



# مقدمه‌ای بر یادگیری ماشین

با کاربردهایی مبتنی بر 

An Introduction to Machine Learning  
With applications in R و Python

تألیف و ترجمه:

دکتر محمود فتحی

(استاد تمام دانشگاه)

علی محمد محمدی

(کارشناس ارشد هوش مصنوعی)

ناشر:

کانون نشر علوم



سرشناسه: فتحی، محمود، ۱۳۳۸ -، مترجم، گرددآورنده

عنوان و نام پدیدآور: مقدمه‌ای بر یادگیری ماشین با کاربردهایی مبتنی بر Python و R /An introduction to machine learning with.... = R

تألیف و ترجمه محمود فتحی، علی محمد محمدی

مشخصات نشر: تهران: کانون نشر علوم، ۱۳۹۸.

مشخصات ظاهری: ۳۳۰ ص: مصور (رنگی)، جدول، نمودار،

شابک: ۹۷۸۹۶۴۳۲۷۱۷۳۲

وضعیت فهرست‌نویسی: فیبا

موضوع: فرآگیری ماشینی

موضوع: Machine learning

موضوع: فرآگیری ماشینی -- نرم‌افزار

موضوع: Machine learning -- Computer software

موضوع: آر(زبان برنامه‌نویسی کامپیوتر)

موضوع: R (Computer program language)

شناسه افزوده: محمدی، علی محمد، ۱۳۶۰ -، مترجم، گرددآورنده

ردیbdندی کنگره: QC ۱۳۹۶ ۷۲۵/۵ ف۲

ردیbdندی دیوبنی: ۰۰۶/۳۱

شماره کتابشناسی ملی: ۴۶۸۵۵۶۰

نام کتاب: مقدمه‌ای بر یادگیری ماشین با کاربردهایی مبتنی بر R

ناشر: کانون نشر علوم

تألیف و ترجمه: دکتر محمود فتحی و علی محمد محمدی

ویراستار: فاطمه سلطان‌آبادی

صفحه‌آرا: فاطمه آدیگوزل‌پور اردبیلی

طرح جلد: زهرا سلطان‌آبادی

چاپ دوم: تابستان ۱۳۹۸

تیراز: ۱۰۰۰

چاپ و صحافی: کانون نشر علوم

قیمت:

شابک: ۹۷۸۹۶۴۳۲۷۱۷۳۲



نashreoloom

[www.nashreoloom.com](http://www.nashreoloom.com)

پیشگام در نشر آثار برگزیده علوم کاربردی کامپیووتر و الکترونیک

## تقدیم به:

فداوندی که سفنوران از ستدون او عاجزند و  
حسابگران از شمارش نعمت‌های او ناتوان و  
تلashگران از ادای حق<sup>۳</sup> او درمانده‌اند. فدایی که  
افکار ژرفاندیش، ذات او را درک (ادرار) نمی‌کنند  
و دست غواصان دریای علوه به او نفواده رسید.





## پیشگفتار

کتاب حاضر مقدمه‌ای بر یادگیری ماشین با استفاده از زبان R و Python می‌باشد. یادگیری ماشین یکی از حوزه‌های علمی مربوط به هوش مصنوعی است و امروزه به صورت گسترده در نرم‌افزارها و وب سایتها، بازی‌های کامپیوترا، در سیستم‌های کامپیوترا همه منظوره، در تلفن‌های هوشمند، دستگاه‌های بازی‌های کامپیوترا و غیره، به طریقی مورد استفاده قرار می‌گیرد. یادگیری ماشین نقش اساسی در هوشمندسازی زندگی بشر داشته و در دنیای امروز و آینده وجود نوعی از الگوریتم‌های یادگیری ماشین به طریقی به چشم خورده یا خواهد خورد.

الگوریتم و مدل‌های یادگیری ماشین برخلاف الگوریتم‌های سنتی برنامه‌ریزی کامپیوترا به صورت خودکار الگوهای موجود در مجموعه داده‌های متنوع را تشخیص و کشف کرده و با یادگیری از این داده‌ها (مجموعه داده آموزشی) مدلی می‌سازد که بتواند داده‌هایی که در مجموعه داده‌های آموزشی وجود نداشته (مجموعه داده آزمون) را تحلیل کرده و خروجی با کارایی بالا برای این داده‌های دیده نشده را تولید کند. مثلاً با یادگیری تغییرات نرخ ارز از مجموعه داده‌های موجود در دوره‌های زمانی گذشته، می‌تواند پیش‌بینی نرخ ارز را در آینده انجام دهد. در حوزه یادگیری ماشین مدل‌های متنوع و گسترده‌ای وجود دارد که طراحی و پیاده‌سازی یک مدل مناسب بستگی به نوع کاربرد؛ نوع داده‌ها و پارامترهای متعددی دارد که در این کتاب به طراحی برخی از این مدل‌ها و نیز کاربردهای مهم این حوزه پرداخته می‌شود. در این کتاب ضمن ورود به مبانی نظری و کاربردی مدل‌های مهم یادگیری ماشین در حد نیاز؛ موضوعات به نحوی ارائه می‌گردد که خواننده علاوه بر درک این مدل‌های یادگیری ماشین؛ تخصص‌های لازم جهت پیاده‌سازی عملی مدل‌های یادگیری ماشین را نیز فرا گیرد. به نحوی که با مطالعه کامل این کتاب خواننده قادر خواهد بود به عنوان یک متخصص یادگیری ماشین وارد بازار کار شده و پژوهش‌های جدید در حوزه‌های متنوع یادگیری ماشین؛ که بسیار مورد نیاز جامعه امروزی می‌باشد را طراحی و پیاده‌سازی کند.

این کتاب به زبان ساده مبانی نظری و عملی بسیاری از الگوریتم‌های مهم یادگیری ماشین را پوشش می‌دهد. این کتاب می‌تواند جهت دانشجویان دوره کارشناسی سال آخر گرایش‌های مختلف مهندسی کامپیوترا، فناوری اطلاعات، علوم کامپیوترا، ریاضی کاربردی، و صنایع و آمار مورد استفاده



قرار گیرد. همچنین دانشجویان کارشناسی ارشد گرایش هوش مصنوعی و سایر گرایش‌های ذکر شده می‌توانند به عنوان کتاب درسی؛ در درس یادگیری ماشین از این کتاب بهره ببرند. برای خواندن این کتاب، درس پیش نیاز خاصی مورد نیاز نیست و داشتن دانش ریاضی عمومی دوره کارشناسی رشته‌های مهندسی کفايت می‌کند. بنابراین کلیه علاقهمندان، اعم از دانشجویان گرایش‌های مختلف مهندسی، مهندسین و محققینی که به دلیل علاقه شخصی خواهان ورود به این حوزه هستند می‌توانند از این کتاب بهره ببرند و تخصص یادگیری ماشین را کسب کرده و وارد بازار کار این حوزه؛ که بسیار فراگیر هم هست؛ شوند. در نگارش این کتاب سعی شده که در انتهای هر فصل مثال‌های عملی با استفاده از زبان R و Python گنجانده شود. کتاب شامل ۱۷ فصل و دو پیوست است که فصل‌های آن اصول طراحی و پیاده‌سازی الگوریتم‌های مهم یادگیری ماشین را به زبانی ساده ارائه می‌دهد. پیوست ۱ اصول روابط و عملیات ماتریس‌ها که به صورت گسترده در این حوزه استفاده می‌شوند را مختصرًا تو ضیح می‌دهد. پیوست ۲ آشنایی مقدماتی ولی کافی در ارتباط با زبان R و بسته‌های نرم‌افزاری مربوط به یادگیری ماشین است و نحوه استفاده از زبان R و این بسته‌ها را آموزش می‌دهد. لازم به ذکر است که زبان R و Python یک زبان منبع باز است که به صورت گسترده در آمار و ریاضیات مورد استفاده قرار می‌گیرد و در حوزه یادگیری ماشین به عنوان یک ابزار کامل و نسبتاً جدید مطرح است. تا کنون چندین هزار جعبه ابزار و بسته‌های نرم‌افزاری آماده به کار و قابل استفاده با زبان R و Python توسعه یافته است که به سادگی کاربر می‌تواند با استفاده از این جعبه ابزار و نرم‌افزارها پروژه‌های نوین و جدید در حوزه‌های متنوع یادگیری ماشین را پیاده‌سازی کند. نحوه استفاده از این زبان و بسته‌های نرم‌افزاری متنوع آن با انواع فرمتهای گوناگون داده‌ها در این کتاب بحث شده است.

این کتاب از جمله منابع محدود به زبان فارسی در موضوع یادگیری ماشین است و هدف نویسنده‌گان از نگارش آن گسترش این علم و فناوری بسیار مهم و پرکاربرد جهت استفاده جامعه علمی به ویژه جامعه علمی مهندسی کامپیوتر کشور می‌باشد. در نگارش این کتاب مقدماتی؛ برخی از جدیدترین موضوعات یادگیری ماشین نیز؛ به صورت نظری و عملی به زبان ساده مورد بحث قرار گرفته است. با توجه به ذکر مثال‌هایی با زبان R و Python این کتاب می‌تواند کاملاً جنبه‌های کاربردی را نیز پوشش داده و خواننده را به یک متخصص تئوری - عملی در این حوزه تبدیل نماید.



اگرچه مطالب اساسی و مهم یادگیری ماشین در این کتاب مطرح شده و نویسنده‌گان تمامی تلاش خود را در نگارش مباحث اصلی این حوزه به کار برده‌اند ولی بدلیل تنوع و گستردگی این موضوع طبیعتاً با توجه به محدودیت در حجم کتاب، کاستی‌هایی در بخشی از موضوعات نیز وجود دارد.

یکی از ویژگی‌های این کتاب تأکید بر زبان R و Python است که امروزه به عنوان یک ابزار کامل و نوین در حوزه یادگیری ماشین مطرح است و با توجه به منبع باز بودن آن روز به روز به تعداد بسته‌های نرم‌افزاری و جعبه‌ابزارهای جدید در حوزه‌های متداول و جدید مانند یادگیری عمیق و سایر حوزه‌های نوین از طریق اینترنت به صورت رایگان در اختیار کاربران و علاقهمندان می‌تواند قرار گیرد. این کتاب مبانی و مقدمات و اصول مدل‌های یادگیری ماشین را به زبان نسبتاً روان و با ذکر مثال‌های کاربردی، با استفاده از زبان R و Python ارائه داده است. ترکیب این دو ویژگی این کتاب را به کتابی خاص در این حوزه تبدیل کرده است.

امید است این کتاب که جزء اندک منابع فارسی در حوزه یادگیری ماشین است، گام کوچکی در گسترش این علم در کشور باشد و همچنین سایر متخصصین و محققین را تشویق به نگارش مطالب کامل‌تر در این حوزه نماید.

در انتهای خوانندگان تقاضا می‌شود که کاستی‌ها و اشکالات احتمالی را جهت بهبود کیفیت کتاب در ویرایش‌های بعدی به آدرس [mahfathy@yahoo.com](mailto:mahfathy@yahoo.com) ارسال نمایند.



## فهرست مطالب

۵	پیشگفتار
۱۹	مقدمه

۲۵

### فصل ۱: رگرسیون

۲۶	۱-۱- رگرسیون
۲۶	۱-۲- رگرسیون خطی تک متغیره
۲۷	۱-۳- تقسیم داده‌ها به داده‌های آموزشی و آزمون
۲۹	۱-۴- تابع هزینه
۳۱	۱-۴-۱- تشریح روش محاسبه تابع هزینه
۳۵	۱-۴-۲- نگاهی عمیق‌تر به تابع هزینه
۳۷	۱-۵- ۱- گرادیان نزولی
۳۸	۱-۵-۱- رابطه گرادیان نزولی
۳۸	۱-۵-۲- الگوریتم گرادیان نزولی
۳۹	۱-۵-۳- نرخ یادگیری در گرادیان نزولی
۴۰	۱-۵-۴- اندازه گام‌های نرخ یادگیری در گرادیان نزولی
۴۲	۱-۶- گرادیان نزولی و تابع هزینه در حل رگرسیون خطی
۴۳	۱-۷- مثال عملی برای یک رگرسیون خطی تک متغیره با زبان R
۴۸	۱-۸- مثال عملی برای یک رگرسیون خطی تک متغیره با زبان پایتون

۵۱

### فصل ۲: رگرسیون خطی چند متغیره

۵۲	۲-۱- رگرسیون خطی چند متغیره
۵۴	۲-۲- گرادیان نزولی برای رگرسیون خطی چندمتغیره
۵۵	۲-۳- بهروزرسانی پارامترها در گرادیان نزولی
۵۵	۲-۴- الگوریتم بهروزرسانی پارامترهای چند متغیره
۵۶	۲-۵- مقیاس‌گذاری ویژگی‌ها
۵۸	۲-۶- نرمال کردن متوسط
۵۸	۲-۷- نرخ یادگیری در رگرسیون خطی چند متغیره
۵۹	۲-۸- درجه چندجمله‌ای معادله فرضیه رگرسیون خطی



۶۱	- مثال عملی برای یک رگرسیون خطی چند متغیره با R.
۶۳	- مثال عملی برای یک رگرسیون خطی چند متغیره با زبان پایتون

## ۶۵

## فصل ۳: رگرسیون لاجستیک

۶۶	-۱- رگرسیون لاجستیک
۶۷	-۲- دسته‌بندی با استفاده از رگرسیون لاجستیک
۶۷	-۳-۱- معرفی خط فرضیه در رگرسیون لاجستیک
۶۷	-۳-۲- تابع سیگموئید
۶۸	-۳-۳- مرز تصمیم در رگرسیون لاجستیک
۶۹	-۳-۴- روش محاسبه محل مرز تصمیم در فضای نمونه‌ها
۷۰	-۳-۵- مرز تصمیم غیر خطی
۷۱	-۳-۶- تابع هزینه
۷۲	-۳-۷- تابع هزینه در رگرسیون لاجستیک
۷۳	-۳-۸- نقطه کمینه تابع هزینه
۷۳	-۳-۹- نقطه بیشینه تابع هزینه
۷۴	-۳-۱۰- گرادیان نزولی برای رگرسیون لاجستیک
۷۴	-۳-۱۱- رابطه تابع هزینه در رگرسیون لاجستیک
۷۵	-۳-۱۲- الگوریتم گرادیان نزولی برای رگرسیون لاجستیک
۷۶	-۳-۱۳- رگرسیون لاجستیک چند کلاسه
۷۸	-۳-۱۴- پیاده‌سازی یک رگرسیون لاجستیک با استفاده از زبان R
۸۳	-۳-۱۵- پیاده‌سازی یک رگرسیون لاجستیک با استفاده از زبان پایتون

## ۸۵

## فصل ۴: راهکارهایی جهت بهبود طراحی الگوریتم‌های یادگیری ماشین

۸۶	-۴- راهکارهایی جهت بهبود طراحی الگوریتم‌های یادگیری ماشین
۸۶	-۴-۱- تعمیم‌پذیری الگوریتم
۸۶	-۴-۲- عوامل رسیدن الگوریتم به حالت بهینه
۸۶	-۴-۳- ظرفیت یک فرضیه
۸۷	-۴-۴- ایجاد مدل مناسب برای الگوریتم یادگیری
۸۸	-۴-۵- مفهوم بیش برازش و کم برازش، بایاس و واریانس



۸۸.....	۱-۶-۴
۸۹.....	۲-۶-۴
۸۹.....	۳-۶-۴
۹۰.....	۷-۶-۴
۹۱.....	۸-۶-۴
۹۱.....	۹-۶-۴
۹۲.....	۱۰-۶-۴
۹۲.....	۱۱-۶-۴
۹۳.....	۱۲-۶-۴
۹۴.....	۱۳-۶-۴
۹۵.....	۱۴-۶-۴
۹۵.....	۱۵-۶-۴

## ۹۷

## فصل ۵: شبکه عصبی

۹۸.....	۱-۱-۵
۹۸.....	۱-۱-۵
۹۸.....	۲-۱-۵
۹۹.....	۱-۲-۵
۹۹.....	۳-۱-۵
۱۰۰.....	۱-۳-۵
۱۰۱.....	۲-۳-۵
۱۰۳.....	۴-۳-۵
۱۰۴.....	۴-۴-۵
۱۰۴.....	۲-۴-۵
۱۰۴.....	۳-۴-۵
۱۰۵.....	۴-۴-۵
۱۱۰.....	۵-۴-۵
۱۱۳.....	۶-۴-۵



## فصل ۶: ماشین بردار پشتیبان

۱۱۶.....	۱- ماشین بردار پشتیبان.....
۱۱۸.....	۲- تابع هزینه ماشین بردار پشتیبان.....
۱۱۹.....	۳- دسته‌بندی بر مبنای بیشترین حاشیه.....
۱۱۹.....	۴- نمودار تابع هزینه در ماشین بردار پشتیبان.....
۱۲۱.....	۵- ایجاد خط تصمیم برای تفکیک دسته‌ها.....
۱۲۲.....	۶- مباحث ریاضی مربوط به دسته‌بندی براساس حاشیه زیاد.....
۱۲۳.....	۷- مقیاس یک بردار.....
۱۲۴.....	۸- مباحث ریاضی خط تصمیم.....
۱۲۷.....	۹- تابع هسته ماشین بردار پشتیبان.....
۱۲۸.....	۱۰- معرفی تابع هسته و عملکرد آنها.....
۱۲۸.....	۱۱- تابع هسته گوسی.....
۱۲۹.....	۱۲- محاسبه شباهت.....
۱۳۲.....	۱۳- نحوه تعیین نشانه.....
۱۳۳.....	۱۴- رابطه ماشین بردار پشتیبان براساس تابع هسته.....
۱۳۴.....	۱۵- اهمیت مقداردهی مناسب پارامترها.....
۱۳۴.....	۱۶- مثالی برای دسته‌بندی به روش ماشین بردار پشتیبان با زبان R.....
۱۳۷.....	۱۷- مثالی برای دسته‌بندی به روش ماشین بردار پشتیبان با زبان پایتون.....

## فصل ۷: یادگیری بدون ناظارت

۱۴۰.....	۱- یادگیری بدون ناظارت.....
۱۴۰.....	۲- روش‌های خوشبندی.....
۱۴۰.....	۳- خوشبندی سخت.....
۱۴۰.....	۴- خوشبندی نرم.....
۱۴۱.....	۵- انواع مختلف الگوریتم خوشبندی.....
۱۴۱.....	۶- نحوه انتخاب تعداد خوشبندی.....
۱۴۲.....	۷- ساختار کلی الگوریتم k-means.....
۱۴۴.....	۸- الگوریتم k-means و روابط محاسباتی آن.....
۱۴۵.....	۹- توضیحات الگوریتم k-means.....



۱۴۶	۳-۵-۷- مقداردهی اولیه الگوریتم k-means
۱۴۷	۶-۷- روش اجتناب از مقداردهی اولیه نامناسب در خوشه‌بندی
۱۴۷	۷-۷- انتخاب تعداد خوشه‌ها در روش‌های بدون نظارت
۱۴۸	۱-۷-۷- انتخاب تعداد خوشه به صورت خودکار
۱۴۸	۱-۷-۷-۷- روش آرنج در انتخاب خودکار تعداد خوشه
۱۴۹	۲-۷-۷- انتخاب تعداد خوشه به صورت دستی
۱۴۹	۸-۷- جرای الگوریتم k-means با استفاده از زبان R
۱۵۲	۹-۷- جرای الگوریتم k-means با استفاده از زبان پایتون

#### ۱۵۵ فصل ۸: تحلیل مؤلفه‌های اصلی

۱۵۶	۱-۸- تحلیل مؤلفه‌های اصلی
۱۵۶	۲-۸- کاربردهای تحلیل مؤلفه اصلی
۱۵۶	۳-۸- حذف ویژگی و کاهش ابعاد
۱۵۶	۱-۳-۸- روش کاهش تعداد ویژگی
۱۵۸	۴-۸- مصورسازی داده‌ها با استفاده از تحلیل مؤلفه‌های اصلی
۱۵۹	۵-۸- نحوه عملکرد الگوریتم تحلیل مؤلفه اصلی
۱۶۰	۶-۸- کاهش سه بعد به دو بعد
۱۶۱	۷-۸- مراحل اجرای الگوریتم تحلیل مؤلفه اصلی
۱۶۲	۸-۸- روش به دست آوردن تعداد مؤلفه‌های اصلی (k)
۱۶۳	۹-۸- برگشت داده‌های حذف شده
۱۶۳	۱-۹-۸- مراحل عکس فشرده‌سازی
۱۶۴	۱۰-۸- مثال عملی برای تحلیل مؤلفه اصلی با زبان R
۱۶۷	۱۱-۸- مثال عملی برای تحلیل مؤلفه اصلی با زبان پایتون

#### ۱۶۹ فصل ۹: تشخیص ناهنجاری

۱۷۰	۱-۹- تشخیص ناهنجاری
۱۷۱	۲-۹- کاربردهای مختلف تشخیص ناهنجاری
۱۷۱	۱-۲-۹- تشخیص تقلب
۱۷۲	۳-۹- میاحث ریاضی برای تعیین ناهنجاری
۱۷۲	۱-۳-۹-تابع گوسی



۱۷۴.....	۲-۳-۹ تخمین پارامترهایتابع گوسی
۱۷۵.....	۳-۳-۹ الگوریتم تشخیص ناهنجاری
۱۷۶.....	۴-۳-۹ مراحل الگوریتم تشخیص ناهنجاری
۱۷۶.....	۴-۹ ارزیابی عملکرد الگوریتم تشخیص ناهنجاری

## ۱۸۱

## فصل ۱۰: یادگیری تقویتی

۱۸۲.....	۱-۱-۱۰ یادگیری تقویتی
۱۸۲.....	۲-۱۰ حوزه عملکرد
۱۸۳.....	۳-۱۰ الگوریتم‌های یادگیری تقویتی
۱۸۳.....	۴-۱۰ روش یادگیری Q
۱۸۵.....	۵-۱۰ آموزش عامل در روش Q
۱۸۵.....	۶-۱۰ مراحل الگوریتم یادگیری Q
۱۸۶.....	۷-۱۰ نرخ یادگیری در روش یادگیری Q
۱۸۶.....	۸-۱۰ عامل تخفیف در روش یادگیری Q
۱۸۶.....	۹-۱۰ ماتریس R یا پاداش در یادگیری Q
۱۸۶.....	۱۰-۴-۶ یک دور آموزشی از عمل عامل در آموزش Q
۱۸۶.....	۱۰-۷-۱ یک مثال مفهومی برای الگوریتم Q
۱۸۹.....	۱۰-۸-۱ برنامه یادگیری Q با زبان R

## ۱۹۳

## فصل ۱۱: سیستم‌های توصیه‌گر

۱۹۴.....	۱-۱۱ سیستم‌های توصیه‌گر
۱۹۴.....	۲-۱۱ تعریف آیتم در سیستم‌های توصیه‌گر
۱۹۴.....	۳-۱۱ تعریف کاربر در سیستم‌های توصیه‌گر
۱۹۵.....	۴-۱۱ عملکردها و اهداف سیستم‌های توصیه‌گر
۱۹۶.....	۵-۱۱ داده و منبع دانش
۱۹۶.....	۶-۱۱ تراکنش
۱۹۶.....	۷-۱۱ امتیاز ارزیابی
۱۹۷.....	۸-۱۱ روش‌های توصیه‌گری
۱۹۷.....	۹-۱۱ انواع رویکردهای اصلی پیاده‌سازی سیستم توصیه‌گر
۱۹۷.....	۱۰-۱۱ بر مبنای محتوا



۱۹۷	۲-۷-۱۱	- فیلتر مشترک
۱۹۷	۳-۷-۱۱	- جمعیتی
۱۹۸	۴-۷-۱۱	- برنمایی دانش
۱۹۸	۵-۷-۱۱	- مبتنی بر جامعه
۱۹۸	۶-۷-۱۱	- توصیه گر ترکیبی
۱۹۸	۸-۱۱	- کاربردهای سیستم توصیه گر
۱۹۸	۹-۱۱	- روابط ریاضی و پیاده‌سازی سیستم توصیه گر
۱۹۹	۱۰-۱۱	- محاسبه شباهت با رابطه پیرسون در فیلتر مشترک
۲۰۰	۱۱-۱۱	- پیاده‌سازی فیلتر مشترک با کدهای R

## ۲۰۱

## فصل ۱۲: داده‌های جریانی

۲۰۲	۱-۱۲	- داده‌های جریانی
۲۰۲	۲-۱۲	- محدودیت‌های داده‌های جریانی
۲۰۳	۳-۱۲	- تعاریف داده‌های جریانی
۲۰۳	۱-۳-۱۲	- تعریف جریان داده
۲۰۳	۲-۳-۱۲	- تعریف مفهوم
۲۰۳	۳-۳-۱۲	- تعریف رانش مفهوم
۲۰۳	۱-۳-۳-۱۲	- انواع رانش مفهوم
۲۰۳	۴-۳-۱۲	- تعریف پنجره زمانی
۲۰۳	۱-۴-۳-۱۲	- انواع پنجره زمانی
۲۰۴	۴-۱۲	- رویکردهای محاسباتی در داده‌های جریانی
۲۰۴	۵-۱۲	- اعتبارسنجی در داده‌های جریانی
۲۰۴	۱-۵-۱۲	- روش‌های اعتبارسنجی در داده‌های جریانی
۲۰۴	۶-۱۲	- روش‌های یادگیری در داده‌های جریانی
۲۰۵	۱-۶-۱۲	- درخت هافدینگ
۲۰۵	۲-۶-۱۲	- دسته‌بند بیزین در داده‌های جریانی
۲۰۵	۳-۶-۱۲	- شبکه عصبی در داده‌های جریانی
۲۰۶	۴-۶-۱۲	- دسته‌بند نزدیک‌ترین همسایه
۲۰۶	۵-۶-۱۲	- درخت سازگار تصمیم مبتنی بر رویکرد یکی در برابر بقیه



۲۰۶	۶-۶-۶-۱۲- انتخاب پویای ویژگی در داده‌های جریانی
۲۰۶	۶-۶-۱- مدل فیلتر در انتخاب ویژگی‌های داده‌های جریانی
۲۰۷	۶-۶-۲- مدل پوششی انتخاب ویژگی‌های داده‌های جریانی
۲۰۷	۶-۷- کاوش در جریان‌های متنه

### فصل ۱۳: ارزیابی کارایی در یادگیری ماشین

۲۱۰	۱۳-۱- ارزیابی کارایی در یادگیری ماشین
۲۱۰	۱۳-۲- ماتریس درهم‌ریختگی در ارزیابی کارایی الگوریتم
۲۱۱	۱۳-۳- ارزیابی نرخ مثبت درست در استخراج داده
۲۱۱	۱۳-۴- ارزیابی نرخ مثبت نادرست در استخراج داده
۲۱۱	۱۳-۵- ارزیابی نرخ منفی درست در استخراج داده
۲۱۱	۱۳-۶- ارزیابی نرخ منفی اشتباه در استخراج داده
۲۱۱	۱۳-۷- روش ارزیابی دقت الگوریتم
۲۱۱	۱۳-۸- روش ارزیابی درستی الگوریتم و حساسیت
۲۱۲	۱۳-۹- روش ارزیابی اختصاصی بودن
۲۱۲	۱۳-۱۰- روش ارزیابی صحت الگوریتم
۲۱۲	۱۳-۱۱- F-Measure
۲۱۲	۱۳-۱۲- منحنی ROC
۲۱۳	۱۳-۱۳- سطح زیر منحنی (AUC)
۲۱۴	۱۳-۱۴- مثالی با R برای ترسیم منحنی ROC
۲۱۶	۱۳-۱۵- ترسیم نمودار ROC با استفاده از پایتون

### فصل ۱۴: مدل‌های گرافی احتمالاتی

۲۲۰	۱۴-۱- مدل‌های گرافی احتمالاتی
۲۲۱	۱۴-۲- بازنمایی مدل‌های گرافی
۲۲۱	۱۴-۳- شبکه‌های بیزی
۲۲۲	۱۴-۴- شبکه‌های مارکوف
۲۲۳	۱۴-۱- استقلال متغیرها در شبکه‌های مارکوف
۲۲۴	۱۴-۵- عمل استنتاج در مدل‌های گرافی
۲۲۵	۱۴-۶- مدل‌های مولد و تمایزی در مدل‌های گرافی



۲۲۵	۷-۱۴	- ساختار مدل گرافی .....
۲۲۶	۸-۱۴	- به دست آوردن پارامترهای مدل های گرافی .....
۲۲۶	۹-۱۴	- توزیع احتمال توان مدل گرافی ترکیبی .....
۲۲۶	۱۰-۱۴	- یادگیری پارامترها در مدل های گرافی .....
۲۲۷	۱۱-۱۴	- استنتاج در مدل های گرافی .....
۲۲۹	۱۲-۱۴	- برنامه بیزین با کدهای R .....
۲۳۰	۱۳-۱۴	- برنامه بیزین با زبان پایتون .....

