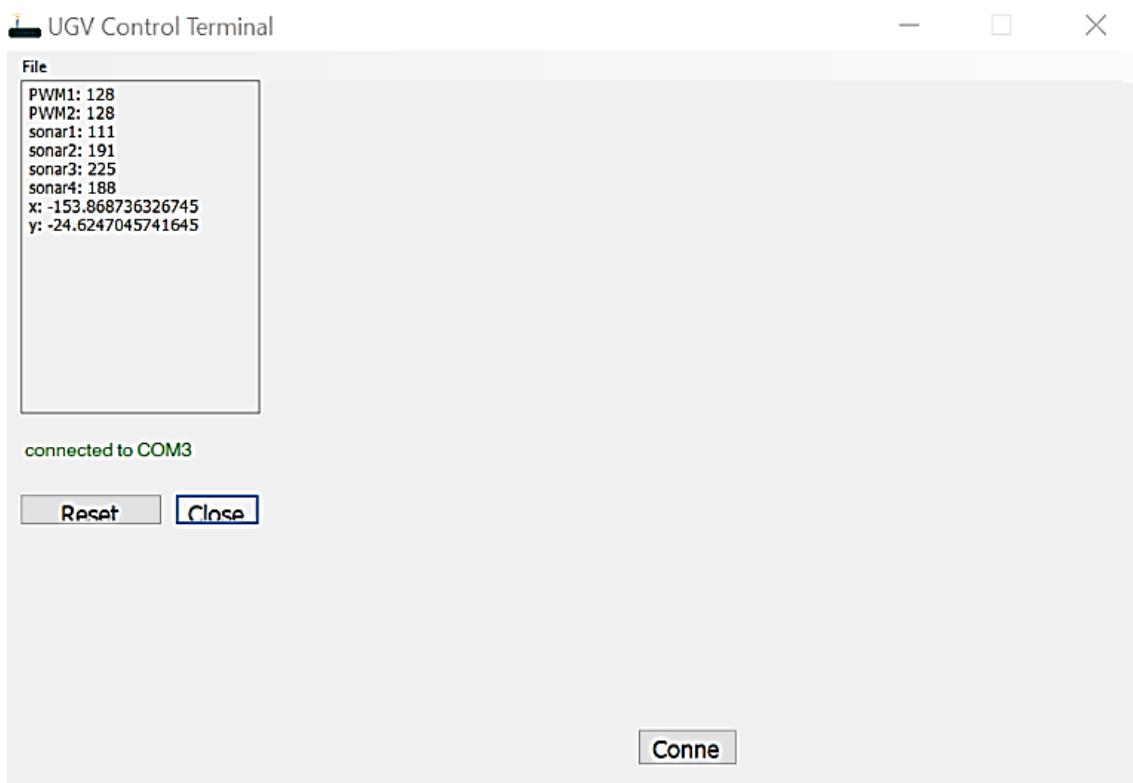
نکته: با فشردن دکمه *Reset*, مقادیر *X*, *Y* خوانده شده توسط سنسورها برابر با صفر قرار داده می‌شوند تا محل شروع به حرکت، به عنوان مبداء در نظر گرفته شود.

حال نرم افزار متلب را باز کرده، دکمه *CILabMatlabUGVGUI.fig* را فشرده و از آدرس *Open* را باز کنید. در این صورت با پنجره زیر مواجه خواهید شد.

## پنجره *UGVTest*

در این محیط می‌توانید با تعیین مقادیری، ورودی‌های ربات را تعریف کرده و با اجرا کردن برنامه، آن را به صورت اتوماتیک حرکت دهید.

نکته: اندازه‌ی سرعت ربات از ۰ تا ۱۰۰ است. برای حرکت هر موتور رو به جلو، از مقادیر مثبت و برای حرکت رو به عقب از مقادیر منفی استفاده می‌شود. همچنین جهت فرمان در سه زاویه در حین حرکت به جلو تعریف شده که شامل مقادیر ۱ برای چرخش به چپ، ۲ برای حرکت مستقیم و ۳ برای چرخش به راست است. نکته: در صورت بروز هرگونه مشکل یا برای ایجاد هرگونه تغییر، دکمه توقف را بزنید و از اجرا و متوقف کردن برنامه در حالت *START* بپرهیزید. با تعیین ورودی‌ها و فشردن دکمه *START*, در ضمن حرکت، در خروجی، مقادیر سنسورها به شرح زیرند.



شکل ۷.۱۹: صفحه ارتباط با *UGV*

جدول ۲.۱۹: خروجی های سیستم

خروجی ها	<i>X, Y</i>	<i>Theta</i>	<i>Battery</i>	<i>Right, Left PWMs</i>
مقدار	موقعیت	زاویه نسبی	شارژ باتری	سرعت موتورهای راست و چپ ربات
محدوده تغییرات	ابعاد محیط	۰ تا ۳۶۰	۰ تا ۱۰۰	۰ تا ۲۵۵
واحد	متر	درجه	درصد	-

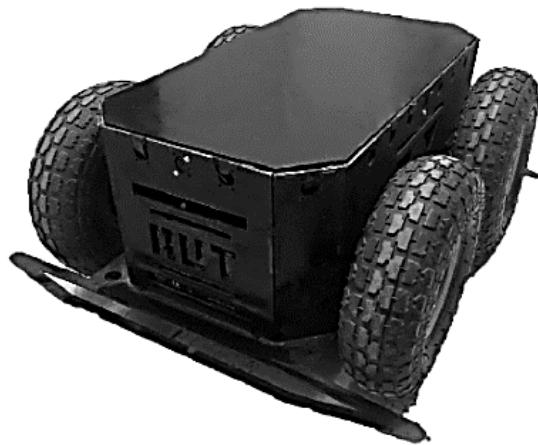
نکته: به میزان درصد باتری در طول آزمایش توجه کنید. در صورت رسیدن به عدد ۰ نیاز به تعویض و شارژ باتری‌ها وجود دارد.



**Amirkabir University of Technology  
(Tehran Polytechnic)**

### UGV TEST

IDENTIFIER SIM	IDENTIFIER REAL
CONTROLLER SIM	CONTROLLER REAL



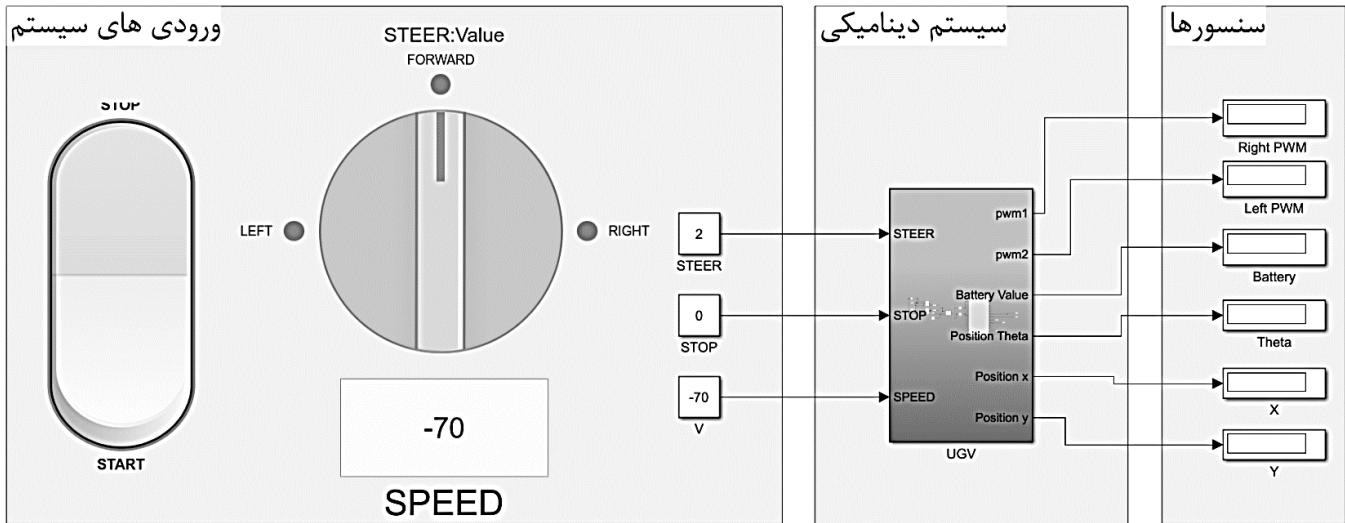
شکل ۸.۱۹: محیط کاربری *UGV*

### پنجره IDENTIFICATIONSIM

با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر ورودی‌های سیستم، محتوای سیستم دینامیکی و محتوای شناساگر وجود دارد. از این بلوک فقط برای شبیه‌سازی استفاده می‌شود. توضیحات بیشتر در بخش شناسایی توضیح داده می‌شود.