## ۲۳۳

## فصل ۱۵: مقدمه‌ای بر یادگیری عمیق

۲۳۴	۱-۱۵	- مقدمه‌ای بر یادگیری عمیق .....
۲۳۴	۲-۱۵	- دسته‌بندی انواع روش‌های یادگیری عمیق .....
۲۳۴	۲-۱۵	- ۱-۲-۱۵
۲۳۴	۲-۲-۱۵	- مدل های مولد .....
۲۳۵	۳-۲-۱۵	- مدل ترکیبی در روش‌های یادگیری عمیق .....
۲۳۵	۳-۱۵	- استخراج ویژگی خودکار .....
۲۳۶	۴-۱۵	- خود رمز کننده .....
۲۳۶	۴-۴-۱۵	- عملکرد کلی یک خود رمز کننده .....
۲۳۷	۴-۴-۱۵	- ۲-۴-۱۵
۲۳۷	۵-۱۵	- ماشین بولتزمن .....
۲۳۸	۵-۱۵	- ۱-۵-۱۵
۲۴۰	۶-۱۵	- بازسازی در الگوریتم ماشین بولتزمن .....
۲۴۱	۶-۱۵	- شبکه عصبی بازگشتی .....
۲۴۲	۷-۱۵	- شبکه باور عمیق .....
۲۴۲	۷-۷-۱۵	- ۱- ساختار شبکه باور عمیق .....
۲۴۳	۸-۱۵	- شبکه عصبی کانولوشن .....
۲۴۵	۸-۱۵	- ۱-۸-۱۵
۲۴۵	۸-۸-۱۵	- ورودی شبکه عصبی کانولوشن .....
۲۴۵	۸-۳-۸-۱۵	- لایه کانولوشن .....
۲۴۶	۴-۸-۱۵	- ۴- تصحیح واحد خطی (ReLU) .....



۲۴۶	لایه ادغام	۱۵-۸-۵
۲۴۷	لایه تماماً متصل	۱۵-۸-۶

#### فصل ۱۶: روش‌هایی جهت افزایش کارایی الگوریتم‌های یادگیری ماشین

۲۵۰	روش‌هایی جهت افزایش کارایی الگوریتم‌های یادگیری ماشین	۱۶-۱
۲۵۱	بهبود کارایی مدل با روش مجمع	۱۶-۲
۲۵۳	بهبود کارایی روی داده‌ای حجمی یا حتی مجموعه داده‌های کوچک	۱۶-۲-۱
۲۵۳	توانایی سنتز داده از حوزه‌های مجزا	۱۶-۲-۲
۲۵۳	درک بهتر از عملیات یادگیری پیچیده	۱۶-۲-۳
۲۵۴	روش Bagging	۱۶-۲-۴
۲۵۴	Boosting	۱۶-۲-۵
۲۵۵	Random forest (RF)	۱۶-۲-۶
۲۵۵	افزایش کارایی در R از نظر زمان اجرا	۱۶-۳
۲۵۶	مدیریت مجموعه داده‌های خیلی بزرگ	۱۶-۴
۲۵۶	یادگیری سریع‌تر با پردازش موازی	۱۶-۵
۲۵۷	روش اندازه‌گیری زمان اجرا	۱۶-۶
۲۵۷	پردازش موازی در محاسبات ابری Hadoop و Mapreduce	۱۶-۷
۲۵۸	واحد پردازش گرافیکی (GPU)	۱۶-۸
۲۵۸	پیاده‌سازی مؤثرتر الگوریتم یادگیری ماشین با R	۱۶-۹

#### پیوست ۱: مقدمه‌ای بر جبر خطی (مبانی عملیات ماتریس‌ها)

#### پیوست ۲: آموزش زبان برنامه‌نویسی R

#### پیوست ۳: آموزش مقدماتی پایتون

#### نمایه (انگلیسی و فارسی)

۳۱۴	«انگلیسی به فارسی»
۳۲۱	«فارسی به انگلیسی»

#### منابع و مراجع



## **مقدمة**

ما در روزگاری به سر می‌بریم که داده‌های عظیم نقش اساسی در بیشتر بخش‌های جامعه‌ای که به هر حال به نوعی با کامپیوتر سر و کار دارند، بازی می‌کنند. امروزه در بیشتر بخش‌های زندگی بشر مانند تجارت، بازاریابی، مالی و اقتصادی، پژوهشی و بیوانفورماتیک، کاربردهای مرتبط با اینترنت و شبکه‌های کامپیوتري مانند جستجوی اطلاعات و غیره و به طور خلاصه از طبیعت، جامعه یا پردازش‌های صنعتی، داده‌های گوناگونی با حجم بسیار بزرگ تولید می‌شوند. این داده‌ها ممکن است با زمان یا وابسته به عوامل مختلفی تغییر کنند. استفاده از آن دسته از بر نامه‌های کامپیوتري که "با استفاده از الگوريتم‌های سنتی توسعه یافته‌اند" برای پردازش این داده‌های با حجم و تنوع وسیع؛ جهت اجرای هوشمندانه‌تر؛ استخراج دانش و یا به ویژه "جهت تصمیم‌گیری یا پیش‌بینی در بسیاری از کاربردها" در هر یک از حوزه‌های بالا دشوار یا غیرممکن است.

استفاده از الگوريتم‌ها و سیستم‌های مبتنی بر یادگیری ماشین که به صورت خودکار الگوهای موجود در این داده‌های متنوع را تشخیص و یا کشف کرده؛ یا اصطلاحاً این "داده‌های آموزشی" را یاد بگیرند و سپس سیستمی ایجاد شود که قادر باشد "تصمیم‌گیری یا پیش‌بینی‌هایی" بر روی "داده‌های جدیدی که در مجموعه داده‌های آموزشی نبوده" را با کارایی بالا انجام دهد؛ امروزه مورد توجه گسترده دانشمندان علوم کامپیوتري به ویژه علوم داده و هوش مصنوعی است.

به طور خلاصه یادگیری ماشین عبارت است از برنامه‌ریزی خودکار کامپیوتري و ساخت یک مدل یادگیرنده که با آموزش و کسب تجربه، برای انجام وظیفه و عمل خاصی آماده شود و بتواند در مواجهه با شرایط جدید محاسبات قابل قبولی را در راستای انجام عمل مورد نظر انجام دهد.

به عبارتی دیگر، در یادگیری ماشین الگوريتم‌های یادگیرنده‌ای ایجاد می‌شود که بتواند با استفاده از داده‌های موجود که اصطلاحاً داده‌های آموزشی گفته می‌شود تجربیاتی را به دست آورد که در مواجهه با داده‌های جدید و دیده نشده در مرحله آموزش، که به آن داده‌های آزمون گفته می‌شود، پاسخ مناسبی را برای مسئله مورد نظر به وجود آورد.

نوع عملیات<sup>۱</sup> در یادگیری ماشین را می‌توان به گروه‌های یادگیری با نظارت<sup>۲</sup> یا مدل‌های پیش‌بینی کننده<sup>۳</sup> که به صورت دسته‌بندی و رگرسیون می‌باشند و یادگیری بدون نظارت<sup>۴</sup> که دانش را از داده‌ها استخراج کرده و به انواع: مدل کردن احتمال توزیع تابع تخمین و یا کشف ساختارهای موجود در داده‌ها (خوشه‌بندی، کاهش ابعاد و غیره) تقسیم‌بندی می‌شوند. همچنین در روش‌های یادگیری تقویتی<sup>۵</sup> با عاملی روبه‌رو هستیم که از طریق سعی و خطا با محیط تعامل کرده و یاد می‌گیرد عملی بهینه یا تصمیم‌گیری مناسب در یک محیط برای رسیدن به هدف مطلوب در محیط را انجام دهد.

- 
1. Task
  2. Supervised learning
  3. Predictor
  4. Unsupervised learning
  5. Reinforcement learning

با توجه به گستردگی مدل‌های یادگیری ماشین؛ آن دسته از الگوریتم‌های یادگیری ماشینی معمولاً کاربردی تر هستند که هم از نظر پیچیدگی محاسباتی و هم حافظه مورد نیاز در سیستم‌های موجود قابل پیاده‌سازی باشند و این معیارها علاوه بر معیارهای کارایی ارزیابی الگوریتم‌های یادگیری ماشین باید در کنار هم سنجیده شوند. بنابر این یک مطالعه بین کارایی و قدرت محاسباتی و حافظه مورد نیاز در انتخاب و طراحی الگوریتم‌های یادگیری ماشین باید صورت پذیرد.

الگوریتم‌های یادگیری ماشین شاخه‌ای از علم هوش مصنوعی می‌باشند و از اولین مدل‌های یادگیری که با "یادگیری نرون‌ها توسعه یافت" تحول چشم‌گیری یافته‌اند. Rosenblatt در سال ۱۹۵۷

امروزه مدل‌های یادگیری ماشین بسیار متنوع هستند و بر اساس روش‌های مختلف تقسیم‌بندی می‌شوند. از آن جمله می‌توان به مدل‌هایی که بر اساس یادگیری داده‌هایی که از قبل جمع‌آوری و ذخیره شده اند و مدل‌هایی که برای کار بر روی داده‌های جرجایی توسعه یافته‌اند، را نام برد. به‌طور کلی الگوریتم‌های یادگیری ماشینی به روش‌های پارامتریک و غیرپارامتریک تقسیم‌بندی می‌شوند. امروزه الگوریتم‌های یادگیری عمیق از حوزه‌های نوین می‌باشند که (مانند برخی از مدل‌های یادگیری مانند مدل‌های گرافیکی احتمالاتی) به صورت کلی به روش‌های مولد<sup>۱</sup>، تمايزی<sup>۲</sup> و ترکیبی<sup>۳</sup> تقسیم‌بندی می‌شوند. یادگیری عمیق تحولی عظیم در روش‌های یادگیری سنتی ایجاد کرده و از مهمترین خصوصیات آنها آموزش خودکار ویژگی‌ها را می‌توان نام برد که در بسیاری از حوزه‌ها از جمله در بینایی ماشین، کارایی الگوریتم‌های استخراج ویژگی‌های سنتی که به صورت دستی<sup>۴</sup> انجام می‌شد. را با شیوه‌های گوناگون یادگیری عمیق خودکار ویژگی‌ها (به عنوان مثال با استفاده از یک روش بدون ناظر موسوم به اتو انکودر) کم رنگ کرده است.

این گستردگی و تنوع روش‌ها و مدل‌های مختلف یادگیری ماشین باعث شده است که در توسعه بسیاری از الگوریتم‌های یادگیری ماشین؛ امروزه از نقطه تلاقی علوم مختلفی مانند تئوری احتمالات، هوش مصنوعی، بهینه‌سازی، کنترل، آمار، فیزیک، علوم اعصاب<sup>۵</sup>، علوم ریاضی و غیره ... استفاده شود.

امروزه الگوریتم‌ها و مدل‌های یادگیری ماشین به صورت مستقیم یا غیرمستقیم، در کاربردها و نرم‌افزارهای متنوعی که در سیستم‌های کامپیوتری همه منظوره و یا سیستم‌های کامپیوتری مختلف، شامل گوشی‌های موبایل، تبلت‌ها<sup>۶</sup> و دستگاه‌های مخصوص بازی‌ها و غیره، اجرا می‌شوند، به صورت گستره استفاده می‌شوند. در بسیاری از جستجوهای اینترنتی بر اساس رفتارهای قبلی کاربر، نتایج و توصیه‌ها در موضوعات مختلف بر مبنای الگوریتم‌های یادگیری ماشین تولید می‌شوند. نرم‌افزارها و وب سایت‌های فراوانی جهت پیش‌بینی بازار سهام، نرخ پیش‌بینی ارز و طلا، انتخاب و خرید کالاهای مختلف از وسایل حائزی گرفته تا خودرو و منزل و یا سرمایه‌گذاری

- 1. Generative
- 2. Discriminative
- 3. Hybrid
- 4. Hand craft
- 5. Neuroscience
- 6. Tablets

به صورت کامل یا جزئی، از الگوریتم‌های یادگیری ماشین بهره می‌برند. تشخیص و پیشگیری بیماری‌ها از روی آزمایشات متداول پزشکی، پردازش تصاویر پزشکی جهت تشخیص بیماری‌ها و یا تحلیل خصوصیات ژنتیکی و اطلاعات استخراج شده از ژن‌ها جهت پیش‌بینی یا تشخیص بیماری‌ها را می‌توان از جمله کاربردهای متداول که به نوعی از الگوریتم‌های یادگیری ماشین استفاده می‌کنند، نام برد.

در تشخیص گفتار، تبدیل گفتار به متن ترجمه ماشینی، پردازش‌های متنوع تصویر و ویدئو اعم از تشخیص چهره، تشخیص رفتار افراد، کاربردهای ترافیکی متنوع شامل اندازه‌گیری سرعت و تشخیص و جلوگیری از تصادفات و رفتار راننده و رانندگی بدون سرنشین، جستجوی فیلم‌ها و عکس‌های مورد نظر کاربر و غیره؛ خروجی نهایی را الگوریتم‌های یادگیری ماشین تولید می‌کنند. در حوزه‌های صنعتی و رباتیک مهمترین عامل مورد استفاده یادگیری ماشین است. طبیعتاً دامنه حوزه کاربردهای الگوریتم‌های یادگیری ماشین آن قدر وسیع هستند که در این کتاب مقدماتی، حتی عنوانی آن‌ها قابل طرح نیستند. در یک کلام دنیای امروز و آینده وابستگی کاملی به الگوریتم‌های یادگیری ماشین جهت هوشمندسازی تمامی عملیات و پروسس‌های مربوط به دنیای بشر دارد.

یکی از کاربردهای مهم امروزی در زمینه هوش مصنوعی داده کاوی است. همانگونه که ذکر شد امروزه با استفاده از فن‌آوری‌های پیچیده‌ای مانند کامپیوترها، ماهواره‌ها، و غیره، حجم بسیار بالایی از اطلاعات تولید و ذخیره می‌شود که دیگر با روش‌های سنتی و به صورت دستی توسط انسان این حجم انبوه داده‌ها امکان تحلیل و بررسی شدن را نداشته و از روش‌های مختلف داده کاوی برای این امر می‌توان استفاده کرد. در داده کاوی ابزارها و روش‌های مختلفی برای گردآوری و مدیریت داده‌های انبوه و سپس تجزیه و تحلیل آنها استفاده می‌شود. الگوریتم‌های یادگیری ماشین که غالباً عملیات پیش‌بینی فرایندها و یا رویدادها را در این علم و فن‌آوری به عهده دارند، نقش مهمی در پیاده‌سازی کاربردهای مختلف این حوزه دارد.

کتاب حاضر مقدمه‌ای بر الگوریتم‌های یادگیری ماشین بر مبنای زبان برنامه‌نویسی R می‌باشد. در این کتاب سعی شده است که از مبنای تئوری و نظری در حد نیاز بهره جسته و موضوعات به صورتی بیان شوند که کاربر بتواند با خواندن کتاب، دانش تئوری و تجربه عملی مناسب جهت پیاده‌سازی بسیاری از کاربردهای مورد نیاز صنایع مختلف را بدست آورد.

این کتاب شامل ۱۶ فصل و ۳ پیوست می‌باشد. پیوست ۱ شامل مقدماتی درباره بخش‌هایی از جبر خطی مربوط به پردازش‌های مورد نیاز ماتریس‌ها؛ که مورد نیاز خواننده جهت درک برخی از موضوعات است؛ می‌باشد. پیوست ۲ و ۳ شامل مباحث مقدماتی ولی کافی و نیز معرفی ابزارهایی جهت استفاده از زبان R و Python جهت پیاده‌سازی الگوریتم‌های یادگیری ماشین با استفاده از این زبان است. در پایان هر فصل سعی شده است که یک مثالی مناسب با استفاده از زبان R و Python در مورد موضوع مطرح شده در آن فصل با جرئیات پیاده‌سازی مطرح شود.

لازم به ذکر است که زبان‌های R و Python جزو زبان‌های متن باز هستند و به صورت گسترده امروزه در یادگیری ماشین استفاده می‌شود. علاوه بر استفاده از ابزارهای یاد شده در این کتاب، صدها بسته نرم‌افزاری در ارتباط با یادگیری ماشین توسط متخصصین این حوزه در سراسر جهان با استفاده از زبان‌های R و Python توسعه یافته که از طریق روش‌های ذکر شده در این کتاب توسط خوانندگان می‌توانند از اینترنت دریافت شده و مورد استفاده قرار گیرند. تعداد این بسته‌ها در کاربردهای مختلف و با مدل‌های کاراتر همواره با سرعت بالایی در حال افزایش است.

در نگارش این کتاب حدود ۵۰ درصد مطالب از ویدئوهای درسی آقای Andrew Ng از دانشگاه استنفورد می‌باشد، که از منابع مهم درسی در دانشگاه‌ها و مراکز تحقیقاتی دنیا است. این ویدئوها در ضمن جذاب بودن از دیدگاه طرح موضوعات بحث شده، نگرش نوین در بحث موضوعات یادگیری ماشین، و جامع بودن مباحث نظری و عملی، دارای مثال‌های اندکی با استفاده از زبان‌های متلب و اکتاو می‌باشند. تغییرات اساسی در بخش‌هایی از محتواهای این منبع، که در این کتاب به نگارش در آمده، اعمال گردیده است که به عنوان یکی از این تغییرات می‌توان به ارائه مثال‌های عملی برای هر مبحث با استفاده از دو زبان R و Python اشاره کرد.

در ضمن باقی مطالب کتاب شامل موضوعاتی است که به نظر نویسنده‌گان جای آنها در ویدئوهای آقای Andrew Ng خالی بوده و این مطالب از منابع جدید و متنوع نگارش شده‌اند. نکته دیگر این است که کلیه مثال‌های ذکر شده نسبتاً جدید بوده و توسط نویسنده‌گان اجرا و آزمایش شده‌اند و جامعیت موضوعات و مثال‌های ذکر شده خواننده را قادر می‌سازد که با فرآگیری آنها به عنوان یک متخصص یادگیری ماشین پژوهش‌های عملی مورد نیاز را به صورت کامل و صنعتی طراحی و پیاده‌سازی نماید.



١

فصل

## رگرسیون

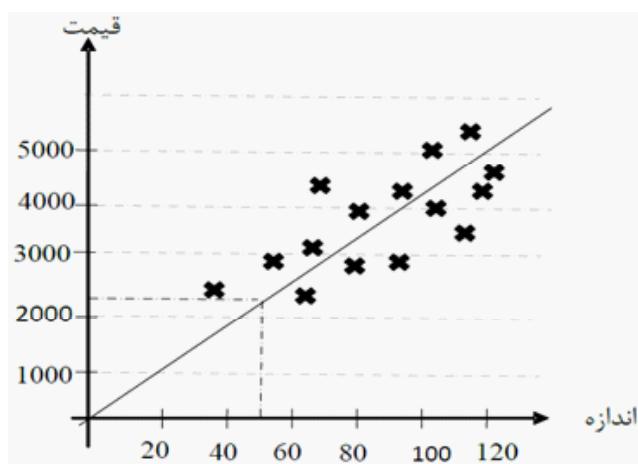
## ۱-۱- رگرسیون

در مدل‌های آماری، تحلیل رگرسیون<sup>۱</sup> یک فرایند آماری برای مدلسازی و تخمین روابط بین متغیرها است. به بیانی دیگر شامل تجزیه و تحلیل داده‌های درگیر دو متغیر یا بیشتر، برای کشف ماهیت رابطه میان آنها و انجام عمل تخمین یا پیش‌بینی است که در شاخه‌های مختلفی از علوم مانند فیزیک، شیمی، اقتصاد، مدیریت و بسیاری دیگر کاربرد زیادی دارد و می‌توان گفت که رگرسیون، پرکاربردترین روش مورد استفاده در میان سایر روش‌های آماری است.

## ۱-۲- رگرسیون خطی تک متغیر<sup>۲</sup>

در رگرسیون خطی<sup>۳</sup> تلاش برای پیش‌بینی متغیر  $y$ ، بر اساس متغیر  $x$  است که در تعریف به متغیر  $x$  متغیر مستقل و به متغیر  $y$  متغیر وابسته می‌گوییم. به عبارتی دیگر می‌توان گفت در رگرسیون خطی ساده یا تک متغیره، تلاش برای پیش‌بینی یک متغیر وابسته براساس مقدار یک متغیر مستقل می‌باشد.

برای درک بهتر مثالی را از کاربرد رگرسیون خطی ارائه می‌دهیم که در آن هدف تخمین قیمت یک خانه یا مسکن است بر این اساس متغیرها یا ویژگی‌های مورد استفاده در این مثال اندازه (مساحت) و قیمت خانه می‌باشد که بر اساس تعریف ارائه شده برای رگرسیون خطی تک متغیره، در این مسئله متغیر وابسته می‌تواند قیمت خانه و متغیر مستقل مساحت خانه باشد. نمودار شکل (۱-۱) نشان‌دهنده نتایج محاسبه و تخمین قیمت مسکن است که در آن مقادیر مربوط به اندازه و قیمت خانه به صورت غیرواقعی و فرضی می‌باشند. همانگونه که بیان شد برای حل مسئله‌ای با رگرسیون ممکن است که مجموعه داده<sup>۴</sup> گفته می‌شود.



شکل (۱-۱): تخمین قیمت خانه بر اساس اندازه آن با استفاده از رگرسیون خطی تک متغیره.

1. Regression
2. Simple regression
3. Dataset

توجه به این نکته ضروری است که برای تحلیل هر مسئله‌ای نیازمند بررسی ویژگی‌های مختلف آن مسئله می‌باشیم که برای مسئله قیمت خانه این ویژگی‌ها می‌توانند اندازه خانه، سال ساخت خانه، تعداد اتاق‌های آن و ویژگی‌های دیگر باشند که شخص حل کننده مسئله باید بتواند ویژگی‌های مهم را شناسایی و در تحلیل خود مورد استفاده قرار دهد و به دلیل نوع روش حل ما که رگرسیون خطی تک متغیره است ماتنها از ویژگی اندازه خانه، به عنوان متغیر مستقل استفاده می‌کنیم تا قیمت خانه را تخمین بزنیم.