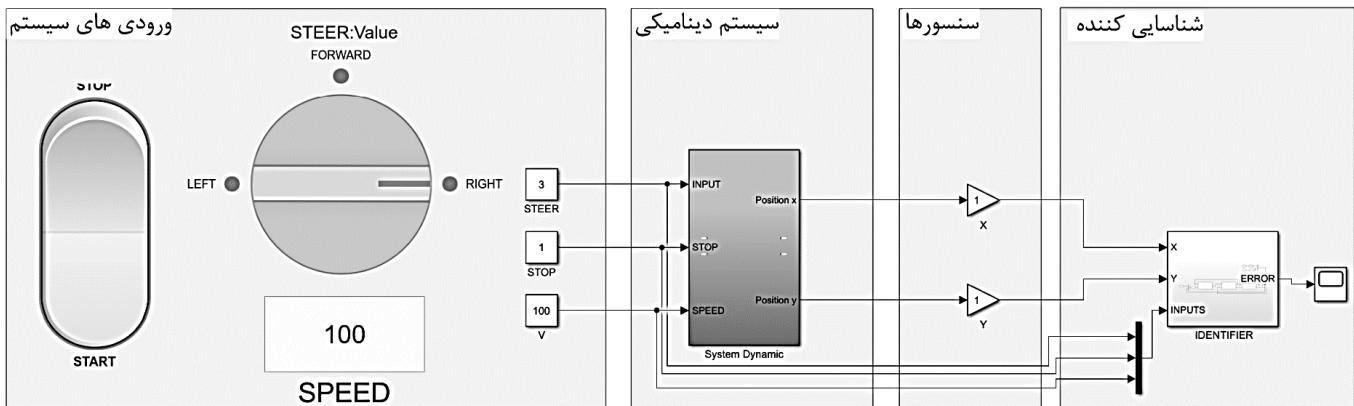
Amirkabir University of Technology  
(Tehran Polytechnic)



شکل ۹.۱۹: تست UGV



Amirkabir University of Technology  
(Tehran Polytechnic)



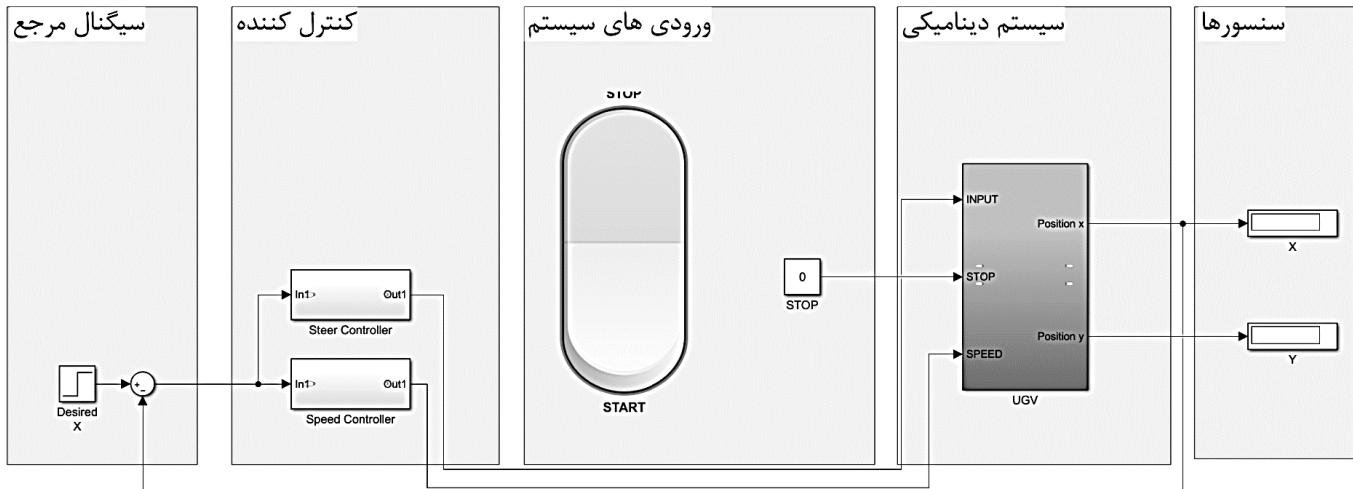
شکل ۱۰.۱۹: شبیه ساز شناساگر

## پنجره CONTROLERSIM

با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر سیگنال مرجع، محتوای سیستم دینامیکی و محتوای کنترل‌کننده‌ها وجود دارد. از این بلوک فقط برای شبیه‌سازی استفاده می‌شود. توضیحات بیشتر در دستورکار آورده شده است.