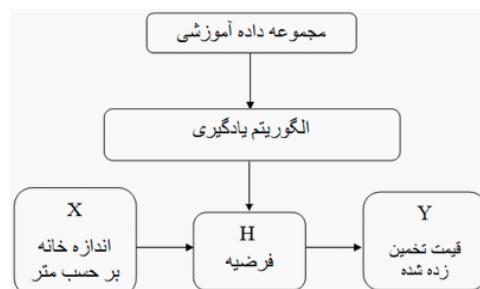
رگرسیون تک متغیره بیشتر از آن که یک روش مؤثر ارائه کند یک مدل ساده برای درک راحت‌تر رگرسیون است زیرا مسائل دنیای واقعی را معمولاً با مد نظر قرار دادن یک ویژگی نمی‌توان به صورت درست تحلیل کرد.

از دیدگاه یادگیری ماشین الگوریتم رگرسیون خطی یک مدل پیش‌بینی کننده است که به مجموعه الگوریتم‌های با نظارت تعلق دارد زیرا در آن، داده‌ها دارای برچسب می‌باشند. به عبارتی برای مثال مسئله خانه، به ازای هر کدام از مقادیر موجود برای ویژگی مساحت خانه در مجموعه داده‌ها، قیمت متناظر با آن نیز وجود دارد و چیستی هر کدام از مقادیر مشخص است. همچنین الگوریتم رگرسیون یک الگوریتم یادگیرنده است یعنی ابتدا ما الگوریتم را براساس داده‌های آموزشی، آموزش می‌دهیم تا مدل مناسب را برای حل مسئله به دست آورد و سپس با استفاده از داده‌های آزمون آن را مورد آزمایش قرار می‌دهیم تا مشخص شود که آیا الگوریتم ما قادر به عملکرد صحیح در شرایط واقعی خواهد بود یا خیر.

### ۱-۳- تقسیم داده‌ها به داده‌های آموزشی و آزمون

در هر الگوریتمی که یادگیرنده باشد روش عمومی این است که مجموعه داده‌ها را به دو بخش داده‌های آموزشی و داده‌های آزمون تقسیم کنیم. درصد تقسیم داده‌ها توسط طراح سیستم تعیین می‌شود، ولی معمولاً ۷۰ درصد برای داده‌های آموزشی و ۳۰ درصد برای داده‌های آزمون در نظر گرفته می‌شود.

بر اساس توضیحات بیان شده برای مسئله خانه ابتدا باید تلاش به ایجاد مدلی مناسب برای الگوریتم خود کنیم. در مرحله اول باید یک مدل فرضی را مقداردهی اولیه کنیم تا در جریان اجرای الگوریتم مدل اولیه ما به یک حالت بهینه برسد. فرایند کلی کار را می‌توانید در شکل (۲-۱) مشاهده نمایید.



شکل (۲-۱): فرایند کلی اجرای الگوریتم رگرسیون خطی.

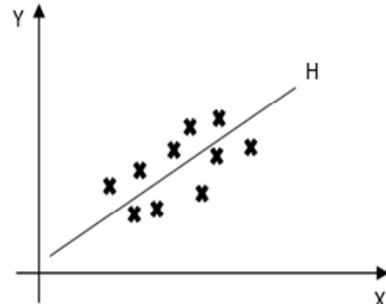
همانگونه که در شکل (۳-۱) مشاهده می‌نمایید یک الگوریتم یادگیری را مدل می‌کنیم که ورودی آن مجموعه داده‌های آموزشی و خروجی نهایی قیمت تخمین زده شده  $y$  است و همچنین  $x$  همان داده‌های مربوط به ویژگی مساحت خانه و  $H$  همان فرضیه‌ای (تابع پیش‌بینی کننده قیمت خانه در مثال فوق) است که ما تمامی تلاش‌مان را برای قرار دادن آن خط (بحث ما در این فصل رگرسیون خطی است بنابراین فرضیه  $H$  یک خط می‌باشد) در مناسب‌ترین حالت ممکن انجام خواهیم داد.

به عبارتی اگر مجموعه داده‌های ما به شکل جدول (۱-۱) باشد ما می‌توانیم آن را به صورت یک محور مختصات مانند شکل (۳-۱) نشان دهیم که محور عمودی و محور افقی درون خود، داده‌های آموزشی را جای داده است. هدف اصلی، ترسیم خط فرضیه در بهترین حالت ممکن در میان داده‌های آموزشی می‌باشد.

به صورت کلی هرچه خط فرضیه بتواند با داده‌های آموزشی مجاورت بهتری داشته باشد دقت الگوریتم بالاتر و در حالت بر عکس هرچه خط فرضیه در راستایی به دور از محل تجمع داده‌ها باشد کارایی الگوریتم برای پیش‌بینی ضعیفتر خواهد بود.

مساحت خانه (x)	قیمت (y)
۲۱۰۴	۴۶۰
۱۴۱۶	۲۳۲
۱۵۳۴	۳۱۵
۸۵۲	۱۷۸
:	:

جدول (۱-۱): تعدادی از مجموعه داده‌های مربوط به خانه.



شکل (۳-۱): محور مختصات مربوط به خانه با خط فرضیه.

به بیان ریاضی، اگر تعداد مثال‌های آموزشی را  $m$  و متغیرهای ورودی هر نمونه<sup>۱</sup> آموزشی را  $n$  (در این مثال  $n = 1$ ) که همان مساحت خانه است) و متغیرهای خروجی را با  $y$  نشان دهیم. با توجه به جدول (۱-۱) که مربوط به داده‌های آموزشی خانه‌های موجود در یک منطقه است  $x^{(1)}$  مربوط به  $1$  امین  $x$  یا مساحت، و  $y^{(1)}$  مربوط به  $1$  امین مقدار برای قیمت خانه است. بر اساس جدول (۱-۱) نمونه‌ای از مقادیر مربوط به  $x$  و  $y$  بصورت  $= 460$  و  $= 2104$  می‌باشد.

با توجه به مسئله خانه می‌توان گفت که هدف الگوریتم رگرسیون خطی تک متغیره، گرفتن مقدار مربوط به مساحت خانه  $(x)$  در ورودی و تخمین قیمت و ارسال آن به خروجی  $(y)$  است که توسط خط فرضیه  $h_{\theta}(x)$  انجام می‌شود.

رابطه ریاضی مربوط به خط فرضیه به صورت زیر می‌باشد:

$$h_{\theta}(x) = \theta_0 + \theta_1 x \quad \text{رابطه (۱)}$$

در رابطه (۱) خط فرضیه یا مدل ما  $(x) h_{\theta}$  و  $\theta_0$  و  $\theta_1$  مربوط به پارامترهای مدل می‌باشند و  $x$  نیز همان متغیر مستقل است. همانگونه که در شکل (۳-۱) مشخص است خط فرضیه یک خط راست است که دلیلش وجود تنها

یک متغیر در مدل مربوطه است. اگر تعداد متغیرهای مستقل زیاد بود مدل ما از پیچیدگی بالایی برخوردار بود و این خط دیگر به صورت یک خط راست ترسیم نمی‌شد. یکی از نکات بسیار مهم، اختصاص مقادیر مناسب به پارامترهای  $\theta_0$  و  $\theta_1$  است که در اصل همین مقادیر مشخص‌کننده محل قرارگیری خط فرضیه خواهد بود.

## I-۴ - تابع هزینه<sup>۱</sup>

در علم آمار تابع هزینه تابعی است که مقدار خطا را در یک پیشامد نشان می‌دهد و در یک مسئله مبتنی بر یادگیری، برای به دست آوردن نتیجه مطلوب باید تابع هزینه به کمینه<sup>۲</sup> خود برسد.

### محاسبه تابع هزینه

رابطه تابع هزینه به صورت زیر است که به تابع مربعات خطا نیز معروف می‌باشد.

$$J(\theta_0, \theta_1) = \frac{1}{2} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \quad \text{رابطه (۲)}$$

در رابطه (۲) تعداد مثال‌های آموزشی را با  $m$  نشان داده‌ایم،  $h_\theta(x^{(i)})$  خط فرضیه ما و  $y^{(i)}$  خروجی یا همان قیمت خانه می‌باشد.

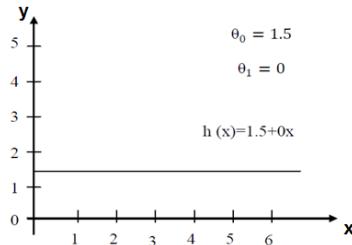
همانطور که گفته شد در مرحله اول الگوریتم،  $\theta$ ها را مقداردهی اولیه می‌کنیم. با اختصاص مقادیر مختلف به  $\theta$ ها خواهیم دید که خط فرضیه، وضعیت متفاوتی را در نمودار به خود می‌گیرد. به صورت کلی می‌توان گفت که الگوریتم رگرسیون در مرحله آموزش پس از آن که مقادیر اولیه به پارامترها اختصاص داده شد با استفاده از رابطه (۲) مقدار تابع هزینه را محاسبه می‌کند و تا زمانی که تابع هزینه به کمینه خود نرسیده است مجدداً تغییر مقادیر پارامترها را انجام می‌دهد تا در نهایت بتواند با اختصاص مناسب‌ترین مقدار به پارامترها، خط فرضیه را در بهترین حالت خود قرار دهد. در حل مسئله قیمت خانه نیز ما برای اینکه بتوانیم مقادیر مناسب را برای  $\theta$ ها اختصاص دهیم باید بتوانیم مقدار تابع هزینه را به کمینه برسانیم. در این حالت پارامترهای  $\theta$  مناسب‌ترین مقادیر را برای ترسیم یک خط فرضیه بهینه برای تخمین قیمت خانه‌ها به خود خواهند گرفت.

با ملاحظه چند مثال در ادامه مطالب خواهید دید که با اختصاص مقادیر مختلف به پارامترها، چه مقداری برای تابع هزینه به دست می‌آید. به منظور درک بهتر، نمودار خط فرضیه و تابع هزینه مربوط به آن مقادیر را نیز ترسیم نموده‌ایم.

نکته: سه مثال اول بدون توضیح است و صرفاً جهت مشاهده چگونگی ترسیم خط فرضی براساس مقادیر متفاوت پارامترها است.

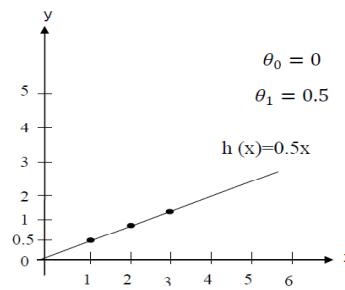
1. Cost function  
2. Minimum

مثال ۱) در این مثال  $\theta_1 = 0$  و  $\theta_0 = 1.5$  می‌باشد.



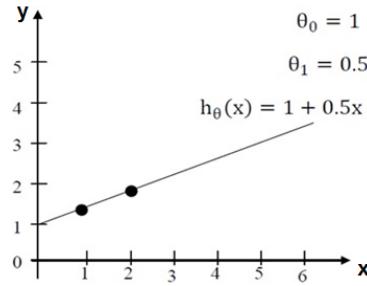
شکل (۴-۴): ترسیم خط فرضی به صورت افقی وقتی که  $\theta_0$  برابر ۱.۵ و  $\theta_1$  برابر صفر است.

مثال ۲) در این مثال  $\theta_1 = 0.5$  و  $\theta_0 = 0$  می‌باشد.



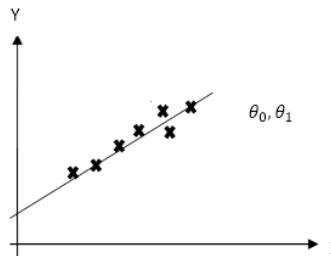
شکل (۴-۵): ترسیم خط فرضیه وقتی که  $\theta_0$  برابر صفر و  $\theta_1$  برابر ۰.۵ است.

مثال ۳) در این مثال  $\theta_1 = 0.5$  و  $\theta_0 = 1$  می‌باشد.



شکل (۴-۶): ترسیم خط فرضیه وقتی که  $\theta_0$  برابر ۱ و  $\theta_1$  برابر ۰.۵ است.

حال اگر ما براساس مسئله موجود مجموعه داده‌هایی را داشته باشیم که مانند شکل (۷-۲) باشد همانگونه که مشخص است خط فرضیه در یک شرایط مناسبی قرار گرفته است که به دلیل مقدار مناسب پارامترهای  $\theta$  است و اگر خط فرضیه در هر حالت دیگری قرار می‌گرفت دیگر تخمین قیمت خانه با دقت مناسبی اتفاق نمی‌افتد.



شکل(۷-۱): قرار گیری مناسب خط فرضیه با توجه به محل داده‌ها.

### ۱-۴-۱- تشریح روش محاسبه تابع هزینه

در ادامه با یک مثال به صورت مرحله به مرحله نحوه عملکرد تابع هزینه و رابطه آن با خط فرضیه را تشریح خواهیم کرد.

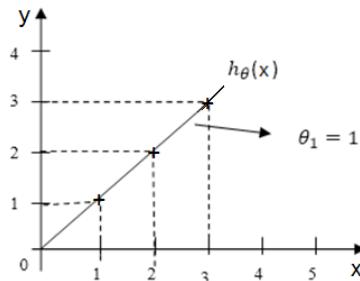
بر اساس شکل (۸-۱) می‌خواهیم محاسبه قیمت خانه را با استفاده از سه داده آموزشی که در محور مختصات در نقطه‌های (۱,۱) و (۲,۲) و (۳,۳) قرار دارند ادامه دهیم و برای سادگی، پارامتر  $\theta_0$  را مساوی صفر قرار می‌دهیم تا خط فرضیه از نقطه صفر مانند شکل (۹-۱) بگذرد و در محاسبات نیز تنها پارامتر  $\theta_1$  را مورد توجه قرار دهیم.

بر این اساس رابطه خط فرضیه به صورت رابطه (۳) و تابع هزینه به صورت رابطه (۴) خواهد بود.

$$h_{\theta}(x) = \theta_1 x \quad (3)$$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \quad (4)$$

همانگونه که مشاهده می‌کنید در رابطه (۴) تنها  $\theta_1$  محاسبه می‌شود و به ازای  $\theta_1 = 1$  فرضیه ما به صورت شکل (۸-۱) خواهد بود.

شکل (۸-۱): خط فرضیه در صورتی که  $\theta_0$  برابر با صفر و  $\theta_1 = 1$  باشد.

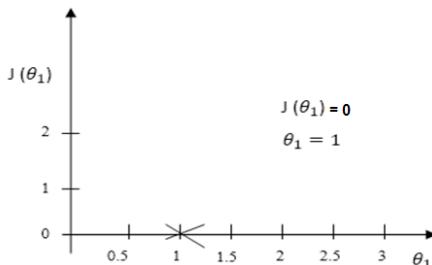
محاسبه تابع هزینه براساس شکل (۸-۱) که در آن تنها پارامتر  $\theta_1$  مورد توجه می‌باشد.

مثال) محاسبه  $J(\theta)$  با توجه به داده‌های آموزشی (۱,۱)، (۲,۲) و (۳,۳) و  $\theta_1 = 1$  به صورت زیر می‌باشد.

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m} \sum_{i=1}^m (\theta_1 x^{(i)} - y^{(i)})^2 = \frac{1}{2m} (0^2 + 0^2 + 0^2) = 0$$

همانگونه که مشاهده می‌نمایید در محاسبات بالا بجای  $h_\theta(x^{(i)})$  می‌توانیم  $\theta_1 x^{(i)}$  را بنویسیم و همچنین نتیجه محاسبات فوق به صورت خلاصه به شرح زیر می‌باشد.

بر اساس شکل (۸-۱) سه داده آموزشی وجود دارد که محل قرارگیری آن در نقطه مربوط به  $x$  و  $y$ ها به صورت (۱)، (۲) و (۳) می‌باشد. اگر در رابطه تابع هزینه مقدار (۱) را جایگذاری کنیم نتیجه  $0^2$  و برای مقدار (۲) و (۳) نتیجه  $0^2$  و همچنین برای مقدار (۳) نتیجه  $0^2$  خواهیم داشت. چون در رابطه از  $\sum$  استفاده شده پس  $(0^2 + 0^2 + 0^2)$  را خواهیم داشت و در نهایت  $J(\theta_1) = 0$  خواهد بود. به عبارتی به ازای مقادیر مربوط به هر داده آموزشی مقدار  $h_\theta(x^{(i)})$  به دست می‌آید و در نهایت نتیجه محاسبه شده برای هر داده را با نتیجه محاسبه بقیه داده‌ها جمع می‌کند. با توجه به  $J(\theta_1) = 0$  پس خطی تابع هزینه به صفر رسیده و خط فرضیه ما همانگونه که در شکل (۸-۱) مشخص است در دقیق‌ترین حالت بر روی داده‌های آموزشی منطبق است و تابع  $J(\theta_1)$  نیز به صورت شکل (۹-۱) بر روی نقطه (۱، ۰) خواهد بود.



شکل (۹-۱): نمودار ترسیم تابع  $J(\theta_1)$

مثال) محاسبه  $J(\theta)$  با توجه به داده‌های آموزشی (۱)، (۲) و (۳) و  $\theta_1 = 0.5$  به صورت زیر می‌باشد.

حال باید تابع هزینه را دوباره با مقدار  $0.5$  برای پارامتر  $\theta_1$  محاسبه کنیم که در اینجا نیز خط فرضیه به صورت  $h_\theta(x) = \theta_1 x$  و داده‌های آموزشی نیز مانند مسئله قبل خواهد بود که در شکل (۸-۱) نیز مقادیر آنها مشخص است و همانطور که در شکل (۱۰-۱) مشاهده می‌کنید همان فاصله داده‌های آموزشی تا خط فرضیه است یا به عبارتی مقدار فاصله نسبی نمونه‌های آموزشی  $y$ ، با مقادیر تخمین زده شده  $h_\theta(x^{(i)})$  را با  $d$  نشان داده‌ایم.

$$J(0.5) = \frac{1}{2m} [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2] = \frac{1}{2 \times 3} (3.5) = \frac{3.5}{6} \approx 0.58$$

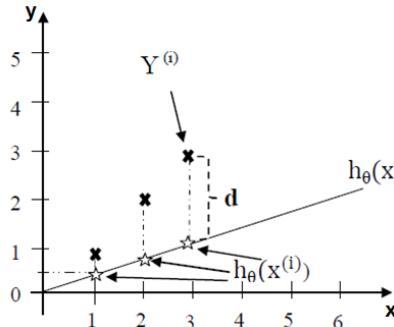
در محاسبات بالا  $(0.5 - 1)^2$  اشاره به

$$(\theta_1(x^{(2)}) - y^{(2)})^2 \quad \text{اشاره به } (1-2)^2$$

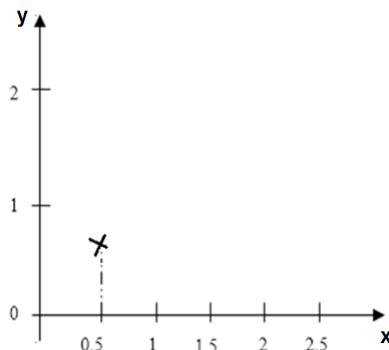
$$\text{و همچنین } (\theta_1(x^{(3)}) - y^{(3)})^2 \quad \text{اشاره به } (1.5-3)^2 \quad \text{دارد.}$$

و برای رابطه  $\frac{1}{2m}$  به دلیل آنکه  $m$  همان تعداد نمونه آموزشی است رابطه  $\frac{1}{2\times 3}$  ایجاد گردیده است.

طبق محاسبات رابطه  $J_{\theta_1} \approx 0.58$  صادق است که نمودار شکل (۱۱-۱) آن را نشان می‌دهد.



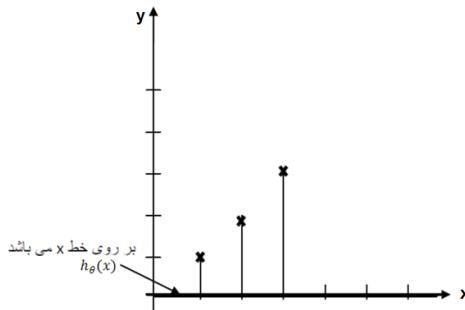
شکل (۱۰-۱): وضعیت قرارگیری خط فرضیه که  $h_{\theta_1}(x)$



شکل (۱۱-۱): نمودار  $J_{\theta_1}$  وقتی که  $\theta_1 = 0.5$  می‌باشد.

مثال) محاسبه  $J_{\theta_1}$  با توجه به داده‌های آموزشی (۱)، (۲) و (۳) و  $\theta_1 = 0$  به صورت زیر می‌باشد.

و همچنین اگر  $\theta_1 = 0$  باشد نمودار شکل (۱۲-۱) را برای  $h_{\theta_1}(x)$  خواهیم داشت که در آن خط فرضیه بر روی محور  $x$  منطبق است و نقطه مختصات برای داده‌های آموزشی به مانند مثال‌های قبلی می‌باشد.

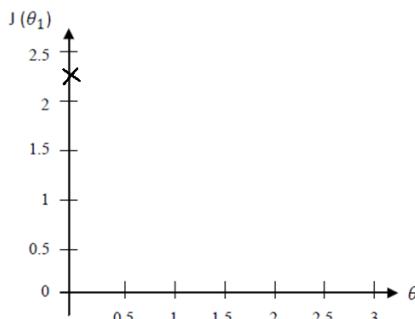


شکل (۱۲-۱): نمودار  $h_\theta(x^{(i)})$  که بر روی محور X منطبق است.

اگر خط فرضیه بر روی محور X منطبق باشد در این حالت  $h_\theta(x)$  ها صفر می شوند و برای  $J(0)$  محاسبات زیر را خواهیم داشت. و نمودار مربوط به تابع هزینه به صورت شکل (۱۳-۱) خواهد بود.

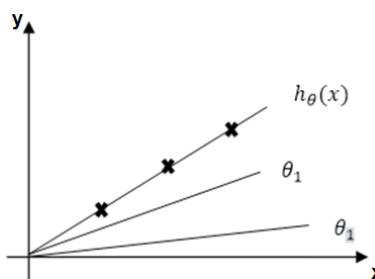
$$J(0) = \frac{1}{2m} (1^2 + 2^2 + 3^2) = \frac{1}{2 \times 3} (1^2 + 2^2 + 3^2) = \frac{1}{6} (14) \approx 2.3$$

$$J(0) \approx 2.3$$



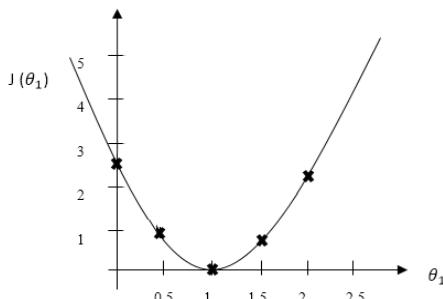
شکل (۱۳-۱): نمودار تابع هزینه مربوط وقتی که  $h_\theta(x)$  بر روی محور X منطبق است.

اگر در زیر به شکل (۱۴-۱) توجه نمایید مشاهده می شود که به ازای مقادیر مختلف  $\theta_1$  که در اصل مقدار فاصله نسبی آن از  $h_\theta(x)$  می باشد تابع هزینه مقادیر متفاوتی را همانگونه که در شکل (۱۵-۱) مشاهده می نمایید نشان می دهد.



شکل (۱۴-۱): خط فرضیه های مختلفی بر اساس مقادیر مختلف  $\theta_1$ .

به ازای مقادیر مختلف  $\theta_1$  مانند -0.5 و 0 و 0.5 و 1 به صورت تقریبی شکل (۱۵-۱) را خواهیم داشت.



شکل (۱۵-۱): نقاط رسم شده برای تابع هزینه بر اساس مقادیر مختلف  $\theta_1$ .

با توجه به اینکه با در نظر گرفتن مقادیر مختلف برای  $\theta_1$  در مثال‌هایی که حل کردیم نتایج متفاوتی برای تابع هزینه به دست آوردیم این موضوع اثباتی برای اهمیت اختصاص مقدار مناسب به  $\theta_1$  است که در مثال‌های ما تنها برای  $\theta_1 = 1$  مقدار تابع هزینه به صفر رسید و دیدیم که خط فرضیه بر روی نمونه‌های آموزشی منطبق بود که در اصل به نقطه کمینه تابع هزینه نیز معروف است.

## ۱-۴-۲- نگاهی عمیق‌تر به تابع هزینه

در این بخش سعی خواهیم کرد، نگاه عمیق‌تری به تابع هزینه داشته باشیم. برخلاف قسمت اول که پارامتر  $\theta_0$  را از محاسبات حذف کردیم در اینجا برای محاسبه تابع هزینه از هر دو پارامتر  $\theta_0$  و  $\theta_1$  استفاده خواهیم کرد.

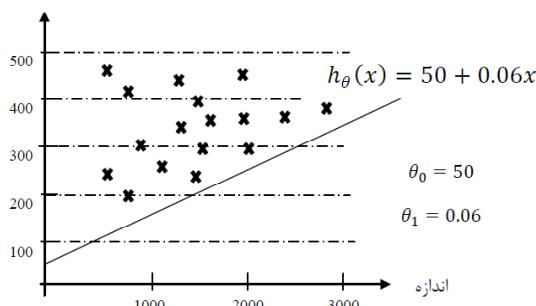
با همان مثال خانه، کار را ادامه می‌دهیم. همانگونه که در شکل (۱۶-۱) مشاهده می‌کنید داده‌های مربوط به خانه با تعداد بیشتری در نمودار مشاهده می‌شوند که خط فرضیه کمی پایین‌تر از آنها ترسیم شده و خط فرضیه به صورت زیر می‌باشد.

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

$$h_{\theta}(x) = 50 + 0.06x$$

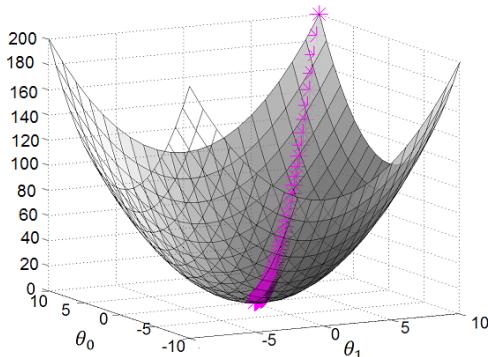
شکل (۱۶-۱): ترسیم خط فرضیه در محل نامناسب.  
شکل (۱۶-۱): ترسیم خط فرضیه در محل نامناسب.

فیمت



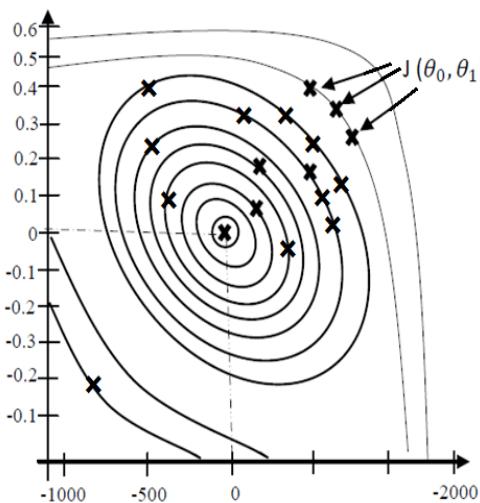
شکل (۱۶-۱): ترسیم خط فرضیه در محل نامناسب.

در مباحث ابتدایی تابع هزینه  $\theta_0$  را برای راحتی کار صفر قرار دادیم و مقادیر  $\theta_1$  مورد بررسی قرار گرفت و برای مقادیر مختلف  $\theta_1$  نمودار دو بعدی برای تابع هزینه، مانند شکل (۱۵-۱) ترسیم گردید. ولی اگر مقادیر هر دو پارامتر  $\theta_0$  و  $\theta_1$  را مورد محاسبه قرار دهیم همانطور که در شکل (۱۷-۱) مشاهده می‌نمایید نمودار آن شکلی محدب، مانند یک کاسه را داشته و به صورت سه بعدی خواهد بود.



شکل (۱۷-۱): نمودار تابع هزینه با استفاده از دو پارامتر  $\theta_0$  و  $\theta_1$ .

همانطور که در شکل (۱۷-۱) مشاهده می‌شود  $\theta_0$  و  $\theta_1$  دو بعد این نمودار و تابع هزینه بعد سوم و یا ارتفاع این نمودار می‌باشد. شکل (۱۸-۱) از نمای بالا نمودار تابع هزینه شکل (۱۷-۱) را به نمایش می‌گذارد که مقادیر تابع هزینه برای نقاط مختلف روی یک مدار برابر می‌باشد و نقطه موجود در درونی ترین مدار همان کوچک‌ترین مقدار برای تابع هزینه را نشان می‌دهد. همانگونه که پیش از این گفتم ما آن را کوچک‌ترین نقطه برای تابع هزینه می‌نامیم زیرا وقتی تابع هزینه در آن نقطه قرار گیرد برای خط فرضیه بهترین حالت را برای تخمین می‌توانیم مصور باشیم.



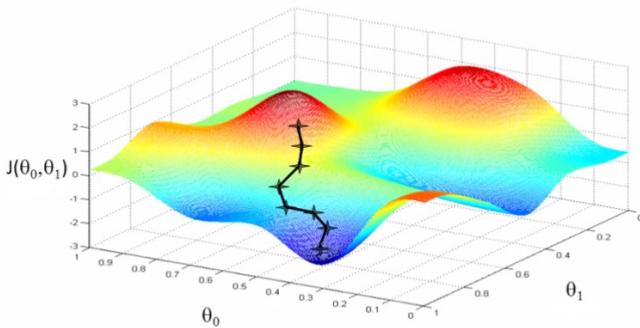
شکل (۱۸-۱): نمودار تابع هزینه از نمای بالا.

## ۱-۵- گرادیان نزولی<sup>۱</sup>

الگوریتم گرادیان نزولی در فضای وزن‌ها به دنبال برداری می‌گردد که خطرا به نقطه کمینه برساند. این الگوریتم از یک مقدار اولیه دلخواه برای بردار وزن شروع کرده و در هر مرحله وزن‌ها را طوری تغییر می‌دهد که در جهت شبکه کاهشی منحنی حرکت کند و خطرا مرتباً کاهش دهد تا به مقدار کمینه برسد.

در این بخش نحوه عملکرد گرادیان نزولی و همچنین نحوه استفاده روش گرادیان نزولی برای کاهشتابع هزینه مورد بررسی قرار گرفته است.

برای تابع هزینه در مسائل واقعی ممکن است تعداد پارامترها زیاد باشد که در این حالت تابع هزینه به شکل  $J(\theta_0, \theta_1, \dots, \theta_n)$  خواهد بود. ما در این مسئله نیز برای راحتی درک آن فرض می‌کنیم، پارامترهای ما  $\theta_1$  و  $\theta_0$  می‌باشند. برای به کمینه رساندن تابع هزینه ابتدا برای  $\theta_1$  و  $\theta_0$  یک مقدار پیش‌فرض انتخاب می‌کنیم و مقدار آنها را تا زمانی که تابع هزینه به کوچک‌ترین حد بهینه خود نرسیده است در جهت کاهش شبکه تغییر می‌دهیم. به شکل (۱۹-۱) که گرادیان نزولی را به تصویر می‌کشد توجه نمایید.



شکل (۱۹-۱): گرادیان نزولی و مینیمم سراسری.

برای پیدا کردن نقطه کمینه از طریق گرادیان نزولی مقداردهی اولیه  $\theta$ ‌ها بسیار مهم است و باید مراقب انتخاب اولیه باشیم زیرا ممکن است اگر نقطه شروع را درست انتخاب نکنیم شبکه به سمت کمینه‌ای حرکت کند که به آن کمینه محلی<sup>۲</sup> می‌گوییم، ولی اگر انتخاب مقدار اولیه برای پارامترها به درستی انجام شود شناس الگوریتم برای رسیدن به کمینه سراسری بالا می‌رود.

چنانچه در شکل (۱۹-۱) مشاهده می‌کنید کمینه سراسری به کمترین مقدار در فضای سه بعدی موجود از نظر ارتفاع گفته می‌شود. همان‌طور که می‌بینید مقداردهی اولیه که در قسمت مرتفع است به گونه‌ای انجام شده که شبکه در زاویه مناسبی به سمت پست‌ترین قسمت نمودار حرکت کرده است که اگر این مقداردهی اولیه نامناسب بود زاویه نقاط در حال حرکت به سمت پایین می‌توانست در مسیر نادرست قرار گیرد و به یک نقطه کمینه دیگر برسد که پایین‌ترین حد ممکن در کل نمودار نیست.

## ۱-۵-۱- رابطه گرادیان نزولی

در صورت اینکه تنها دو پارامتر را در نظر بگیریم رابطه گرادیان نزولی به صورت زیر خواهد بود.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{رابطه (۵)}$$

در رابطه (۵) هدف محاسبه مقادیر مختلف پارامترها است و  $(\theta_j)$  اشاره به پارامتر  $\theta_j$  دارد و علامت (=) گویای این موضوع است که نتیجه سمت راست تساوی با مقدار موجود در  $\theta_j$  جمع می‌شود. همچنین  $\alpha$  نرخ یادگیری<sup>۱</sup> می‌باشد. در بخش بعد در مورد اینکه مقدار مناسب برای  $\alpha$  چه باید باشد، بحث خواهیم کرد.

$$\begin{aligned} \text{برای قسمت } (\theta_1) \text{ فرض کنید برای سادگی تعداد پارامترها را فقط یکی در نظر می‌گیریم که به شکل} \\ \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \text{ شود در این حالت رابطه (۶) را خواهیم داشت.} \\ \frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \quad \text{رابطه (۶)} \end{aligned}$$

## ۱-۵-۲- الگوریتم گرادیان نزولی

به صورت کلی الگوریتم گرادیان نزولی برای یافتن پارامترهای مناسب به صورت زیر می‌باشد.

اجرای الگوریتم تا زمانی که به کمینه برسد {

$$\begin{aligned} (\text{j}=1 \text{ و } \text{j}=0 \text{ به روزرسانی همزمان}) \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\ \} \end{aligned}$$

الگوریتم گرادیان نزولی برای به روزرسانی  $\theta_j$  است که در اصل وظیفه‌اش به دست آوردن مقادیر  $\theta_1$  و  $\theta_0$  و  $\theta_j$  رساندن گرادیان به نقطه کمینه است. در این الگوریتم تمامی پارامترها باید به صورت همزمان محاسبه شوند و مقادیر پارامترها در هم اثر نداشته باشند. مثلاً پارامتر  $\theta_0$  در نتیجه محاسبه پارامتر  $\theta_1$  دخالتی نداشته باشد و تنها مقادیر به دست آمده از محاسبات جدید برای هر پارامتر، تنها بر خود همان پارامتر اثر گذارد. به عبارتی دیگر باید  $\theta_1$  و  $\theta_0$  به صورت همزمان به روزرسانی شوند که مراحل اجرای الگوریتم برای محاسبه پارامترها در حالت همزمان به صورت زیر می‌باشد.

$$\begin{cases} \text{Temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ \text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \end{cases} \quad \text{مرحله اول:}$$

- 
1. Learning rate
  2. Simultaneous update

$$\begin{cases} \theta_0 := temp0 \\ \theta_1 := temp1 \end{cases}$$

مرحله دوم:

براساس مراحل بالا در هر دور هر کدام از  $\theta$  ها با فرمول مطرح شده به صورت مجزا محاسبه شده و مقادیر پارامترهای متفاوت به صورت هم زمان به روز می‌شود.

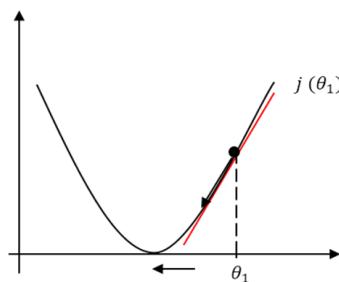
### ۱-۵-۳- نرخ یادگیری در گرادیان نزولی

نرخ یادگیری که در رابطه (۵) با علامت  $\alpha$  نشان داده شد تعیین کننده اندازه گام‌های گرادیان نزولی است. اگر مقدار  $\alpha$  بزرگ باشد گرادیان نزولی با گام‌های بزرگ حرکت خواهد کرد و اگر  $\alpha$  کوچک باشد اندازه هر گام گرادیان نزولی کوچک خواهد بود.

در ادامه به تشریح کامل  $\alpha$  و  $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$  و دلیل وجود این دو کنار هم خواهیم پرداخت. با یک مثال توضیحات را ادامه می‌دهیم. اگر مانند شکل (۲۰-۱) تابع  $j(\theta)$  را داشته باشیم و  $\theta_1$  نیز همانگونه که مشاهده می‌کنید در قسمت راست نقطه کمینه قرار گرفته بر این اساس خط مماس بر تابع  $j(\theta)$  با یک شیب مثبت خواهد بود که با رابطه  $\frac{\partial}{\partial \theta_j} J(\theta_1)$  مشخص می‌شود پس رابطه  $\frac{\partial}{\partial \theta_j} J(\theta_1)$  مقداری مثبت بوده و در رابطه محاسبه  $\theta_1$  همانگونه که در زیر مشاهده می‌کنید  $\alpha$  در یک عدد مثبت ضرب می‌شود و منفی می‌ماند و  $\theta_1$  منهای یک عدد مثبت شده و نتیجه نهایی  $\theta_1$  کمی کوچک‌تر از قبل خواهد بود.

$$\theta_1 := \theta_1 - \alpha \times (\text{یک عدد مثبت})$$

که مشخص می‌کند  $\theta_1$  در محور  $x$  به سمت صفر میل می‌کند پس اگر به سمت صفر حرکت کند کمی به نقطه کمینه تابع  $j(\theta)$  نزدیک‌تر خواهد شد.

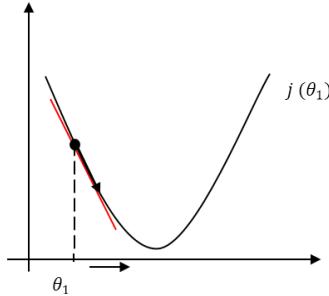


شکل (۲۰-۱): پارامتر  $\theta_1$  در سمت راست تابع هزینه.

و اگر مانند شکل (۲۱-۱) پارامتر  $\theta_1$  در قسمت چپ نقطه کمینه بر روی محور  $x$  باشد خط مماس بر تابع شیب منفی خواهد داشت که از رابطه  $\frac{\partial}{\partial \theta_j} J(\theta_1)$  به دست می‌آید. پس بر این اساس مانند رابطه زیر  $\alpha$  در یک عدد منفی ضرب شده و تبدیل به یک عدد مثبت خواهد شد و بر این اساس  $\theta_1$  با یک عدد مثبت جمع شده و مقدارش بیشتر می‌شود.

$$\theta_1 := \theta_1 - \alpha (\text{یک عدد منفی})$$

بیشتر شدن  $\theta_1$  یعنی حرکت آن مقداری به سمت راست در محور  $x$  و در نتیجه نزدیکتر شدن  $\theta_1$  به نقطه کمینه تابع.



شکل (۲۱-۱): پارامتر  $\theta_1$  در سمت چپ تابع هزینه.

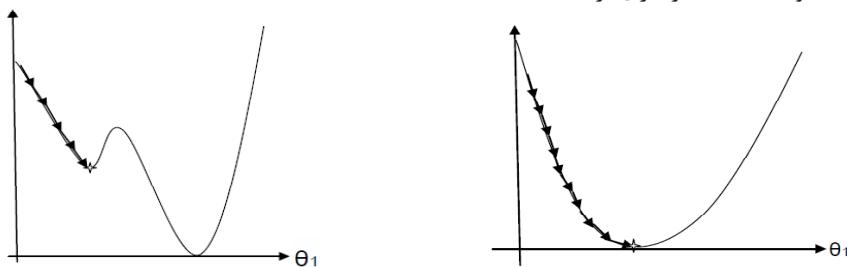
### ۱-۵-۳-۱- اندازه گام های نرخ یادگیری در گرادیان نزولی

برای رابطه گرادیان نزولی،  $\alpha$  را نیز در حالت های مختلف شرح می دهیم.

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

اگر مقدار  $\alpha$  خیلی کوچک باشد گام های رسیدن به نقطه کمینه بسیار کوچک خواهد شد و معایبی مانند بالا رفتن بسیار زیاد زمان رسیدن به حالت کمینه را خواهیم داشت به شکل (۲۲-۱) توجه نمایید.

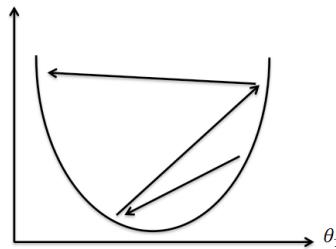
همچنین اگر  $\alpha$  کوچک باشد همانطور که در شکل (۲۳-۱) ملاحظه می نمایید ممکن است الگوریتم در کمینه محلی گیر کند و به کمینه سراسری نرسد.



شکل (۲۳-۱): الگوریتم در مینیمم محلی گیر کده است.

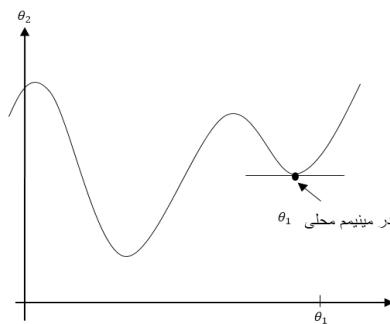
شکل (۲۲-۱): الگوریتم به نقطه مینیمم تابع رسیده است.

و اگر  $\alpha$  مقدارش خیلی بزرگ باشد همانطور که در شکل (۲۴-۱) مشاهده می نمایید ممکن است گام های الگوریتم خیلی بزرگ بوده و از روی نقطه کمینه سراسری عبور کند و به دلیل عدم پیدا کردن کمینه محلی و بالا بودن مقدار خطا در تلاش برای رسیدن به نقطه کمینه،  $\theta$  را در جهت اشتباه به روز کند و مقدار گامها بزرگ تر شده و شرایط بدتری پیدا کند.



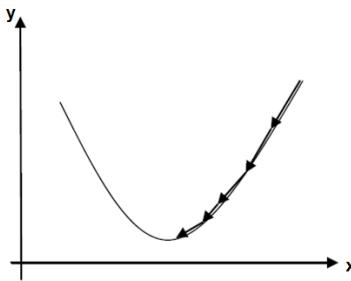
شکل (۲۴-۱): پرش الگوریتم از روی نقطه کمینه سراسری.

اگر نقطه ابتدایی که برای  $\theta$  مقداردهی می‌شود در همان نقطه کمینه محلی باشد به صورت شکل (۲۵-۱) خواهد بود. یعنی در همان کمینه محلی گیر کرده و هرگز به نقطه کمینه سراسری نمی‌رسد. در اصل گرادیان نزولی هیچ فایده‌ای نداشته و چیزی را تغییر نخواهد داد.



شکل (۲۵-۱): مقدار دهی اولیه نامناسب برای بارامترها.

در گرادیان نزولی اندازه هر گام رفته کوچکتر می‌شود تا به نقطه کمینه برسد که یکی از مزایای این امر شتاب گرفتن گرادیان نزولی در رسیدن به نقطه کمینه است که در شکل (۲۶-۱) می‌توانید ملاحظه نمایید.



شکل (۲۶-۱): تغییر گام‌های گرادیان نزولی.

در گرادیان نزولی باید برای  $\alpha$  مقدار مناسب اختصاص یابد تا بتواند با دقت و سرعت مناسب به نقطه کمینه برسد و همچنین اگر شیب گرادیان به سمت یک کمینه محلی باشد، گرادیان نزولی آن شیب را با شیبی که به کمینه سراسری می‌رساند اشتباه گرفته و گام‌های خود را کوچکتر و کوچکتر کرده و در کمینه محلی گیر می‌کند که

این یکی از معایب گرادیان نزولی می‌باشد که بر آن اساس هنگام مشخص کردن مقادیر اولیه  $\theta$ ‌ها و نقاط شروع و جهت حرکت گرادیان نزولی باید با دقت زیادی وارد عمل شد.

## ۱-۶- گرادیان نزولی و تابع هزینه در حل رگرسیون خطی

در این بخش ما گرادیان نزولی را با تابع هزینه تلفیق می‌کنیم تا از آن الگوریتمی برای حل مسائل رگرسیون خطی ایجاد کنیم. کاری که در این بخش انجام می‌دهیم این است که الگوریتم گرادیان نزولی را به  $J(\theta_0, \theta_1)$  اضافه می‌کنیم تا مربعات خطای تابع هزینه را به کمینه برسانیم.

الگوریتم حل رگرسیون با گرادیان نزولی

$$h_{\theta}(x) = \theta_0 + \theta_1 x \quad \{$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\{ \quad \text{Min } J(\theta_0, \theta_1)$$

الگوریتم بالا برای به کمینه رساندن  $J(\theta_0, \theta_1)$  مورد استفاده قرار می‌گیرد که با توجه به الگوریتم بالا روابط زیر را خواهیم داشت.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \times \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$= \frac{\partial}{\partial \theta_j} \times \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

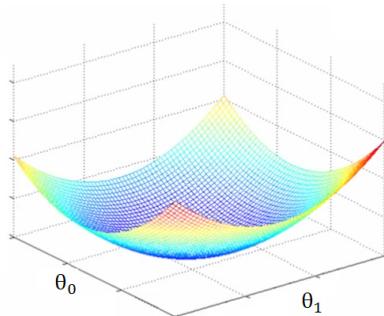
و برای به روزرسانی  $\theta_0$  و  $\theta_1$  روابط زیر را داریم.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

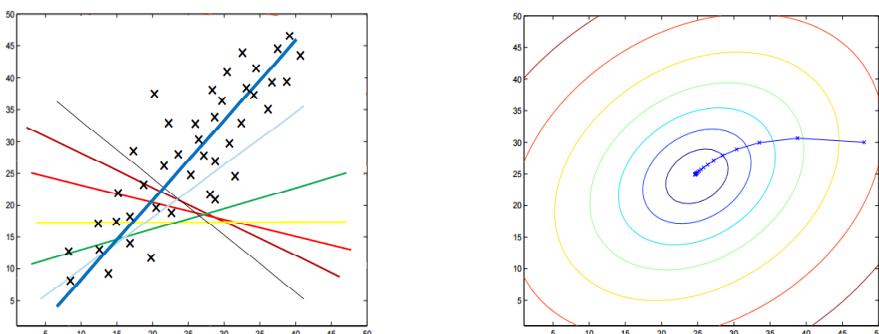
همانطور که می‌دانیم نمودار گرادیان نزولی به شکل صفحه غیرصاف است که ما براساس مقدار دهی اولیه  $\theta$  ممکن است در کمینه محلی گیر کنیم یا به کمینه سراسری برسیم. ولی برای نمودار رگرسیون خطی همیشه ما

شکل کاسه را در فضای چند بعدی خواهیم داشت که در شکل (۲۷-۱) می‌توانید مشاهده نمایید. در اصطلاح به آن تابع محدب<sup>۱</sup> گوییم که برای اینتابع هیچ کمینه محلی وجود ندارد و تنها یک کمینه وجود دارد که آن هم کمینه سراسری است.



شکل (۲۷-۱): تابع هزینه برای رگرسیون خطی.

به شکل (۲۸-۱ و ۲۹-۱) توجه نمایید که در آن نحوه رسیدن الگوریتم به حالت مطلوب نشان داده شده است. شکل (۲۸-۱) فرایند رسیدن الگوریتم توسط گرادیان نزولی را به نقطه کمینه نشان داده که با به روزرسانی پارامترها تابع هزینه کاهش می‌یابد و در نهایت بعد از چند تکرار به نقطه کمینه می‌رسد و متناظر با آن در شکل (۲۹-۱) ملاحظه می‌شود که با تغییر پارامترها خط فرضیه چه تغییراتی را در راستای رسیدن به بهترین حالت در محور مختصات داشته است.



شکل (۲۹-۱): نمایش نحوه رسیدن گرادیان نزولی به نقطه کمینه.

## ۱-۷- مثال عملی برای یک رگرسیون خطی تک متغیره با زبان R

در برنامه زیر هدف تخمین میانگین ارزش خانه‌ها براساس ویژگی میانگین تعداد اتاق‌های هر خانه است. چنانچه در این فصل ملاحظه نمودید برای حل یک رگرسیون خطی تک متغیره نیاز به یک متغیر مستقل و یک متغیر وابسته است. عموماً مجموعه داده‌های استاندارد موجود که در یک رگرسیون مورد استفاده قرار می‌گیرند دارای تعداد زیادی متغیر یا ویژگی می‌باشند که ما در برنامه زیر تنها از یک ویژگی به عنوان متغیر مستقل و یک ویژگی دیگر به عنوان

1. Convex function

متغیر وابسته، از مجموعه ویژگی‌های موجود استفاده می‌نماییم. از طریق خط اول و دوم برنامه یک مجموعه داده با نام boston که یک فایل از نوع csv است و شامل اطلاعاتی درباره تعداد زیادی خانه است را خوانده و با استفاده از خط دوم برنامه، اسمی متغیرهای درون مجموعه داده چاپ می‌شوند. توجه شود که مجموعه داده boston و یا هر مجموعه داده استاندارد دیگری را می‌توان از اینترنت به دست آورد. مخازن و منابع معروف بسیاری مانند سایت UCI با آدرس <http://archive.ics.uci.edu/ml> وجود دارند که بیشتر مجموعه داده‌های استاندارد را از آنجا می‌توان تهیه نمود و همچنین برای آگاهی از جزئیات دقیق مجموعه داده مورد نظر می‌توانید به مستندات موجود برای مجموعه داده مربوطه مراجعه نمایید که معمولاً برای مجموعه داده‌های استاندارد می‌توان مستندات کافی برای مطالعه ساختار مجموعه داده مانند اسمی متغیرها، تعداد نمونه‌های درون آن و غیره را به دست آورد.

```
> boston.dataset = read.csv("d://datasets/Boston.csv")
> names(boston.dataset)
```

```
[1] "crim" "zn" "indus" "chas" "nox" "rm" "age"
[8] "dis" "rad" "tax" "ptratio" "black" "lstat" "medv"
```

خطوط برنامه در بخش زیر مجموعه داده موجود را به دو بخش داده‌های آموزشی و داده‌های آزمون تقسیم می‌کند تا پس از ایجاد مدل در مرحله آموزش، داده‌های جدیدی را بتوان برای آزمون مدل و در نهایت انجام عمل تخمین یا پیش‌بینی، مورد استفاده قرار داد. به صورت کلی در خطوط زیر هدف یافتن تعداد ردیف‌های تشکیل‌دهنده مجموعه داده مورد استفاده است که با آگاهی به آن بتوان ۷۵ درصد از آن را برای استفاده در مرحله آموزش و ۲۵ درصد را برای مرحله آزمون تقسیم کرد.

همان‌طور که مشاهده می‌نمایید تابع nrow() تعداد ردیف مجموعه داده را مشخص می‌کند و تابع floor() بزرگ‌ترین عدد صحیح را برمی‌گرداند. در اصل در آرگومان تابع floor() به تعداد ۷۵ درصد از کل ردیف‌های مجموعه داده محاسبه شده و مشخص می‌شود که باید تا چه ردیفی از کل مجموعه داده باید انتخاب شود و در نهایت تابع sample() بزرگ‌ترین عدد صحیح به دست آمده را انتخاب می‌کند و در ذخیره می‌کند که این عدد نشان‌دهنده ۷۵ درصد از کل ردیف‌های داده موجود در مجموعه داده است.