Amirkabir University of Technology  
(Tehran Polytechnic)



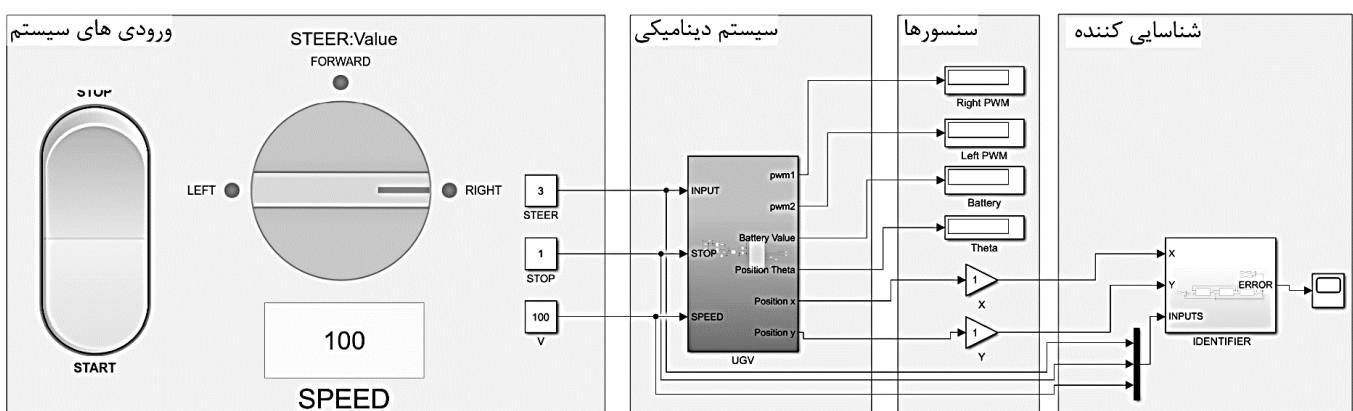
شکل ۱۱.۱۹: شبیه ساز کنترل کننده

## پنجره IDENTIFICATIONSIM

با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر ورودی‌های سیستم و محتوای شناساگر وجود دارد. توضیحات بیشتر در بخش شناسایی آورده شده است.



Amirkabir University of Technology  
(Tehran Polytechnic)

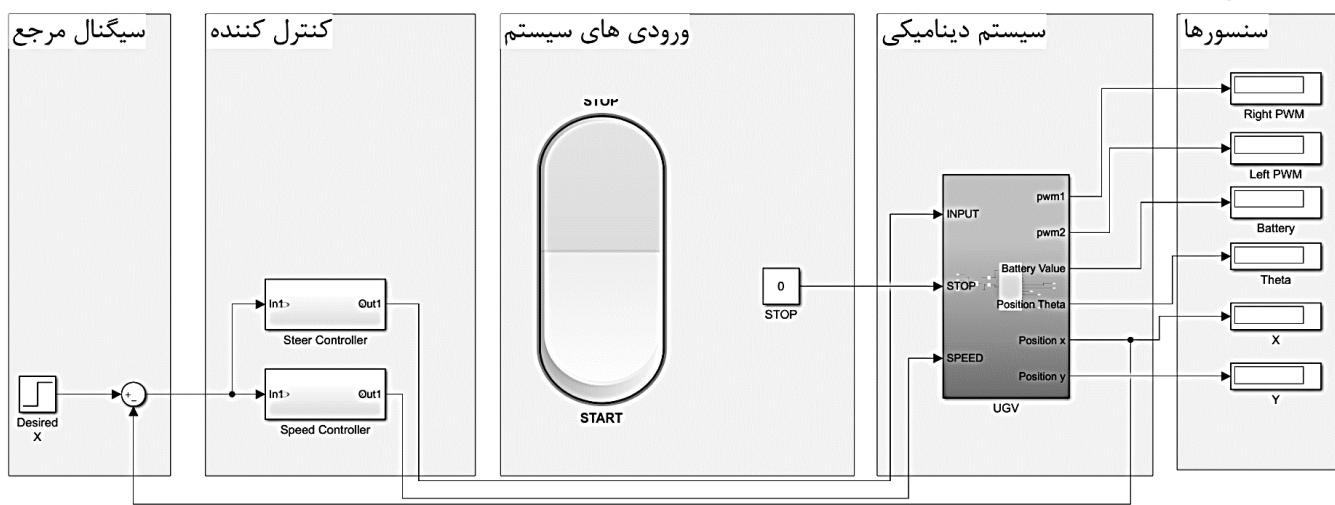


شکل ۱۲.۱۹: شناساگر واقعی UGV

با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر سیگنال مرجع، محتوای سیستم دینامیکی و محتوای کنترل‌کننده‌ها وجود دارد. از این بلوک فقط برای شبیه‌سازی استفاده می‌شود. توضیحات بیشتر در دستورکار آورده شده است.



Amirkabir University of Technology  
(Tehran Polytechnic)



شکل ۱۳.۱۹: کنترل واقعی UGV