تابع seq\_len() کارش تولید توالی منظم از مقدار موجود در آرگومان است و تابع sample() کارش نمونه‌برداری از کل داده‌های موجود است که در آرگومان آن مشخص می‌شود. یعنی تنها ۷۵ درصد از مجموعه داده را که از طریق sample.size مشخص شده از مجموعه داده نمونه‌برداری می‌کند و در train.index ذخیره می‌کند و بر این اساس 'مجموعه داده‌های آموزشی' مشخص می‌شود و با دستور [ ] در متغیر train ذخیره می‌شود.

'مجموعه داده آزمون' نیز همان داده‌هایی هستند که با کم کردن مجموعه داده‌های آموزشی از کل مجموعه داده موجود باقی می‌مانند که آنها نیز در متغیر test قرار می‌گیرند.

```
> sample.size <- floor(0.75 * nrow(boston.dataset))
```

```
> train.index <- sample(seq_len(nrow(boston.dataset)), size = sample.size)
> train <- boston.dataset[train.index, ]
> test <- boston.dataset[-train.index, ]
```

خط زیر یک مدل مناسب برای رگرسیون خطی تک متغیره براساس متغیر وابسته `medv` و متغیر مستقل `rm` که داده‌های درون آنها مربوط به مجموعه داده آموزشی است ایجاد می‌کند. تابع (`lm()`) که در خط زیر برای ایجاد مدل استفاده شده، تابعی برای ایجاد مدل‌های خطی و رگرسیون است. متغیر `medv` میانگین ارزش خانه و متغیر `rm` میانگین تعداد اتاق‌های هر خانه می‌باشد.

```
> lm.fit = lm(medv ~ rm, data = train)
> summary(lm.fit)
```

خروجی تابع (`summary(lm.fit)`) به صورت زیر می‌باشد که در اصل جزئیات مدل ایجاد شده براساس داده‌های آموزشی را با استفاده از تابع (`lm`) نشان می‌دهد. خروجی زیر که حاصل مرحله آموزش و ایجاد مدل است نشان‌دهنده این نکته بسیار مهم است که آیا الگوریتم مورد استفاده و مدل ایجاد شده در مواجهه با داده‌های مورد نظر توانسته است نتایج قابل قبولی را ارائه کند و مدل مناسبی را بسازد یا خیر.

در خروجی زیر بخش `Coefficients` نشان‌دهنده عملکرد مناسب و توانایی مدل ایجاد شده است و بخش مربوط به `Residuals` نشان می‌دهد که مدل ایجاد شده به چه اندازه ضعف در تحلیل و ارائه داده‌های مورد استفاده داشته است.

در بخش `Residuals`، تفاوت بین مقادیر واقعی نتیجه مشاهده شده، با نتیجه به دست آمده از مدل پیش‌بینی کننده را می‌توان مشاهده نمود که خروجی آن به صورت پنج مقدار عددی می‌باشد. در این پنج مقدار عددی Median عددی نزدیک به صفر باید باشد و دو مقدار عددی کناری یعنی  $Q_1$  با  $Q_3$  و  $Min$  با  $Max$  هرچه بیشتر متقاضی با هم باشند نشان‌دهنده عملکرد بهتر مدل است.

بخش `Coefficient` ضرایب مدل را نشان می‌دهد. به عبارتی دیگر چنانچه در مباحث مربوط به رگرسیون خطی<sup>۱</sup> ارائه گردید مدل ایجاد شده ما تلاش می‌کند تا یک خط فرضیه<sup>۲</sup> مناسبی را در بهترین و نزدیک‌ترین حالت قرارگیری در بین داده‌ها رسم کند که در آن نقطه مبدأ و شیب خط، مشخص کننده وضعیت و نحوه قرارگیری خط فرضیه است.

Estimate: این بخش متشکل از دو ردیف است که اولی مربوط به تفکیک (`intercept`) و ردیف دوم مربوط به عامل مشترک یا ضریب تأثیر (`coefficient`) است.

Standard Error: این بخش نشان‌دهنده قوت ارتباط متغیرها و تأثیر آنها بر متغیر پیش‌بینی است. در اصل مقدار عددی این خطای نشان‌دهنده میزان انحراف پیش‌بینی کننده از حالت بهینه مطلق است.

$t$ : بیانگر برآورد میزان انحراف معیار است که نشان می‌دهد فرضیه صفر (null) در محاسبات تأثیر داده نشده است.

$\Pr(|t| > t)$ : احتمال مشاهده شدن مقادیر بزرگ‌تر از قدر مطلق  $t$  است.

Residual Standard Error: مشخص‌کننده کیفیت عملکرد رگرسیون خطی در ارائه نتیجه مناسب (fit) است و هرچه خطای بیشتر باشد یعنی عملکرد ضعیف است.

R-squared: نشان‌دهنده عملکرد مناسب مدل در مواجهه با داده‌های واقعی است. اگر به مدل متغیرهای بیشتری اضافه شود و این عمل در بهبود کارایی مدل نتیجه نامناسبی داشته باشد مقدار موجود در این بخش عدد بزرگ‌تری خواهد بود.

F-statistic: نشان می‌دهد که آیا رابطه خوبی بین مدل پیش‌بینی کننده و متغیرهای موجود وجود دارد یا خیر. بزرگی مقدار عددی در این بخش بستگی به تعداد نقاط داده و نقاط پیش‌بینی کننده دارد.

p-value: تخمین احتمال وقوع  $t$ -value است و به صورت کلی نشان‌دهنده این است که به چه اندازه مدل ایجاد شده توانسته در مواجهه با مجموعه داده استفاده شده مناسب عمل کند.

Call:

```
lm(formula = medv ~ rm, data = train)
```

### Residuals:

Min	1Q	Median	3Q	Max
-23.782	-2.914	0.215	3.053	39.884

### Coefficients:

	Estimate	Std. Error	t value	$\Pr(> t )$
(Intercept)	-36.2793	3.1217	-11.62	<2e-16 ***
rm	9.3350	0.4935	18.92	<2e-16 ***
---				

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.864 on 377 degrees of freedom

Multiple R-squared: 0.487, Adjusted R-squared: 0.4856

F-statistic: 357.8 on 1 and 377 DF, p-value: < 2.2e-16

در خط زیر با استفاده از تابع predict() مدل ایجاد شده مورد آزمون قرار گرفته و عمل تخمین انجام گرفته است که بر اساس تابع head() نتایج آن نمایش داده می‌شود.

> head(predict(lm.fit, test, interval = "confidence"), 5)

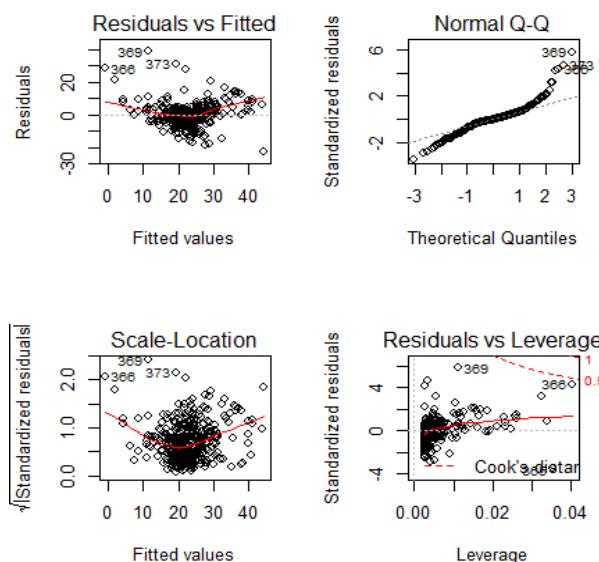
خروجی زیر همان نتیجه نهایی الگوریتم یا مقادیر تخمین است. lwr حد پایین، upr حد بالا و fit مقادیر بین حد بالا و حد پایین است که همان نتیجه مناسب و صحیح (fit) برای اجرای الگوریتم و عمل پیش‌بینی یا تخمین است.

	fit	lwr	upr
2	23.66053	22.95484	24.36622
4	29.04681	28.06763	30.02599
5	30.43772	29.35161	31.52382
18	19.63716	18.88693	20.38740
20	17.18207	16.30215	18.06199

با تابع plot() نتایج به صورت نموداری نشان داده می‌شود که قبل از آن اگر از تابع par() استفاده کنیم هر چهار حالت مختلف نموداری، چنانچه مشاهده می‌نمایید در قالب یک تصویر نشان داده می‌شود.

>par(mfrow = c(2,2))

>plot(lm.fit)



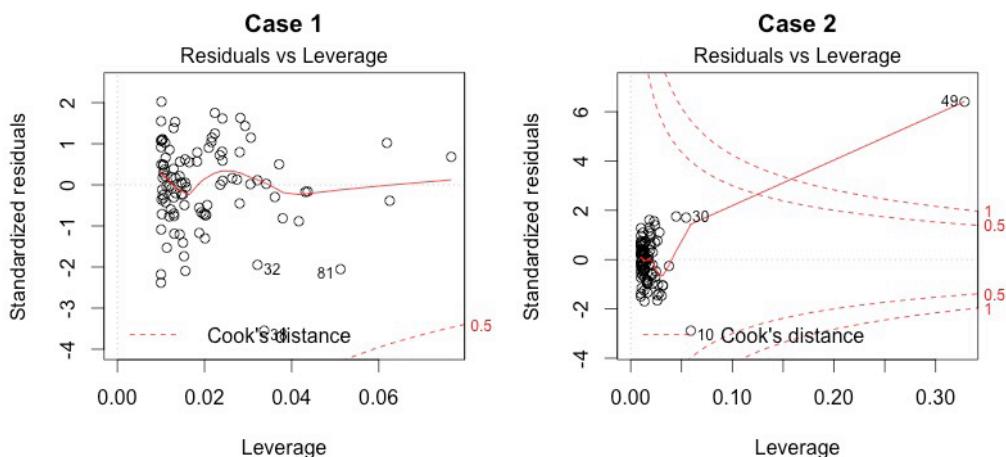
پس از اجرای برنامه رگرسیون خطی وقتی نمودار نهایی را به دست می‌آورید بسیاری از افراد تازه کار گمان می‌کنند که کار با موفقیت به پایان رسیده است اما این طور نیست و حتی اگر در مرحله آموزش، الگوریتم مناسب عمل کند امکان دارد که نتیجه اجرای الگوریتم در مرحله آزمون نامناسب باشد. پس با تحلیل خروجی الگوریتم به صورت نموداری می‌توان به کیفیت خروجی نهایی آن پی برد. توضیحات هر یک از چهار نمودار بالا به صورت زیر می‌باشد.

نمودار Residuals vs Fitted: این نمودار مشخص کننده این است که آیا باقیمانده‌ها (Residuals) دارای یک الگوی غیرخطی می‌باشند یا خیر. باید توجه نمایید که در رگرسیون خطی چنانچه از نامش پیداست الگوهای موجود باید از یک رابطه خطی پیروی کنند که در الگوریتم ایجاد شده ممکن است یک رابطه غیرخطی بین متغیرهای پیش‌بینی و متغیر نتیجه وجود داشته باشد که این نشانه خوبی نیست. در این نمودار هرچه باقیمانده‌ها به صورت مساوی حول یک خط افقی تراویح باشند که این نشانه خوبی نیست. در این نمودار هرچه باقیمانده‌ها به صورت مساوی حول یک خط افقی تراویح باشند که این نشانه خوبی نیست.

نمودار Normal Q-Q: این نمودار مشخص کننده توزیع نرمال<sup>۱</sup> باقیمانده‌ها می‌باشد. اگر باقیمانده‌ها به صورت یک خط راست به ترتیب بر روی خط نقطه‌چین موجود در نمودار قرار گیرند نشان‌دهنده یک نتیجه بهتر است.

نمودار Scale-Location: این نمودار به نمودار منطقه پراکندگی نیز معروف است و نشان‌دهنده این است که آیا باقیمانده‌ها به حول محور خط فرضیه به صورت مساوی پراکنده شده‌اند یا خیر.

نمودار Residuals vs Leverage: براساس این نمودار می‌توان سایر عوامل مؤثر در خروجی الگوریتم مانند داده‌های پرت (outliers) را در صورت وجود شناسایی و بررسی نمود. هرچه میزان (cook's distance) در نمودار کمتر باشد بهتر است. در دو مثال زیر، case1 وضعیت بهتری را نشان می‌دهد.



## ۱-۱-۱- مثال عملی برای یک رگرسیون خطی تک متغیره با زبان پایتون

با توجه به اینکه در ادامه برای برنامه‌نویسی و اجرای الگوریتم‌های یادگیری ماشین از زبان پایتون استفاده می‌شود، بر این اساس اگر آشنایی کافی به این زبان ندارید می‌توانید به پیوست مربوط به آموزش مقدماتی زبان پایتون در ابتدای کتاب مراجعه نمایید.

1. Gaussian distribution

رگرسیون خطی ساده همان رگرسیونی است که از یک متغیر مستقل و یک متغیر وابسته برای ایجاد خط فرضی استفاده می‌نماید. در زیر ما یک الگوریتم رگرسیون تک متغیره را پیاده‌سازی نموده‌ایم که از مجموعه داده diabetes استفاده می‌نماییم که از UCI (یک مخزن مجموعه داده‌های مورد نیاز در داده کاوی می‌باشد که در آدرس <https://archive.ics.uci.edu/ml/datasets.html> می‌باشد) دریافت گردیده است.

بسته‌های نرم‌افزاری مورد نیاز برنامه در سه خط زیر مشخص گردیده است که Matplotlib برای ترسیم نمودار، numpy برای محاسبات ماتریسی، و scikit-learn همان بسته همان باشد که کتابخانه‌ای برای پیاده‌سازی الگوریتم اصلی رگرسیون می‌باشد.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
```

خط زیر مجموعه داده را بارگذاری می‌نماید.

```
diabetes = datasets.load_diabetes()
```

در خط زیر مشخص می‌نماییم که چه تعداد ویژگی از مجموعه داده مورد استفاده قرار گیرد که در اینجا ما تنها از یک ویژگی استفاده می‌نماییم. عدد ۲ مشخص می‌نماید که تنها یک متغیر مستقل و یک متغیر وابسته در این الگوریتم مورد استفاده قرار گیرد.

```
diabetes_X = diabetes.data[:, np.newaxis, 2]
```

خطوط زیر به صورت جدا داده‌های متغیر مستقل و متغیر وابسته را به دو بخش مساوی مجموعه داده‌های آموزشی و مجموعه داده آزمایشی تقسیم می‌نماید.

```
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]
```

خط زیر رگرسیون خطی را ایجاد می‌نماید.

```
regr = linear_model.LinearRegression()
```

خط زیر با مدل رگرسیون ایجاد شده را با استفاده از مجموعه داده‌های آموزشی آموزش می‌دهد.

```
regr.fit(diabetes_X_train, diabetes_y_train)
```

خط زیر با استفاده از مجموعه داده آزمایشی مدل آموزش دیده شده را مورد آزمون قرار می‌دهد.

```
diabetes_y_pred = regr.predict(diabetes_X_test)
```

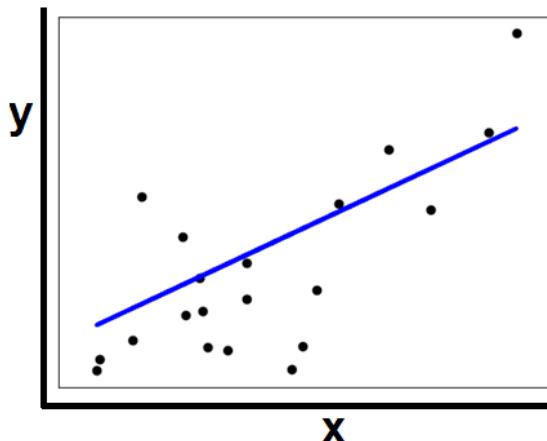
خط زیر خطای الگوریتم را در مرحله آزمون مشخص می‌نماید که به عبارت دیگر همان مشخص کننده توان الگوریتم در تخمین صحیح نیز مشخص می‌گردد.

```
print("Mean squared error: %.2f"
% mean_squared_error(diabetes_y_test, diabetes_y_pred))
```

در ادامه برنامه نمودار نهایی برنامه ترسیم می‌شود که نشان دهنده وضعیت قرار گیری خط فرضی در نمودار است.

```
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show()
```

نمودار زیر خروجی نهایی الگوریتم بالا می‌باشد که در آن نحوه قرار گیری خط فرضی در نمودار بر روی داده‌ها نشان دهنده نتیجه تخمین الگوریتم است.





فصل

## رگرسیون خطی چند متغیره

## ۱-۲- رگرسیون خطی چند متغیره

رگرسیون خطی چند متغیره روشی برای استفاده از چند متغیر مستقل برای تحلیل تغییرات متغیر وابسته است به عبارت دیگر برخلاف رگرسیون خطی تک متغیره، رگرسیون خطی چند متغیره میزان اثر دو یا چند متغیر بر متغیر وابسته را می‌سنجد. در حل مسائل واقعی، رگرسیون خطی تک متغیره کاربرد چندانی ندارد، زیرا در حل یک مسئله با استفاده از شرکت دادن تأثیر ویژگی‌های مختلف، نتایج درست‌تر و قابل قبول‌تری را می‌توان به دست آورد. در این بخش نیز مانند بخش مربوط به رگرسیون تک متغیره به دلیل آشنایی شما با صورت مسئله از مثال خانه استفاده می‌کنیم. چنانچه در رگرسیون خطی تک متغیره گفته شد هدف تخمین قیمت خانه (Y) از روی ویژگی مساحت خانه (X) با فرضیه‌ای به صورت معادله (۱) بود که تنها یک متغیر را شامل می‌شود.

$$h_{\theta}(x) = \theta_0 + \theta_1 x \quad \text{معادله (۱)}$$

و همانطور که در جدول (۱-۲) نشان داده شده است تنها مقادیر مربوط به یک متغیر مستقل را مد نظر قرار می‌دادیم.

جدول (۱-۲): مقادیر مربوط به متغیر مستقل X و متغیر وابسته Y برای رگرسیون تک متغیره.

(x) مساحت	(y) قیمت
۲۱۰۴	۴۶۰
۱۴۱۶	۲۳۲
۱۵۳۴	۳۱۵
۸۳۵	۱۷۸
.....	.....

حال فرض کنید ما تنها با ویژگی مساحت رویه رو نیستیم و با توجه به جدول (۲-۲) ویژگی‌های بیشتری را می‌خواهیم در حل مسئله شرکت دهیم.

جدول (۲-۲): مقادیر مربوط به یک متغیر وابسته و چهار متغیر مستقل.

(x <sub>1</sub> ) مساحت کلی خانه	(x <sub>2</sub> ) تعداد اتاق‌ها	(x <sub>3</sub> ) تعداد طبقات	(x <sub>4</sub> ) عمر خانه	(y) قیمت
۲۱۰۴	۵	۱	۴۵	۴۶۰
۱۴۱۶	۳	۲	۴۰	۲۳۲
۱۵۳۴	۳	۲	۳۰	۳۱۵
۸۳۵	۲	۱	۳۶	۱۷۸
.....	.....	.....	.....	.....

مشاهده می‌شود که علاوه بر مساحت با متغیرهای عمر خانه، تعداد طبقات، تعداد اتاق‌ها، مساحت کلی خانه نیز در این مسئله مطرح است و حتی می‌توانستیم متغیرهای خیلی بیشتری را شرکت دهیم.

برای نشان دادن تعداد ویژگی‌ها از  $n$ ، برای نشان دادن تعداد مقادیر متغیرها از  $m$ ، برای متغیر وابسته از  $y$ ، برای متغیر مستقل از  $x$  که  $\text{x}^{(i)}$  مشخص‌کننده سطر  $i$ م و  $\text{x}_j^{(i)}$  نشان‌دهنده ستون  $j$  است. حال براساس قراردادهای مطرح شده در جدول (۲-۲) تعداد ویژگی‌ها برابر چهار با  $n = 4$  می‌باشد و  $\text{x}^{(2)}$  مشخص‌کننده مقادیر [1416 3 2 40] است. به مثال زیر توجه نمایید.

مثال) اگر  $X^{(4)} = [852 \quad 2 \quad 1 \quad 36]$  را داشته باشیم می‌توان چنانچه در زیر مشاهده می‌نمایید.

$X^{(4)}$  را به صورت برداری نوشت.

$$X^4 = \begin{bmatrix} 852 \\ 2 \\ 1 \\ 36 \end{bmatrix}$$

همان‌طور که می‌دانیم اگر تعدادی از بردارها را در کنار هم قرار دهیم یک ماتریس را شکل می‌دهد و در اینجا نیز با قرار دادن بردارهای ویژگی در کنار هم می‌توان یک ماتریس از مقادیر مربوط به ویژگی‌ها را ایجاد کرد. که در آن  $\text{x}_j^{(i)}$  نشان‌دهنده شماره اندیس یک ویژگی می‌باشد. برای مثال  $\text{x}_2^{(4)}$  نشان‌دهنده بردار چهارم که همان  $X_2^{(4)}$  است و سطر دوم از مقادیر درون آن که بردار است و معادل ۲ می‌باشد. به عبارتی  $2 = X_2^{(4)}$  است.

در گذشته دیدیم که فرضیه ما برای تک متغیره به صورت  $h_{\theta}(x) = \theta_0 + \theta_1 x_1$  بود، ولی برای چند متغیره دیگر نمی‌توان از این فرم ساده استفاده کرد که به صورت کلی  $(x)$  برای  $h_{\theta}$  برای چند متغیره به صورت رابطه (۲) می‌باشد.

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n \quad (2)$$

به مثالی برای رگرسیون چند متغیره توجه نمایید.

مثال) فرض کنید برای تخمین قیمت خانه بر اساس جدول (۲-۲) چهار متغیر داریم و برای مقادیر مربوط به پارامترهای این متغیرها، از مقادیر موجود در بردار زیر استفاده کنیم.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} = \begin{bmatrix} 80 \\ 0.1 \\ 0.01 \\ 3 \\ 2 \end{bmatrix}$$

در این حالت به دلیل وجود چهار متغیر، معادله خط فرضیه زیر را خواهیم داشت.

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

$$h_{\theta}(x) = 80 + 0.1x_1 + 0.01x_2 + 3x_3 + 2x_4$$

چنانچه در بردار  $\theta$  مشاهده می‌کنید پارامتر  $\theta_0$  مساوی عدد  $80$  است. توجه شود که  $\theta_0$  در اصل به صورت  $\theta_0 x_0$  می‌باشد که در هر حالتی  $x_0^{(i)}$  برابر یک می‌باشد و بر این اساس  $[\theta_0 x_0^{(i)}]$  برابر با همان  $\theta_0$  می‌باشد. و همچنین  $x_1^{(i)}$  مقدار  $x_1$  یا متغیر اول با مقدار موجود در سطر  $i$  ام و  $x_2^{(i)}$  مقدار  $x_2$  یا متغیر دوم با مقدار موجود در سطر  $i$  ام و غیره می‌باشد. در ادامه حل مسئله باید گفت که اگر مقادیر پارامترها و متغیرها را با توجه به توضیحات ارائه شده جایگذاری کنیم می‌توانیم خط فرضیه خود را به دست آوریم.

همان‌طور که در بخش مربوط به رگرسیون تک متغیره به صورت مفصل در مورد راه حل رسیدن خط فرضیه به بهترین حالت خود بحث شد، باید به یاد داشته باشیم که چگونگی مقادیر پارامترهای  $\theta$  در رسیدن مسئله به حالت بهینه و در نهایت رسیدن به یک تخمین قیمت مناسب اهمیت دارد.

### ۳-۲- گرادیان نزولی برای رگرسیون خطی چندمتغیره

در این بخش روش استفاده از گرادیان نزولی برای رگرسیون چندمتغیره معرفی می‌گردد. در مباحث قبلی کتاب در مورد نحوه عملکرد گرادیان نزولی بحث شد که در آن تنها دو پارامتر  $\theta$  مطرح بود ولی اگر تعداد ویژگی‌ها زیاد باشد تعداد پارامترها بیش از دو مورد بوده و ابعاد مسئله بالا خواهد بود که در ادامه مطالب نحوه محاسبه مسئله‌ای با ابعاد زیاد را با گرادیان نزولی معرفی می‌کنیم.

در زیر به روابط خط فرضیه، تعداد پارامترها و رابطه تابع هزینه که نشان‌دهنده ابعاد بالای مسئله است توجه نمایید.

$$h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \quad \text{خط فرضیه}$$

$$(\theta_0, \theta_1, \dots, \theta_n) == (\theta) \quad \text{پارامترها}$$

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \quad \text{تابع هزینه}$$

به عبارتی  $J(\theta_0)$  که در اصل می‌توان گفت  $J(\theta_0, \theta_1, \dots, \theta_n)$  را می‌توان با  $J(\theta)$  نشان داد. و بر این اساس به الگوریتم کاهش گرادیان نزولی توجه نمایید.

$$\left\{ \begin{array}{l} \theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \end{array} \right. \quad \text{تکرار تا زمانی که مقدار مناسب به دست آید}$$

همان‌طور که در فصل گشته در مباحث مربوط به الگوریتم گرادیان نزولی مطرح شد برای هر  $\theta_j$  یعنی  $(j=0, \dots, n)$  باید به روزرسانی مقادیر به صورت همزمان و موازی انجام شود و در غیر اینصورت الگوریتم درست عمل نخواهد کرد.

برای یادآوری گرادیان نزولی برای مسئله تک متغیره و مقایسه آن با گرادیان نزولی برای مسئله چندمتغیره به قسمت  $\theta_j$  از الگوریتم گرادیان نزولی در زیر توجه نمایید.

### ۳-۳- بهروزرسانی پارامترها در گرادیان نزولی

اگر تعداد ویژگی برابر یک یا  $n=1$

$$\begin{aligned} \theta_0 &= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 &= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)} \end{aligned} \quad \text{تکرار } \}$$

بهروزرسانی موازی و همزمان  $\theta_0, \theta_1$

در الگوریتم بالا توجه نمایید که در بخش  $\theta_1$  در انتهای رابطه مربوطه،  $x^{(i)}$  اضافه شده است که در رابطه مربوط به  $\theta_0$  وجود ندارد. در مورد دلیل این موضوع همان‌طور که در ابتدای مطالب این فصل گفته شد می‌توان گفت  $x$  مربوط به پارامتر  $\theta_0$  برابر یک است و تأثیری در محاسبه ندارد و در اینجا تنها برای  $\theta_1$  مقدار  $x$  را محاسبه می‌نماییم.

اگر تعداد ویژگی‌ها بیشتر از یک باشد یعنی  $n > 1$  باشد به این معنی است که تعداد پارامترهای ما بیشتر از دو مورد است و برای تمامی پارامترها مقدار  $x^{(i)}$  را محاسبه می‌نماییم و البته در اینجا نیز برای رابطه مربوط به پارامتر  $\theta_0$  همچنان از  $x^{(i)}$  استفاده می‌کنیم.

### ۴-۴- الگوریتم بهروزرسانی پارامترهای چند متغیره

اگر ویژگی‌ها زیاد باشد و یا  $n > 1$

تکرار }

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

بهروزرسانی موازی و همزمان برای  $\theta_j$  که  $j=0, \dots, n-1$  است.

}

با یک مثال برای چندمتغیره که به صورت جداگانه بهروز شدن پارامترها را نشان می‌دهد توجه نمایید.

مثال) فرض کنید که سه ویژگی داریم که در این صورت  $\theta_j$  ها را به صورت زیر خواهیم داشت.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_{10} - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$$\theta_3 := \theta_3 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_3^{(i)}$$

## ۲-۵- مقیاس‌گذاری ویژگی‌ها

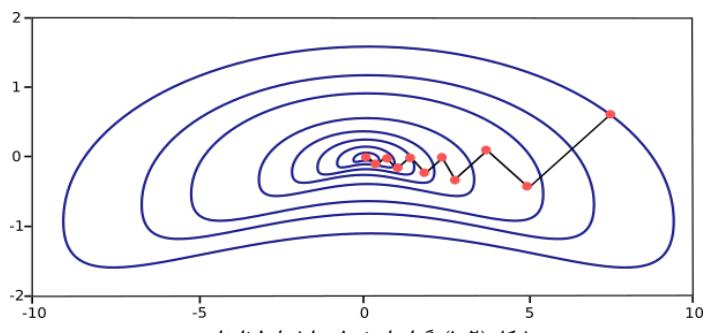
در این بخش در مورد نکاتی صحبت خواهد شد که باعث بهتر شدن کارایی گرادیان نزولی می‌شوند. در اصل اگر ویژگی‌های مسئله در مقیاس، تفاوت زیادی داشته باشند گرادیان نزولی برای رسیدن به نقطه کمینه، بیشتر دچار مشکل می‌شود. به عنوان مثال می‌توان به مسئله خانه اشاره کرد که در آن مقادیر مربوط به تعداد اتاق‌ها با مقادیر مربوط به مساحت خانه در مقیاس‌هایی با اختلاف زیاد می‌باشند که در محاسبات باعث به وجود آمدن مشکلاتی خواهند بود که در ادامه معرفی شده است.

به مثالی برای اختلاف مقیاس ویژگی‌ها توجه نمایید.

مثال) ویژگی مساحت خانه را داریم که در آن کمترین مساحت  $30 \text{ m}^2$  و بیشترین که در  $2000 \text{ m}^2$  است در نظر گرفته شده است ( $x_1 = 30 \text{ m}^2$ ) و برای ویژگی تعداد اتاق‌ها که از ۱ تا ۵ است ( $x_2 = 1 \text{ to } 5$ ) را داریم.

در شرایطی که مقادیر متغیرها در مقیاس‌هایی با اختلاف زیاد باشند تابع هزینه در رسیدن به نقطه کمینه ممکن است دچار مشکل شود و یا اصلاً نتواند به وضعیت بهینه برسد.

به شکل (۱-۲) توجه شود که در آن نمودار تابع هزینه با عرض زیاد و ارتفاع خیلی کم نشان داده شده است و این وضعیت بیانگر این نکته می‌باشد که مسیر رسیدن به نقطه کمینه بسیار پیچیده‌تر بوده و بهینه نخواهد بود.



شکل (۱-۲): گرادیان نزولی با شرایط نامناسب.

برای از بین بردن مشکل اختلاف مقیاس باشد مقادیر متغیرها را مقیاس‌گذاری<sup>۱</sup> کنیم تا مسیر رسیدن به نقطه کمینه را هموارتر کنیم.

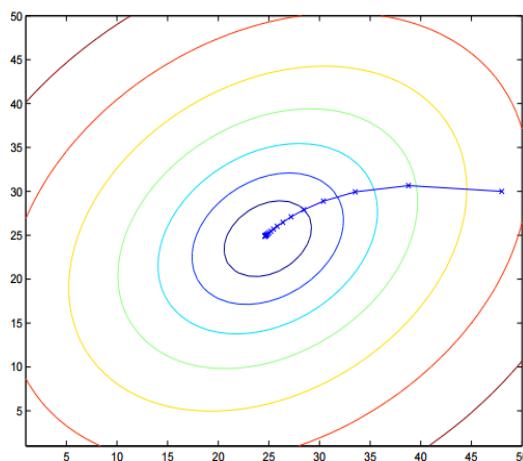
فرض کنید برای مسئله خانه همان ویژگی‌های اندازه و تعداد اتاق‌ها را می‌خواهیم به یک مقیاس نزدیک به هم تبدیل کنیم که برای این موضوع با توجه به رابطه (۳) هر کدام از مقادیر ویژگی‌ها را به بیشترین مقدار همان ویژگی تقسیم می‌کنیم و در نهایت با این روش شکل (۲-۲) را خواهیم داشت که چنانچه مشاهده می‌نمایید در آن نمودار تابع هزینه شکل دوارتری را دارد و تابع هزینه، مسیر راحت‌تری برای رسیدن به نقطه کمینه تابع را خواهد داشت.

$$x_{j \text{ scaled}} = \frac{x_j}{\max} \quad \text{رابطه (۳)}$$

مثال) برای هر دو ویژگی  $x_1$  و  $x_2$  از مسئله خانه روابط زیر را خواهیم داشت.

$$x_{2 \text{ scaled}} = \frac{\text{تعداد اتاق}}{5} \quad \text{و} \quad x_{1 \text{ scaled}} = \frac{\text{مساحت}}{1000}$$

در کل باید توجه داشت که هدف ما تبدیل مقادیری در مقیاس‌های مختلف به یک مقیاس نزدیک به هم است. مقدار  $x_1$  و  $x_2$  در این مثال به صورت  $0 \leq x_2 \leq 1$  و  $0 \leq x_1 \leq 1$  خواهد بود و مقدار  $x_1$  و  $x_2$  دیگر مثل قبل تفاوت بالایی نخواهد داشت و مقادیر به هم نزدیک بوده و در یک مقیاس قرار گرفته و باعث می‌شود گرادیان نزولی برای رسیدن به نقطه کمینه مسیر ساده‌تری را پیماید و سرعت بالاتری نیز داشته باشد.



شکل (۲-۲): گرادیان نزولی با شرایط مناسب.

باید در فرایند مقیاس‌گذاری تمام ویژگی‌ها را گرفته و آنها را با روش تغییر مقیاس که توضیح داده شد در محدوده تقریبی منفی یک و یک  $-1 \leq x_i \leq 1$  - مقیاس‌گذاری کرد. توجه شود اگر رنج فاصله مقادیر کمی بیشتر از

فاصله منفی یک و یک باشد اشکالی ندارد مانند  $\frac{1}{3} \leq x_2 \leq \frac{1}{3}$  و یا حتی  $-3 \leq x_3 \leq 3$  می‌تواند قابل قبول باشد که توسط طراح الگوریتم براساس شرایط موجود انتخاب می‌شود.

## ۴-۶- نرمال کردن متغیر

علاوه بر روش مقیاس‌گذاری ذکر شده در بخش قبلی، روش‌های متعددی برای این منظور وجود دارد که یکی از این روش‌ها به نرمال‌سازی متوسط معروف است که در رابطه (۴) ملاحظه می‌نمایید.

$$\text{رابطه (4)} \quad s_i = \frac{x_i - \mu_i}{\max - \min}$$

توجه شود که در روابط فوق:  $x_i$  متغیر مربوطه و  $\mu_i$  حد وسط مقادیر موجود در مجموعه داده‌های مربوط به ویژگی مربوطه می‌باشد و  $s_i$  با تغیریکمترین مقدار از بیشترین مقدار متغیر مشخص می‌شود.

مثال) اگر خانه‌ای با ویژگی‌های (۲۰۰۰ تا ۳۰) و (۱ تا ۵) ( $x_1 = 1$  تا  $x_2 = 5$ ) را داشته باشیم با استفاده از رابطه (۴) به صورت زیر عمل می‌کنیم.

$$x_1 = \frac{x_i - \mu_i}{\max - \min} = \frac{\text{اندازه خانه} - 1000}{2000}$$

$$x_2 = \frac{x_i - \mu_i}{\max - \min} = \frac{\text{تعداد اتاق‌ها} - 2}{4}$$

مثلًا برای مساحت خانه که از صفر تا دو هزار می‌باشد مقدار هزار را برای  $\mu_i$  در نظر گرفته و برای تعداد اتاق‌ها از ۱ تا ۵ بود، مقدار ۲ را که حد وسط است در نظر گرفتیم.

## ۴-۷- نرخ یادگیری در رگرسیون خطی چند متغیره

همان‌گونه که در بخش‌های قبلی گفته شد نرخ یادگیری تعیین‌کننده اندازه گام‌های گرادیان نزولی برای رسیدن به نقطه کمینه می‌باشد که آن را با  $\alpha$  مشخص می‌کنیم.

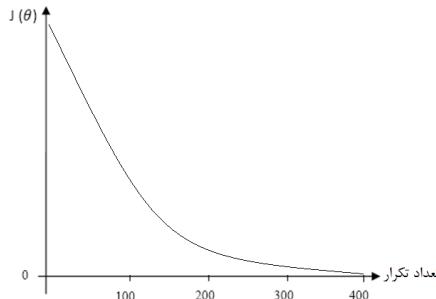
نرخ یادگیری در هر تکرار اندازه کاهش یا افزایش مقدار  $\theta_j$  را مشخص می‌کند. که برای این منظور به رابطه زیر توجه شود.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

در گرادیان نزولی آگاهی از اینکه در هر تکرار شبکه کاهش پیدا می‌کند مهم است و باید برای گرادیان نزولی مقدار مناسبی برای  $\alpha$  انتخاب شود و باید تعیین شود که مقدار  $J(\theta)$  در هر تکرار به میزان مناسبی کاهش پیدا می‌کند

1. Mean normalization

یا خیر. در نهایت وقتی به مقدار کمینه ممکن رسید دیگر در هر تکرار مقدارش کاهش نمی‌یابد و ثابت می‌شود. به شکل (۳-۲) توجه شود که پس از ۱۰۰ بار تکرار به میزان قابل توجهی کاهش یافته و بعد از ۴۰۰ تکرار به نقطه کمینه رسیده و دیگر تغییراتی در آن مشاهده نمی‌شود.



شکل (۳-۲): شرایط قرارگیریتابع هزینه براساس تعداد تکرار.

### ۱-۳- درجه چندجمله‌ای معادله فرضیه رگرسیون خطی

یکی از مباحث مهم رگرسیون خطی درجه چندجمله‌ای معادله فرضیه می‌باشد. در مسائل مختلف تعیین مناسب درجه چندجمله‌ای ارتباط مستقیم با رسیدن الگوریتم به شرایط بهینه دارد.

در ادامه در مورد متغیرها و اینکه دلایل به توان رساندن متغیرها و ایجاد چندجمله‌ای‌های درجات بالاتر برای  $h_{\theta}(x)$  چه می‌باشد بحث خواهیم کرد.

ابتدا به مثال زیر توجه نمایید:

مثال) خانه‌ای به شکل مستطیل که طول و عرض آن برای ما معلوم است را داریم. حال فرض کنید برای تخمین قیمت آن، اندازه آن یکی از ویژگی‌های تعیین کننده قیمت است.

برای حل این مسئله ما می‌توانیم طول را به عنوان یک پارامتر و عرض را به عنوان یک پارامتر دیگر درنظر بگیریم.

$$h_{\theta}(x) = \theta_0 + \theta_1 \text{عرض خانه} + \theta_2 \text{طول خانه} \quad (\text{عرض خانه})$$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

به دلیل اینکه طول و عرض خانه همان مساحت را به ما می‌دهد پس احتیاجی به نوشتن طول و عرض به صورت دو ویژگی جداگانه نیست و می‌توان آن دو را با هم ضرب کرده و مساحت را به عنوان یک ویژگی واحد نوشت و بر این اساس می‌توان از افزودن ویژگی‌های بی‌مورد و اضافه جلوگیری کرد.

مساحت = طول  $x$  عرض

$$h_{\theta}(x) = \theta_0 + \theta_1 \text{مساحت} = \theta_0 + \theta_1 x$$

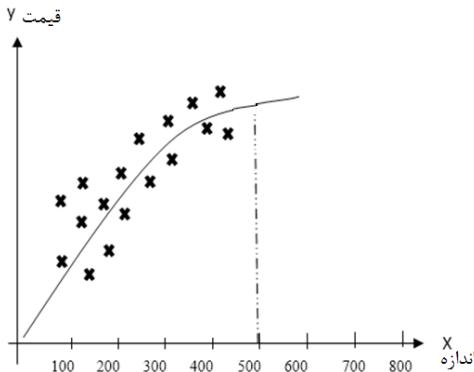
حال با توجه به مسئله فوق به دلایل انتخاب چند جمله‌ای‌ها می‌پردازیم.

مدل چندجمله‌ای برای  $h_{\theta}(x)$  روشی دیگر برای رسیدن به جواب بهینه است که در زیر به توضیح آن خواهیم پرداخت.

درجه چندجمله‌ای در مسائل مختلف بسیار تعیین‌کننده می‌باشد در برخی از مسائل انتخاب  $h_{\theta}(x)$  با درجه بالاتر می‌تواند مناسب باشد و نتیجه مطلوبی را حاصل نماید.

به عنوان مثال فرض کنید می‌خواهیم از معادله درجه دوم که در زیر مشاهده می‌نمایید استفاده کنیم و بر این اساس شکل (۶-۲) را داریم.

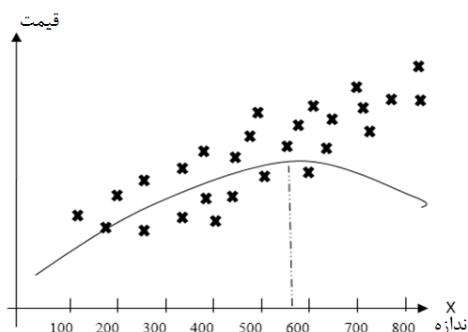
$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$$



شکل (۶-۲): قرار گیری خط فرضیه در شرایط مناسب.

نکته: مقادیر موجود در اشکال فرضی می‌باشد و صرفاً جهت درک بهتر ارائه شده است.

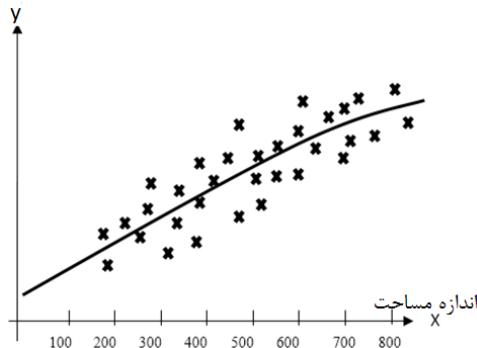
در برخی مواقع درجه دوم کافی نبوده و برای حل آن باید از درجه سوم استفاده شود همان‌طور که در شکل (۶-۲) مشاهده می‌شود درجه دو جمله‌ای مانند  $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$  برای قیمت‌های خانه‌ای تامساحت ۵۰ متر به پایین مناسب عمل کرده ولی در مترازهای بالا این درجه چندجمله‌ای ممکن است مناسب عمل نکند.



شکل (۷-۲): قرار گیری خط فرضیه در حالت نامناسب.

چنانچه در شکل (۷-۲) مشاهده می‌شود تابع ما بعد از محدوده‌ای در حدود ۶۰۰ متر از حالت بهینه و مناسب خارج شده است. برای مسائل مختلف که ویژگی‌های آنها در مقیاس‌های مختلفی می‌باشد درجه چندجمله‌ای مناسب می‌تواند بسیار کلیدی باشد. مثلاً می‌توانیم از تابعی با درجه سه جمله‌ای برای مسئله جاری استفاده کنیم که در زیر می‌توانید معادله چندجمله‌ای درجه سوم و براساس شکل (۸-۲) نحوه ترسیم خط فرضیه را مشاهده نماییم که توانسته مقادیر بالاتری از اندازه مساحت را پوشش دهد.

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$



شکل (۸-۲): قرارگیری خط فرضیه در شرایط مناسب.

لازم به ذکر است که هرچه درجه چندجمله‌ای بالاتر رود هزینه محاسبات نیز بیشتر می‌شود. چنانچه در گذشته گفته شد در مسائلی که مقیاس مقادیر فاصله زیادی با هم دارند حتماً باید از تکنیک مقیاس‌گذاری استفاده شود. مثلاً اگر برای مساحت خانه‌ها مقادیر در مقیاس‌های خیلی متفاوت از هم باشند مانند مقادیری که در زیر ملاحظه می‌نماییم، در این حالت برای پایین آوردن هزینه محاسبات می‌توان از مقیاس‌گذاری استفاده کرد.

$$\begin{cases} size^1 = 1 \\ size^2 = 1,000,000 \\ size^3 = 1,000,000,000 \end{cases} \quad <= \quad \begin{cases} x_1 = (size)^1 \\ x_2 = (size)^2 \\ x_3 = (size)^3 \end{cases}$$

چنانچه مشاهده می‌شود برای متغیر ( $x_3$ ) محدوده مقادیر بسیار بزرگ می‌باشد و انتخاب در حد یک جمله‌ای یا حتی دو جمله‌ای نمی‌تواند برای محاسبه مقادیری در آن محدوده مناسب عمل کند و آن مقادیر را پوشش دهد که با عمل مقیاس‌گذاری می‌توان مشکل را برطرف کرد.

### ۹-۳ - مثال عملی برای یک رگرسیون خطی چند متغیره با $\mathbb{R}$ .

این برنامه که برای رگرسیون چند متغیره می‌باشد حالت کامل تری از برنامه رگرسیون تک متغیره‌ای است که در فصل قبل معرفی گردید. این برنامه در تمامی خطوط خود با برنامه قبلی یکسان است و تنها در خط مربوط بهتابع () که برای ایجاد مدل رگرسیون است تفاوت دارد. برای اینکه هدف ایجاد یک رگرسیون چند متغیره است

حداقل از دو متغیر مستقل استفاده شده است که می‌تواند تعداد متغیرهای مستقل بیشتر هم باشد. در تابع lm() متغیر وابسته medv و متغیرهای مستقل lstat و age می‌باشد. چنانچه در فصل قبل نیز بیان گردید برای کسب اطلاعات بیشتر درباره متغیرهای موجود در یک مجموعه داده می‌توانید به مستندات مجموعه داده که عموماً در منبع ذخیره شده مجموعه داده استاندارد وجود دارد، مراجعه نمایید.

```
> boston.dataset = read.csv("d://datasets/Boston.csv")
> names(boston.dataset)
[1] "X"   "crim" "zn"   "indus" "chas"  "nox"  "rm"   "age"   "dis"
[10] "rad"  "tax"  "ptratio" "black" "lstat" "medv"
> set.seed(123)
> sample.size <- floor(0.75 * nrow(boston.dataset))
> train.index <- sample(seq_len(nrow(boston.dataset)), size = sample.size)
> train <- boston.dataset[train.index, ]
> test <- boston.dataset[-train.index, ]
>
> lm.fit = lm(medv ~ lstat + age,data = train)
>
> summary(lm.fit)
```

Call:

lm(formula = medv ~ lstat + age, data = train)

Residuals:

	Min	1Q	Median	3Q	Max
	-16.093	-3.914	-1.351	1.830	22.913

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	32.77880	0.84970	38.577	< 2e-16 ***
lstat	-1.03735	0.05545	-18.707	< 2e-16 ***
age	0.04194	0.01404	2.988	0.00299 **
---				

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.158 on 376 degrees of freedom

Multiple R-squared: 0.5434, Adjusted R-squared: 0.541

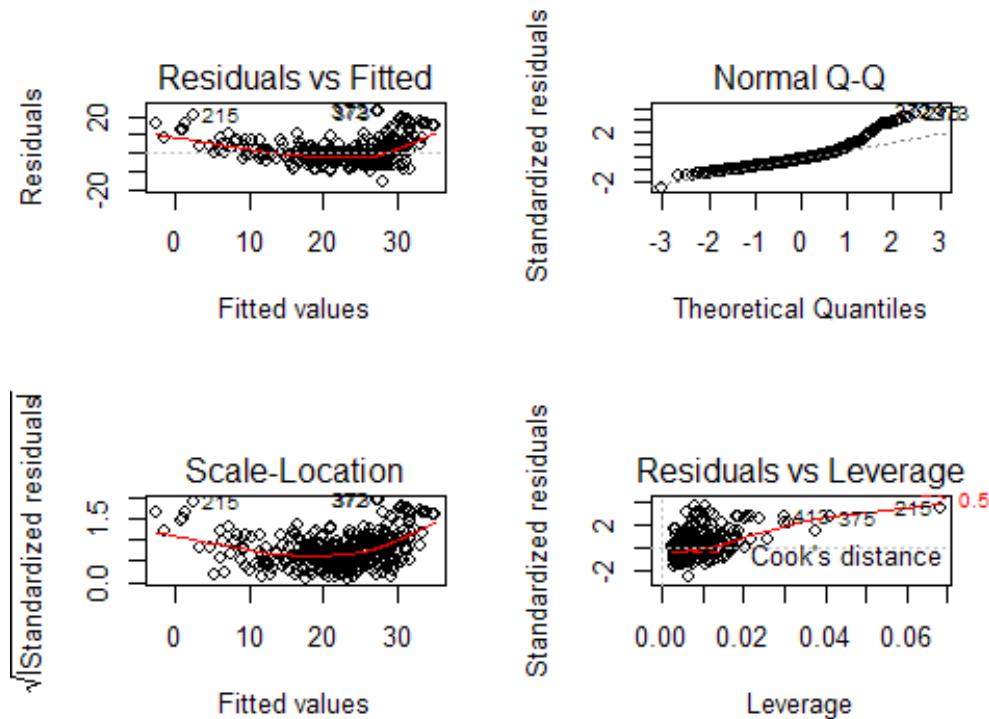
F-statistic: 223.7 on 2 and 376 DF, p-value: < 2.2e-16

```
> head(predict(lm.fit, test, interval = "prediction"), 5)
```

	fit	lwr	upr
6	29.836143	17.693372	41.97891
8	16.944119	4.801176	29.08706
9	5.925098	-6.298592	18.14879
10	18.642884	6.510586	30.77518
18	20.987484	8.859785	33.11518

```
> par(mfrow = c(2,2))
```

```
> plot(lm.fit)
```



## ۲- مثال عملی برای یک رگرسیون خطی چند متغیره با زبان پایتون

برای پیاده‌سازی رگرسیون خطی چند متغیره دقیقاً مانند رگرسیون خطی ساده یا تک متغیره عمل می‌نماییم. تنها تفاوت موجود در کدهای رگرسیون تک متغیره و رگرسیون چند متغیره در تعداد متغیر استفاده شده در

الگوریتم است. برای اجرای الگوریتم چند متغیره باید دقیقاً کدهای فصل قبل را مورد استفاده قرار داد و تنها عدد موجود در خط زیر را به ۹ تغییر داد.

```
diabetes_X = diabetes.data[:, np.newaxis, 9]
```

خط برنامه بالا به دلیل تعداد متغیرهای موجود در مجموعه داده مورد استفاده می‌تواند حد اکثر تا ۹ مقداردهی شود و لی اگر شما بخواهید متغیرهای کمتری را در محاسبات شرکت دهید می‌توانید در خط بالا مشخص نمایید.



فصال

## رگرسیون لاجستیک

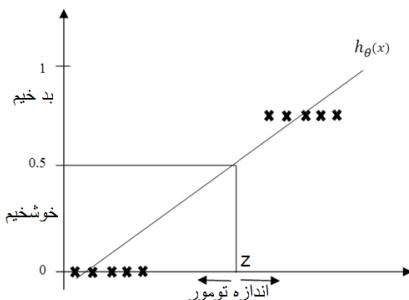
### ۱-۳- رگرسیون لاجستیک

در این فصل می‌خواهیم نحوه عملکرد رگرسیون لاجستیک<sup>۱</sup> و مسائلی که این الگوریتم در آن کاربرد دارد را مورد بررسی قرار دهیم. رگرسیون لاجستیک برای حل مسائل دسته‌بندی مورد استفاده قرار می‌گیرد و یکی از الگوریتم‌های یادگیری با نظارت به حساب می‌آید. همان‌گونه که در فصل اول به آن اشاره شد دسته‌بندی از جمله روش‌هایی است که در آن برای هر کدام از رکوردهای مجموعه داده مورد کاوش، یک برچسب که بیانگر حقیقتی در مسئله است، وجود دارد. این برچسب سبب می‌شود که هر الگوریتم دسته‌بندی، یک الگوریتم با نظارت محاسب شود که رگرسیون نیز در این دسته قرار دارد. حال برای شروع به دلیل اینکه در فصل‌های گذشته با رگرسیون خطی آشنا شده‌اید یک مسئله ساده دسته‌بندی را با آن حل می‌کنیم تا در مورد فرایند دسته‌بندی آگاهی پیدا کنید. به مثال زیر توجه نمایید.

مثال) فرض کنید یک بیمار مبتلا به غده سرطانی وجود دارد که درباره غده بیمار یک مجموعه داده به دست آورده‌ایم و بر آن اساس می‌خواهیم از خوش‌خیم یا بدخیم بودن غده اطلاع پیدا کنیم.

ابتدا باید دو کلاس یا دو دسته برای تفکیک داده‌های مربوط به غده در نظر بگیریم. دسته مربوط به غده خوش‌خیم و دسته مربوط به غده بدخیم و تمامی داده‌های مربوط به غده را باید در دسته مناسب با آن قرار دهیم.

در این مسئله برای هر نمونه آموزشی بدخیم بودن یا مثبت است یا منفی که بر این اساس داده‌ها را می‌توانیم با یک تقسیم‌بندی شرطی در دسته مربوط به خود قرار دهیم به عبارتی اگر شرط یک داده مربوط به غده برای بدخیم بودن برقرار است، آن داده در دسته بدخیم‌ها و در غیر این صورت در دسته خوش‌خیم‌ها قرار خواهد گرفت. معمولاً یک حد آستانه‌ای<sup>۲</sup> را به عنوان شرط موردنظر تعیین می‌کنند چنانچه در شکل (۱-۳) مشاهده می‌نمایید، برای سادگی تنها ویژگی اندازه تومور در نمودار مشخص گردیده است که اگر از یک حد آستانه‌ای مانند ۵٪ کوچکتر باشد آن را خوش‌خیم و از حد آستانه بزرگ‌تر باشد آن را در دسته بدخیم قرار می‌دهیم.



شکل (۱-۳): دسته‌بندی غده‌های خوش‌خیم و بدخیم با رگرسیون خطی.

چنانچه در شکل (۱-۳) مشاهده می‌شود یک خط افقی به خط فرضیه (که از روش رگرسیون یک متغیره که در

1. Logistic regression  
2. Threshold

فصل ۲ بحث شد بدست می‌آید) از عدد حد آستانه که همان ۰.۵ است رسم شده و از نقطه اتصال آن خط با خط فرضیه به نقطه  $z$  یک خط دیگر ترسیم شده است. هر داده موجود در منطقه کوچک‌تر از ۰.۵ و  $z$  را در دسته خوش‌خیم و بقیه را در دسته بدخیم قرار می‌دهیم.

به عبارتی دیگر حد آستانه برای  $h_\theta(x)$  برابر با ۰.۵ می‌باشد و شرط زیر برقرار است.

$$\begin{cases} h_\theta(x) \geq 0.5 & \Rightarrow y = 1 \\ h_\theta(x) < 0.5 & \Rightarrow y = 0 \end{cases}$$

در شرط فوق  $y=1$  یعنی داده به غده‌های سلطانی و  $y=0$  یعنی داده به غده‌های غیرسلطانی متعلق است. با اینکه مثال فوق به دلایلی می‌تواند به مسئله پاسخ مناسب دهد ولی شرایطی وجود دارد که در آن رگرسیون خطی نمی‌تواند مناسب عمل کند.

### ۳-۱-۱ - دسته‌بندی با استفاده از رگرسیون لاجستیک

در دسته‌بندی هدف قرار دادن داده‌هایی که دارای برچسب مشخصی هستند در دسته‌های مربوط به آن برچسب مشخص است که رگرسیون لاجستیک یک الگوریتم با نظارت برای دسته‌بندی داده‌های دارای برچسب است.

### ۳-۱-۲ - معرفی خط فرضیه در رگرسیون لاجستیک

رابطه خط فرضیه برای رگرسیون لاجستیک به صورت زیر می‌باشد.

$$h_\theta(x) = g(\theta^T X) \quad (1)$$

در رگرسیون لاجیسک مقادیر  $h_\theta(x)$  را همیشه به مقدار صفر یا یک تبدیل می‌کنیم. برای این کار نتیجه به دست آمده با  $h_\theta(x)$  را درون یکتابع سیگموئید<sup>۱</sup> قرار می‌دهیم تا مقادیر عددی را به مقادیر صفر یا یک تبدیل کند.

### ۳-۱-۳ - تابع سیگموئید

تابع سیگموئید که در اینجا آن را با حرف  $g$  نشان می‌دهیم، یک تابع برای حد آستانه‌گیری است که بر آن اساس ما می‌توانیم مقادیر عددی را به صفر یا یک تبدیل کنیم. اگر  $\theta^T x$  را با حرف  $Z$  نشان دهیم روابط زیر را برای سیگموئید به صورت زیر خواهیم داشت.

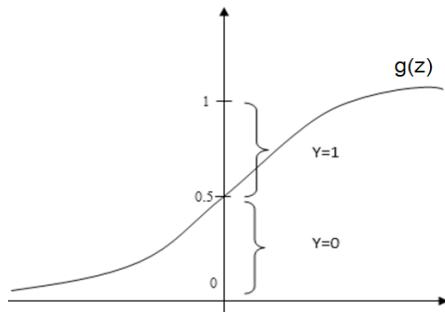
$$Z = (\theta^T X) \quad (2)$$

$$g(Z) = \frac{1}{1+e^{-Z}} = \frac{1}{1+e^{-\theta^T x}} \quad (3)$$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \quad (4)$$

1. Sigmoid function

تابع سیگموئید به تابع لاجستیک نیز معروف است و نمودار آن به مانند شکل (۲-۴) می‌باشد.



شکل (۲-۳): نمودار تابع سیگموئید.

شرط زیر براساس نمودار شکل (۲-۳) برای تابع سیگموئید برقرار است.

$$\begin{cases} 0 \leq h_{\theta}(x) < 1 \\ h_{\theta}(x) \geq 0.5 \Rightarrow y = 1 \\ h_{\theta}(x) < 0.5 \Rightarrow y = 0 \end{cases}$$

به یک مثال دیگر در مورد غده سرطانی توجه نمایید که با رگرسیون لاجستیک محاسبه شده است. البته هدف نشان دادن نحوه عملکرد کلی رگرسیون لاجستیک می‌باشد.

مثال) هدف تخمین احتمال مثبت شدن غده سرطانی برای حالت بدخیم یا  $y=1$  به ازای ورودی  $x$  است.

فرض کنید بردار  $x$  ورودی ما به صورت  $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{اندازه غده} \end{bmatrix}$  می‌باشد و جواب نهایی برای  $h_{\theta}(x)$  به صورت

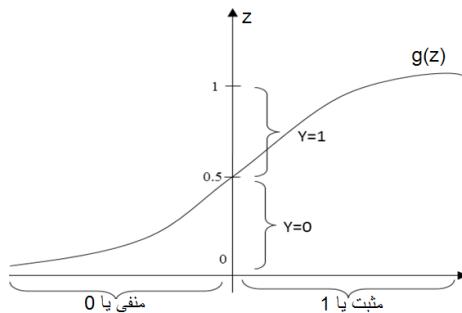
حال اگر مقدار  $h_{\theta}(x)$  را در یک تابع سیگموئید قرار دهیم به دلیل بیشتر بودن آن از مقدار حد آستانه که همان 0.5 در نظر گرفته شده است،  $y=1$  یا غده بدخیم خواهد بود. توجه شود که  $h_{\theta}(x) = 0.7$  نشان‌دهنده آن است که احتمال بدخیم بودن و قرار گرفتن غده براساس ورودی ( $x_1$ ) در دسته بدخیم‌ها 0.7 است.

### ۳-۳- مرز تصمیم<sup>۱</sup> در رگرسیون لاجستیک

مرز تصمیم خطی است که ما می‌توانیم با آن مرز حالتی را از حالت دیگر جدا کرده و بر آن اساس تصمیمات خود را اتخاذ کنیم. همان‌گونه که در مثال قبلی ملاحظه نمودید احتمال مثبت بودن غده بدخیم را 0.7 به دست آوردهیم حال این سؤال پیش می‌آید که چگونه می‌توان گفت این عدد نشان‌دهنده سرطانی بودن غده است و پاسخ این است که ما براساس یک حد آستانه‌ای که همان مرز تصمیم ما از آن می‌گذرد توانستیم تصمیم بگیریم. مثلاً

1. Decision boundary

برای مقادیر بالای ۰.۵ غده را سلطانی در نظر گرفتیم. به شکل (۳-۳) توجه نمایید.



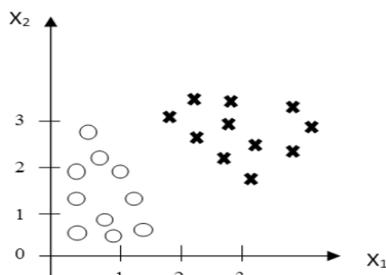
شکل (۳-۳): تعیین بدخیم یا خوش خیم بودن غده براساس داده ورودی  $X$ .

### ۳-۳-۱- روش محاسبه محل مرز تصمیم در فضای نمونه‌ها

فرض کنید طبق نمودار شکل (۴-۳) مجموعه‌ای از داده‌های آموزش داریم که خط فرضیه و مقادیر پارامترها نیز به صورت زیر می‌باشد.

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$\theta_0 = -3, \quad \theta_1 = 1, \quad \theta_2 = 1$$



شکل (۴-۳): مجموعه نمونه‌های آموزشی مربوط به غده‌ای بدخیم و خوش خیم.

طبق مقادیر ارائه شده در فرض بالا ما برداری به شکل زیر خواهیم داشت.

$$\theta = [-3 \quad 1 \quad 1] \Rightarrow \theta^T = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

و اگر  $\theta$  را برای  $h_{\theta}(x)$  مقداردهی کنیم روابط زیر را خواهیم داشت.

$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$h_{\theta}(x) = g(\theta^T x)$$

$$-3 + 1(x_1) + 1(x_2) = \theta^T x$$

و براساس معادلات بالا شرط‌های زیر برقرار است.

$$-3 + x_1 + x_2 \geq 0$$

$$x_1 + x_2 \geq 3$$

در نتیجه روابط زیر را خواهیم داشت.

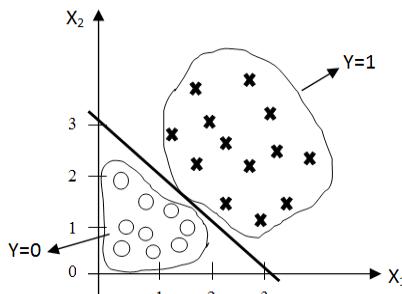
$$x_1 + x_2 = 3$$

$$x_1 + x_2 > 3$$

نتیجه دو رابطه بالا نشان می‌دهد که با توجه به وجود دو محور  $x_1$  و  $x_2$  همان‌گونه که در شکل (۵-۳) ملاحظه می‌نماییم. مرز تصمیمی بر روی عدد بزرگتر و مساوی ۳ قرار دارد و همین‌طور شرط برقراری درستی برای فرض موجود نیز به صورت زیر است.

$$(اگر \theta^T x \geq 0 \text{ سپس } y=1)$$

یعنی در صورتی که مقدار  $\theta^T x$  بزرگ‌تر یا مساوی صفر باشند داده‌های موجود در آن منطقه گویای مثبت یا حالت بدخیم می‌باشند که با  $y=1$  مشخص شده است.



شکل (۵-۳): نحوه ترسیم مرز تصمیمی برای تشخیص غده بدخیم.

### ۳-۴ - مرز تصمیمی غیر خطی<sup>۱</sup>

همانطور که به خاطر دارید در قسمت رگرسیون خطی در مورد درجه چندجمله‌ای صحبت شد. در اینجا می‌خواهیم نشان دهیم که چگونه برای حل مسائل مختلف در رگرسیون لاجستیک می‌توانیم برای  $h_\theta(x)$  از چندجمله‌ای درجات بالاتر استفاده کنیم.

به یک مثال توجه نمایید:

مثال) فرض کنید  $h_\theta(x)$  یا فرضیه ما به صورت زیر باشد.

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

در اینجا فرض ما بر این است که مقادیر  $\theta$ ‌ها، بردار زیر را تشکیل می‌دهند.

1. None linear decision boundary

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

طبق مفروضات فوق به روابط زیر توجه نمایید.

در صورتی که  $\theta^T x \geq 0$  باشد  $y=1$  خواهد بود.

$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

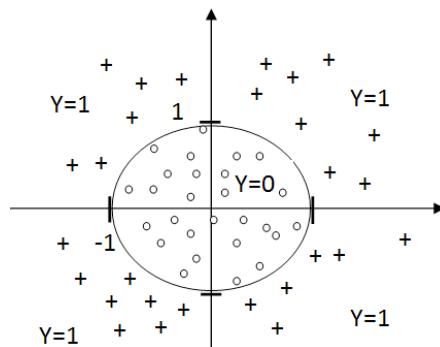
$$h_{\theta}(x) = g(-1 + (0 \times x_1) + (0 \times x_2) + (1 \times x_1^2) + (1 \times x_2^2))$$

$$h_{\theta}(x) = g(-1 + x_1^2 + x_2^2)$$

بر اساس محاسبات بالا روابط زیر مفروض است.

$$\theta^T x \geq 0 \Rightarrow -1 + x_1^2 + x_2^2 \geq 0$$

$$x_1^2 + x_2^2 \geq 1$$



شکل (۳): مرز تصمیم با درجه چند جمله‌ای و غیر خطی.

وهمچنین توجه نمایید که درجه چندجمله‌ای بر حسب مسائل مختلف می‌تواند در درجات خیلی بالاتری باشد و مسئله از ابعاد بسیار بالایی برخوردار باشد و محاسبه  $h_{\theta}(x)$  بسیار پیچیده‌تر از مثالی باشد که مشاهده نمودید.

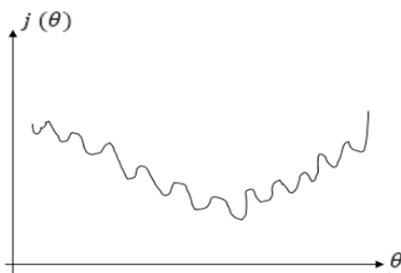
### ۳-۵- تابع هزینه

همان‌گونه که در رگرسیون خطی گفته شد تابع هزینه  $J(\theta)$  باید به کمترین مقدار ممکن و مناسب برسد تا الگوریتم نتایج مناسبی را ارائه دهد. رابطه تابع هزینه برای رگرسیون خطی به شکل زیر بود.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2 \quad \text{رابطه (۵)}$$

اما در رگرسیون لاجستیک به دلیل اینکه جواب  $h_{\theta}(x)$  با استفاده از تابع سیگموئید  $g(z)$  به دست می‌آید پس در صورت استفاده از رابطه (۵) که برای رگرسیون خطی استفاده می‌کردیم، تابع هزینه به صورت محدب<sup>۱</sup> نخواهد

بود و رسیدن گرادیان نزولی به نقطه کمینه تابع غیرممکن به نظر می‌آید. به نمودار شکل (۷-۳) توجه نمایید.



شکل (۷-۳): تابع هزینه غیرمحدب.

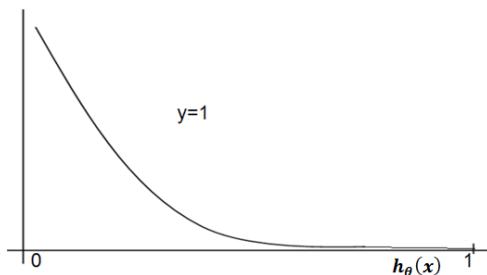
توجه شود که  $j(\theta) = \frac{1}{1+e^{-z}}$  یک تابع غیرمحدب می‌باشد و در این حالت تابع در نقاط کمینه محلی زیادی گیر می‌کند و به حالا بهینه نخواهد رسید. برای همین برای محاسبه تابع هزینه در رگرسیون لاجستیک از قوانین دیگری پیروی می‌کنیم.

### ۳-۵-۱- تابع هزینه در رگرسیون لاجستیک

رابطه تابع هزینه در رگرسیون لاجستیک به صورت زیر می‌باشد که بر آن اساس ثابت می‌کنیم که این تغییر موجب می‌شود تا تابع هزینه جدید محدب شود.

$$\text{cost}(h_\theta(x), y) = \begin{cases} -(\log(h_\theta(x))) & \text{if } y = 1 \\ -(\log(1 - h_\theta(x))) & \text{if } y = 0 \end{cases} \quad \text{رابطه (۶)}$$

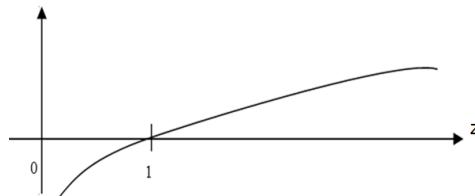
رابطه (۶) نشان می‌دهد که اگر  $h_\theta(x)$  مقدارش از یک حد آستانه‌ای مانند ۰.۵ کمتر باشد  $y=0$  شده و در اصل تابع ما به صورت  $-(\log(1 - h_\theta(x)))$  می‌شود و اگر  $h_\theta(x)$  از همان مقدار حد آستانه بیشتر باشد در این وضعیت  $y=1$  و تابع ما همان  $-(\log(h_\theta(x)))$  خواهد بود که می‌توانید نمودار تابع را در شرایطی که  $y=1$  است، در شکل (۸-۳) مشاهده نمایید.



شکل (۸-۳): نمودار تابع هزینه برای رگرسیون لاجستیک.

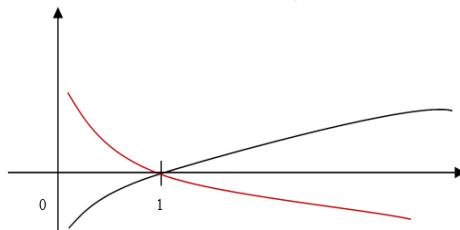
حال به نحوه به وجود آمدن نمودار شکل (۸-۳) توجه نمایید.

براساس تعریف لگاریتم برای  $\log(z)$  نمودار شکل (۹-۳) را در زیر داریم که نشاندهنده نمودار لگاریتم تابع روی متغیر  $z$  است.



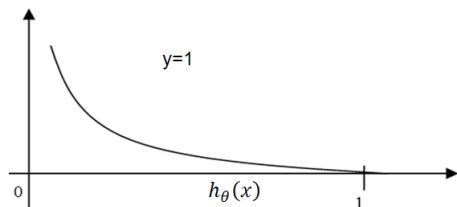
شکل (۹-۳): نمودار لگاریتم.

همان‌گونه که در رابطه (۶) توضیح دادیم ما برای تابع هزینه خود  $-\log(z)$ - را داریم که در این حالت با توجه به شرط موجود با قرینه شکل (۹-۳) روبه رو خواهیم شد که نمودار آن را در شکل (۱۰-۳) مشاهده می‌کنید.



شکل (۱۰-۳): قرینه نمودار لگاریتم.

و به دلیل آنکه منطقه مورد نظر ما فاصله بین صفر و یک می‌باشد پس محل مورد نظر از نمودار شکل (۱۰-۳) به صورت شکل (۱۱-۳) که در زیر ملاحظه می‌نمایید؛ می‌باشد که همان نمودار اصلی تابع ما برای  $(\log(h_\theta(x)))$ - می‌باشد.



شکل (۱۱-۳): نمودار تابع اصلی تابع هزینه در رگرسیون لاجستیک.

و همان‌طور که در بخش‌های قبلی گفته شد برای رسیدن به نقطه کمینه، تابع هزینه باید به صفر برسد تا محدب شود.

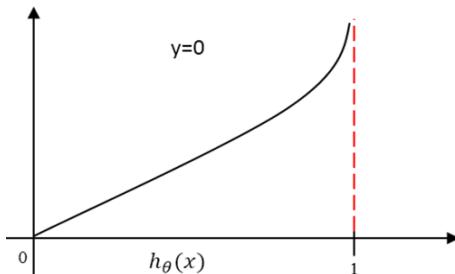
### ۳-۵-۲- نقطه کمینه تابع هزینه

اگر  $y=1$  و  $h_\theta(x) = 1$  باشد تابع هزینه به صفر مایل شده و در نتیجه به کمترین مقدار خود می‌رسد و یا صفر می‌شود ( $\text{cost}=0$ ). این شرایط در شکل (۱۱-۳) قابل ملاحظه می‌باشد.

### ۳-۵-۳- نقطه بیشینه تابع هزینه

چنانچه گفته شد اگر  $y=0$  و  $h_\theta(x) = 0$  باشد تابع هزینه به سمت مثبت بی‌نهایت خواهد رفت. رابطه تابع هزینه همان‌طور که کامل آن در رابطه (۶) آورده شده است به صورت رابطه (۷) و نمودار آن به صورت شکل (۱۲-۳) می‌باشد.

$$cost(h_\theta(x), y) = -(\log(1 - h_\theta(x))) \quad (رابطه ۷)$$



شکل (۱۲-۳): تابع هزینه برای حالت  $y=0$  به سمت مثبت بینهایت میل می‌کند.

### ۳-۶- گرادیان نزولی برای رگرسیون لاجستیک

در بخش‌های قبلی گرادیان نزولی را برای رساندن تابع هزینه به وضعیت کمینه ممکن مورد استفاده قرار داده‌ایم در این بخش نیز خواهید دید که چگونه می‌توان از گرادیان نزولی برای رگرسیون لاجستیک استفاده کرد.

### ۳-۷- رابطه تابع هزینه در رگرسیون لاجستیک

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m cost(h_\theta(x^{(i)}), y^{(i)})^2 \quad (رابطه ۸)$$

$$cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases} \quad (رابطه ۹)$$

براساس رابطه (۹) با توجه به اینکه  $y$  همیشه یا یک و یا صفر می‌باشد پس رابطه (۱۰) را به عنوان تابع هزینه کلی خواهیم داشت که یک تابع محدب است و از مجموع دو تابع نشان داده شده در شکل (۱۱-۳) و (۱۲-۳) به دست آمده است.

$$cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x)) \quad (رابطه ۱۰)$$

اثبات رابطه (۱۰) به صورت زیر می‌باشد.

با توجه به شرط موجود در رابطه (۹) که معرف دو حالت برای  $y$  است یعنی حالتی که در آن  $y=0$  و حالتی دیگر که در آن  $y=1$  می‌باشد، رابطه (۱۰) را محاسبه و مورد تحلیل قرار می‌دهیم.

برای حالتی که  $y=1$  را در نظر داشته باشیم محاسبات زیر را خواهیم داشت.

$$cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

$$cost(h_\theta(x), y = 1) = -1 \log(h_\theta(x)) - (1 - 1) \log(1 - h_\theta(x))$$

$$\text{cost}(h_\theta(x), (y = 1)) = -1 \log(h_\theta(x)) - (0) \log(1 - h_\theta(x)) = -\log(h_\theta(x))$$

$$\text{cost}(h_\theta(x), (y = 1)) = -\log(h_\theta(x))$$

برای حالتی که  $y=0$  را در نظر داشته باشیم محاسبات زیر را خواهیم داشت.

$$\text{cost}(h_\theta(x), (y)) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

$$\text{cost}(h_\theta(x), (y = 0)) = -(0) \log(h_\theta(x)) - (1 - 0) \log(1 - h_\theta(x))$$

$$\text{cost}(h_\theta(x), (y = 0)) = -\log(1 - h_\theta(x))$$

رابطه اصلی تابع هزینه برای رگرسیون لاجستیک به صورت زیر می‌باشد.

در نتیجه تابع هزینه  $J(\theta)$  برای رگرسیون لاجستیک به صورت زیر می‌باشد.

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{cost}(h_\theta(x^{(i)}), y^{(i)}) = \\ &= \frac{1}{m} [\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] \end{aligned} \quad (11)$$

حال با توجه به رابطه (11) به الگوریتم زیر برای رساندن تابع هزینه به نقطه کمینه تابع با استفاده از گرادیان نزولی توجه نمایید.

### ۱-۳-۱- الگوریتم گرادیان نزولی برای رگرسیون لاجستیک

باید  $J(\theta)$  به کمینه برسد

تکرار }

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m h_\theta(x^{(i)}) - y^{(i)} x_j^{(i)}$$

(به روزرسانی هم‌زمان و موازی پارامترها)

{

نکته: الگوریتم فوق شبیه الگوریتمی می‌باشد که ما در رگرسیون خطی مورد استفاده قرار دادیم و باید توجه کرد که با اینکه از نظر ظاهری یکسان به نظر می‌رسند ولی در این الگوریتم  $h_\theta(x)$  براساس تابع سیگموئید محاسبه می‌شود (طبق آنچه که در بخش‌های قبل توضیح داده شد) و کاملاً تفاوت دارد.

به عبارتی در رگرسیون خطی  $h_\theta(x) = \theta^T X$  به صورت  $h_\theta(x) = \theta^T X$  بود ولی در این الگوریتم که برای رگرسیون لاجستیک می‌باشد برای  $h_\theta(x)$  از تابع سیگموئید زیر استفاده می‌شود.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

### ۹-۳- رگرسیون لاجستیک چند کلاسه

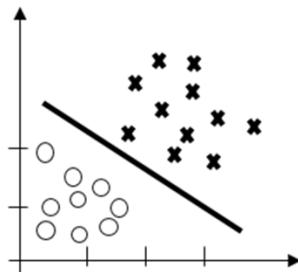
مسائل مربوط به دسته‌بندی در یادگیری با نظارت تنها برای دو کلاس یا ایجاد دو دسته نمی‌باشد و می‌تواند چند کلاس برای حل یک مسئله مورد نیاز باشد.

فرض کنید می‌خواهیم پست‌های الکترونیکی<sup>۱</sup> دریافتنی را در دسته‌های زیر قرار دهیم.  
ایمیل کاری. ۲- ایمیل خانوادگی. ۳- ایمیل‌هایی از دوستان. ۴- ایمیل‌های سرگرمی.

در این حالت ما به چهار کلاس برای دسته‌بندی ایمیل‌ها احتیاج داریم. به جدول زیر توجه نمایید.

Y=1	Y=2	Y=3	Y=4
ایمیل‌های سرگرمی	ایمیل از دوستان	ایمیل خانوادگی	ایمیل کاری

چنانچه ملاحظه نمودید در بخش‌های قبل برای دو کلاس دسته‌بندی انجام شد که در آن برای غده سلطانی دو دسته بدخیم و خوش‌خیم را با استفاده از مقادیر دودوبی<sup>۲</sup> که همان صفر و یک بود را در نظر گرفتیم که نمودار مرز تصمیم به صورت شکل (۹-۳) بود.



شکل (۹-۳): نمایش عملکرد یک کلاسه‌بند باینری با حالات عملکرد درست و نادرست.

در روش چند کلاسه یکی در مقابل همه<sup>۳</sup> از  $k$  دسته‌بند دو کلاسه می‌توان استفاده کرد. حال داده از نمونه دیده نشده به همه این  $k$  دسته‌بند اعمال می‌شود و خروجی این دسته‌بندها بررسی می‌شوند تا کلاس داده مشخص شود، بسته به نوع دسته‌بند می‌توان دسته‌بندی که بالاترین احتمال تعلق به خود را برای این داده آزمون داشت به عنوان کلاس داده آزمون انتخاب کرد.

در روش یکی در برابر یکی<sup>۴</sup>، از تعداد  $(k-1)/2$  دسته‌بندی کننده باینری استفاده می‌شود که یک دسته را از یک دسته دیگر جدا می‌کند سپس خروجی‌های همه دسته‌بندها ترکیب و رأی‌گیری<sup>۵</sup> می‌شوند تا کلاس داده اعمال شده (ازمجموعه داده آزمون) مشخص شود.

- 
1. Email
  2. Binary
  3. One-versus-all
  4. One-versus-one
  5. Voting

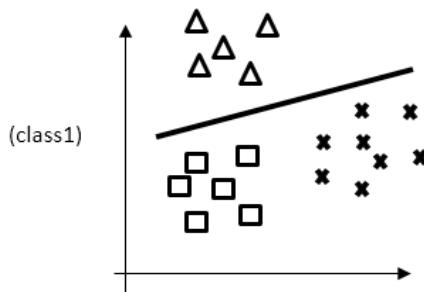
در روش یکی در مقابل همه، فرض کنید مجموعه داده‌های برحسب خوردهای داریم و می‌خواهیم آنها را در دسته‌های خود قرار دهیم. داده‌های مربوط به هر دسته را با علامت‌های زیر نشان خواهیم داد.

$\times = \text{class3}$	$\blacksquare \blacksquare = \text{class2}$	$\Delta \Delta = \text{Class1}$
--------------------------	---	---------------------------------

توجه شود که در مسائل واقعی تفاوتی ندارد که اول برای پیدا کردن کدام دسته اقدام می‌کنیم. در اینجا ما ابتدا برای پیدا کردن داده‌هایی که به صورت مثلث می‌باشند اقدام می‌کنیم و در پردازش داده‌های کل وقتی به یک داده با ویژگی داده‌هایی مانند مثلث رسیدیم آن را به صورت صحیح<sup>۱</sup> یا یک و مابقی را صفر یا نادرست<sup>۲</sup> در نظر می‌گیریم. به مثال زیر برای دسته‌بندی چند کلاسه توجه نمایید.

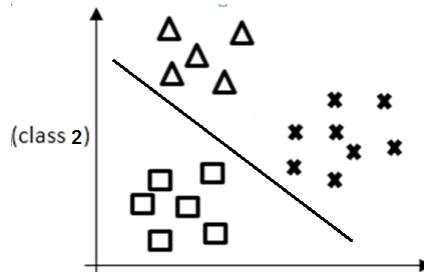
مثال) اگر  $h_{\theta}^{(i)}(x) = p(y = i|x; \theta)$  و  $i=1, 2, 3$  باشد به نمودارهای مربوط به دسته‌بندی‌های مختلف و نحوه ترسیم مرز تصمیم آنها توجه نمایید.

$$\text{الف) } i=1 \text{ و } h_{\theta}^{(1)}(x) = p(y = 1|x; \theta)$$



شکل (۱۴-۳): دسته‌بندی از روی تفکیک کلاس مثلث‌ها از بقیه.

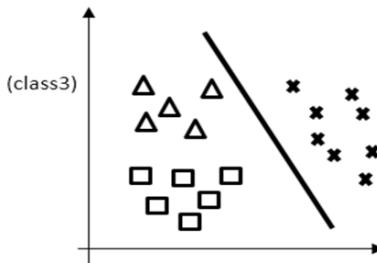
$$\text{ب) } i=2 \text{ و } h_{\theta}^{(2)}(x) = p(y = 2|x; \theta)$$



شکل (۱۵-۳): دسته‌بندی از روی تفکیک کلاس مربع‌ها از بقیه.

$$\text{پ) } i=3 \text{ و } h_{\theta}^{(3)}(x) = p(y = 3|x; \theta)$$

1. True
2. False



شکل (۳-۱۶): دسته‌بندی از روی تفکیک کلاس ضربدرها از بقیه.

همان‌گونه که در مثال بالا مشاهده نمودید به تفکیک هر کلاس از بقیه در اصل تمامی داده‌ها را در کلاس‌های متفاوت دسته‌بندی کردایم.

### ۳-۱۰- پیاده‌سازی یک رگرسیون لاجستیک با استفاده از زبان R

در برنامه زیر با استفاده از رگرسیون لاجستیک یک تخمین و دسته‌بندی با استفاده از مجموعه داده iris انجام شده که به صورت کلی نحوه تحلیل مجموعه داده و تفکیک داده‌های موجود به دسته‌های مربوط به خود را نشان می‌دهد.

مجموعه داده iris شامل ۵۰ نمونه از هر یک از سه گونه گل زنبق (Iris setosa, Iris virginica, Iris versicolor) است که در مجموع یک مجموعه داده با ۱۵۰ نمونه را تشکیل می‌دهد که از هر نمونه چهار ویژگی بررسی گردیده است و این ویژگی‌ها با ترکیب ویژگی‌های طول و عرض کاسبرگ و گلبرگ به وجود می‌آید. مجموعه داده iris یک مجموعه داده استاندارد و معروف است که ما هم در این فصل و هم در برخی از فصل‌های دیگر از این مجموعه داده استفاده نموده‌ایم.

در دو خط ابتدایی برنامه زیر بسته نرم‌افزاری موردنیاز نصب و مجموعه داده مورد نیاز بارگذاری شده است. با توجه به اینکه همواره برای بارگذاری و استفاده از یک بسته نرم‌افزاری باید از تابع () library استفاده شود، در صورتی که با اجرای این تابع خطای را مبنی بر عدم وجود بسته نرم‌افزاری دریافت نمودید، باید بسته نرم‌افزاری را قبل از بارگذاری نصب کنید و سپس با استفاده از تابع مربوطه بارگذاری کنید.

```
> install.packages("GGally")
> library(ggplot2); library(GGally)
> library(datasets)
```

پس از بارگذاری بسته نرم‌افزاری datasets به راحتی می‌توانید با دستور زیر مجموعه داده iris را در درون متغیر iris\_データ ذخیره کنید تا در برنامه از آن استفاده نمایید.

```
> iris_data<- iris
```

خط برنامه زیر یک تابع است که به صورت پیشفرض ۶ ردیف از مجموعه داده را با اسمی متغیرها یا ویژگی‌های آن به نمایش می‌گذارد. نکته بسیار مهم در این خط برنامه این است که به دلیل امکان تغییر نام متغیرها در نسخه‌های مختلف یک مجموعه داده، می‌توانید از اسمی دقیق درون مجموعه داده مانند `class` که نشان‌دهنده گونه‌های مختلف است آگاهی پیدا کنید.

```
> head(iris_data)
```

	Sep.Length	Sep.Width	Pet.Length	Pet.Width	Class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

و همچنین با استفاده از دستور زیر می‌توانید اطلاعات بیشتری را از مجموعه داده در قالب یک data frame به دست آورید. Data frame زیر نشان‌دهنده ۱۵۰ نمونه با ۵ متغیر موجود در مجموعه داده است. چهار متغیر ابتدایی نشان‌دهنده اطلاعاتی درباره خصوصیات گیاه زنبق است و متغیر آخر مشخص‌کننده نام گونه‌های مختلف گیاه زنبق می‌باشد.

```
> str(iris_data)
```

```
'data.frame': 150 obs. of 5 variables:
 $ Sep.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sep.Width: num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Pet.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Pet.Width: num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Class : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

با استفاده از دستور زیر می‌توانید اسمی موجود در متغیر `class` را که همان نام گونه‌های مختلف گیاه زنبق است را به دست آورید.

```
> levels(iris_data$Class)
```

```
[1] "setosa" "versicolor" "virginica"
```

در دستور زیر می‌توان تعداد مقادیر NA را در کل مجموعه داده به دست آوریم که توسط `is.na()` مقادیر NA ها مشخص شده و با تابع `sum()` مجموع آن محاسبه می‌گردد.

```
> sum(is.na(iris_data))
```

```
[1] 0
```

حال با توجه به اینکه اطلاعات کافی در رابطه با مجموعه داده به دست آورده بود از برنامه اقدام به ایجاد رگرسیون لاجستیک می‌نماییم. هدف ایجاد یک دسته‌بند باینری است که تمایز بین دو گروه از انواع گیاه زنبق را مشخص کند. پس ما در اصل به دو گونه از سه گونه موجود در مجموعه داده نیاز داریم که با استفاده از دستور زیر از ۱ تا ۱۰۰ رديف اول مجموعه داده را که به دو گونه موجود تعلق دارد، انتخاب می‌کنیم.

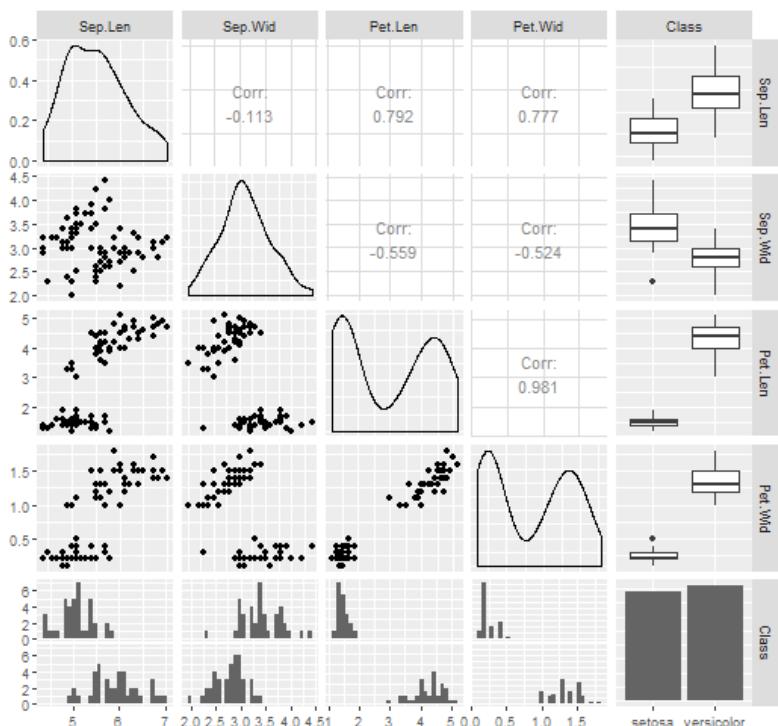
```
> iris_data<-iris_data[1:100,]
```

در خط برنامه زیر با استفاده ازتابع sample() به صورت تصادفی از ۱۰۰ داده موجود ۸۰ نمونه را به داده آموزشی و ۲۰ نمونه را به داده آزمون اختصاص می‌دهیم.

```
> set.seed(100)
> samp<-sample(1:100,80)
> iris_train<-iris_data[samp,]
> iris_test<-iris_data[-samp,]
```

با استفاده از تابع زیر که از بسته نرم‌افزاری GGallery است نمودارهای مختلف مناسبی را برای اطلاع از وضعیت داده‌های آموزشی می‌توانید به دست آورید که صرفاً جهت مطالعه داده‌های آموزشی است و در صورت تمایل می‌توانید این خط از برنامه را استفاده نکنید.

```
> ggpairs(iris_train)
```



با استفاده از خطوط برنامه زیر داده‌ها را برای استفاده در رگرسیون لاجستیک آماده می‌کنیم. برای اینکه برنامه ساده باشد تا خواننده راحت‌تر آن را درک کند، تنها از یک متغیر استفاده می‌نماییم. دو متغیر  $x$  و  $y$  را ایجاد می‌کنیم که  $x$  مربوط به یک ویژگی یا متغیر و  $y$  مربوط به نام گونه گیاه زنبق است.

```
> y<-iris_train$Class; x<-iris_train$Sep.Length
```

در خط برنامه زیر مدل رگرسیون لاجستیک را با استفاده ازتابع  $(\text{glm})$  ایجاد می‌نماییم. هدف این است که هر کدام از نمونه‌های موجود با توجه به ویژگی  $\text{sep.length}$  مشخص شود که متعلق به کدام گونه موجود در  $\text{class}$  است. توجه نمایید به دلیل اینکه ما از روش کلاسه‌بندی باینری استفاده می‌کنیم اگر براساس ویژگی موجود هر نمونه داده شده، به یک دسته تعلق نداشت می‌توانیم آن را در دسته دیگر جای دهیم و با تابع  $(\text{summary})$  نحوه اجرای مدل رگرسیون لاجستیک را براساس داده‌های آموزشی می‌توانید مطالعه نمایید.

```
> glfit<-glm(y~x, family = "binomial")
> summary(glfit)
```

Call:

```
glm(formula = y ~ x, family = "binomial")
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.94538	-0.50121	0.04079	0.45923	2.26238

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-25.386	5.517	-4.601	4.20e-06 ***
x	4.675	1.017	4.596	4.31e-06 ***

Signif. Codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 110.854 on 79 degrees of freedom  
 Residual deviance: 56.716 on 78 degrees of freedom  
 AIC: 60.716

Number of Fisher Scoring iterations: 6

در این مرحله مدل ما براساس داده‌های آموزشی ایجاد شده و می‌توانیم داده‌های جدید را برای انجام عمل تخمین و دسته‌بندی مورد استفاده قرار دهیم. با استفاده از خط زیر یک  $\text{data frame}$  از داده‌های آزمون ایجاد می‌کنیم.

```
> newdata<- data.frame(x=iris_test$Sep.Length)
```

در خط زیر با استفاده از تابع predict() عمل تخمین و دسته‌بندی را انجام می‌دهیم.

```
> predicted_val<-predict(glfit, newdata, type="response")
```

در خط زیر نیز خروجی دسته‌بند را در قالب یک data frame در خروجی چاپ می‌نماییم. چنانچه مشاهده می‌نمایید از ردیف ۱ تا ردیف ۱۱ نمونه‌های موجود در ویژگی sept.len به دسته گونه setosa و از ۲۰ تا ۲۹ به دسته گونه versicolor تعلق بافته است.

```
> prediction<-data.frame(iris_test$Sep.Length, iris_test$Class)
```

```
> prediction
```

	iris_test\$Sep.Length	iris_test\$Class
1	5.1	setosa
2	4.7	setosa
3	4.6	setosa
4	5.0	setosa
5	4.6	setosa
6	4.3	setosa
7	4.6	setosa
8	5.2	setosa
9	5.2	setosa
10	5.0	setosa
11	5.0	setosa
12	6.6	versicolor
13	5.2	versicolor
14	5.8	versicolor
15	6.2	versicolor
16	6.6	versicolor
17	5.5	versicolor
18	6.3	versicolor
19	5.7	versicolor
20	5.7	versicolor

```
>
```

در صورت تمایل می‌توانید با استفاده از خط زیر و با تابع qplot() نتیجه تخمین را به صورت نموداری مشاهده نمایید. توجه نمایید استفاده از تابع round() در برنامه زیر موجب می‌شود تا داده‌ها یا به صفر و یا به یک نسبت داده شوند و اگر از آن تابع استفاده نشود نتیجه نمودار به صورت یک مدل فازی نمایش داده می‌شود.

```
> qplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab = 'Prediction using Logistic Reg.')
```

### ۱۱-۳- پیاده‌سازی یک رگرسیون لاجستیک با استفاده از زبان پایتون

الگوریتم زیر مراحل پیاده‌سازی یک رگرسیون لاجستیک را نشان می‌دهد که مجموعه داده iris را مورد استفاده قرار می‌دهد. مجموعه داده استفاده شده دارای برچسب می‌باشد و این دسته بند داده‌ها را به سه دسته جدا می‌سازد که هر دسته در نهایت با رنگی جدا در بقیه دسته‌ها مصور می‌شود. مجموعه داده Iris یعنی مجموعه داده استاندارد می‌باشد که داده‌های آن دارای برچسب می‌باشد و برای دسته‌بندی مورد استفاده قرار می‌گیرد. این مجموعه داده دارای داده‌های مربوط به مشخصات سه گونه گیاه Iris می‌باشد که هر گونه با طول و عرض گلبرگ با سایرین متمایز می‌گردد.

در چهار خط اول کتابخانه‌های مورد استفاده مشخص می‌شوند.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
```

در سه خط زیر مجموعه داده iris به برنامه افزوده می‌شود که تنها دو ویژگی اول موجود در مجموعه داده در اینجا مدنظر قرار دارد.

```
iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target
```

در خط زیر پارامترهای مشخص کننده دسته بند مشخص می‌گردد که شیء logreg را تولید نماید.

```
logreg = LogisticRegression(C=1e5, solver='lbfgs', multi_class='multinomial')
```

خط زیر با استفاده از logreg دسته بند را برای دریافت داده آموزشی و دسته‌بندی آنها ایجاد می‌نماید که مدل رگرسیون لاجستیک را ایجاد نماییم.

```
logreg.fit(X, Y)
```

در زیر تخمین نهایی دسته داده‌های آزمایشی با استفاده از logreg.predict() انجام می‌شود و نمودار خط تصمیم ترسیم می‌شود. برای این منظور برای هر نقطه داده موجود در نمودار یک رنگ مشخص کننده دسته مورد نظر اختصاص داده می‌شود.

```
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
h = .02
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
```

نتایج بالا را با استفاده از دستورات زیر در یک نمودار رنگی چاپ می‌نماییم که هم زمینه نمودار به صورت سه تکه رنگی جدا مصور می‌شود و هم نقاط مختلف در رنگ‌های متفاوت ظاهر می‌گردند.

سه خط زیر زمینه تصویر را مشخص می‌نماید.

```
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
```

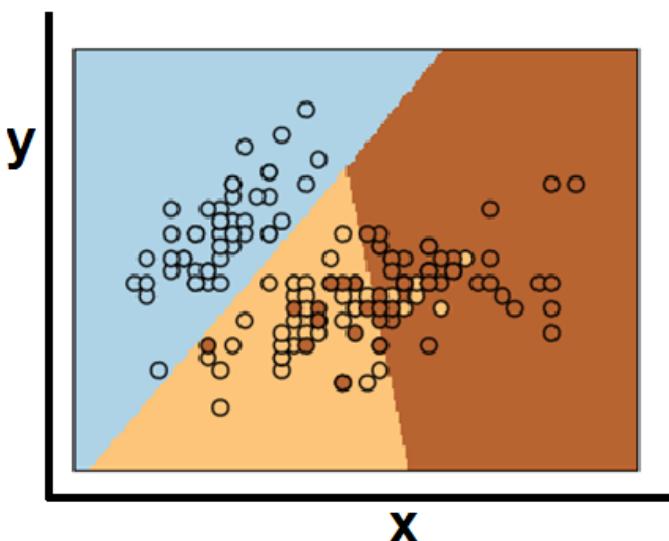
خطوط زیر نقاط داده‌ای را به صورت رنگی در می‌آورند.

```
plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
plt.xlabel('X')
plt.ylabel('Y')
```

خطوط زیر نمودار نهایی را در خروجی چاپ می‌نمایند.

```
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())
plt.show()
```

نمودار زیر خروجی نهایی رگرسیون لاجستیک می‌باشد که داده‌ها را در دو محور x و y نشان می‌دهد که در سه دسته با رنگ‌های مختلف مشخص شده‌اند.





فصل

## راهکارهای جهت بهبود طراحی الگوریتم‌های یادگیری ماشین

## ۴-۱- راهکارهایی جهت بهبود طراحی الگوریتم‌های یادگیری ماشین

در این فصل با مبحث بهبود الگوریتم‌های یادگیری آشنا خواهید شد. در هر الگوریتم مبتنی بر یادگیری چنانچه در مطالب فصل‌های قبل مشاهده نمودید رسیدن الگوریتم به یک حالت بهینه یکی از اهداف اصلی طراحی الگوریتم برای حل مسائل مختلف است. در این راستا آگاهی از موانع سر راه پیاده‌سازی بهینه الگوریتم و آگاهی از روش‌های تنظیم کردن در این جهت، یکی از موارد اجتناب‌ناپذیر به حساب می‌آید.

## ۴-۲- تعمیم‌پذیری الگوریتم

اگر الگوریتم ما قرار باشد تا در مرحله آزمایش دقیقاً با مجموعه داده‌هایی که در مرحله آموزش استفاده شد کار کند و در صورتی که خطای آموزشی به صفر نزدیک شود، در این حالت نتیجه آزمایش الگوریتم بسیار خوب بوده و الگوریتم با درجه بالایی قابل اطمینان خواهد بود. اما معمولاً هدف الگوریتم‌های یادگیری این است که با یادگیری داده‌های آموزشی به یک مدلی برسند که بتواند در شرایط وجود داده‌های جدید، نتایج قابل قبولی را ارائه دهنند. یعنی اگر در مرحله آزمایش داده‌ها دیگر همان داده‌های مورد استفاده در مرحله آموزش نباشند، بتواند مناسب عمل کند. برای این موضوع هرگز نباید خطای الگوریتم را در مرحله آموزش به صفر مطلق رساند و باید به یک حالت متعادلی رسید که به این خاصیت، تعمیم‌پذیری<sup>۱</sup> الگوریتم گویند. برای رساندن الگوریتم به این حالت متعادل باید از یک سری قوانین پیروی کرد که الگوریتم بتواند به حالت بهینه مورد نیاز برسد.

## ۴-۳- عوامل رسیدن الگوریتم به حالت بهینه

حالت بهینه به وضعیتی گفته می‌شود که در آن الگوریتم یادگیری در مرحله آزمایش بتواند به یک مدل مناسب دست پیدا کند به گونه‌ای که نتایج الگوریتم در مرحله آزمون بسیار مطلوب باشد. عوامل بسیاری در این موضوع دخالت دارند، از جمله می‌توان به انتخاب الگوریتم مناسب برای حل یک مسئله در حوزه خاص، ایجاد مدلی مناسب برای الگوریتم مورد استفاده، انتخاب ویژگی‌های مناسب، استفاده از نمونه‌های آموزشی کافی برای آموزش الگوریتم، ایجاد روش مناسبی برای ارزیابی حین آموزش الگوریتم، به عنوان مهم‌ترین موارد اشاره کرد.

## ۴-۴- ظرفیت یک فرضیه

یکی از روش‌های تعیین ظرفیت یک خانواده از فرضیه با استفاده از روش‌های فرمال روش بعد VC<sup>۲</sup> می‌باشد. اگر مجموعه‌ای داشته باشیم که دارای N نمونه باشد و این N نمونه در دو کلاس مثبت و منفی هر وضعیتی را بتواند به خود بگیرند این نقاط<sup>۳</sup> 2<sup>N</sup> حالت مختلف خواهند داشت. بنابراین 2<sup>N</sup> مسئله یادگیری مختلف می‌توانیم داشته باشیم. اگر برای هر یک از این مسائل بتوانیم فرضیه‌ای h<sup>eH</sup> یک خانواده از فرضیه است، h<sup>eH</sup> یکی از حالات این خانواده است) پیدا کنیم که بتواند نمونه مثبت<sup>۴</sup> و منفی را از هم جدا کند در این حالت H می‌تواند N نقطه را خرد<sup>۵</sup> کند. یعنی هر مسئله یادگیری قابل تعریف توسط N نمونه بدون خطا توسط یک فرضیه h که از خانواده

1. Generalization

4. Shatter

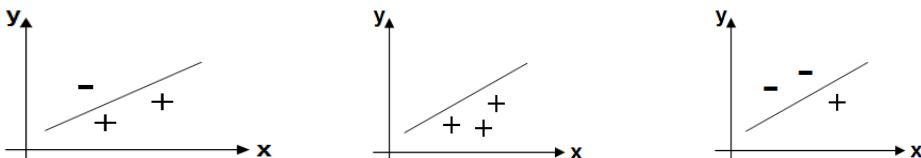
2. Vapnic cheronekic dimension

3. Positive example

می‌باشد می‌تواند یاد گرفته شود. بیشینه‌ترین تعداد نقاطی که می‌تواند با  $H$  خرد شود بعد  $VC$  مربوط به  $H$  یا  $VC(H)$  که اصطلاحاً ظرفیت  $H$  گفته می‌شود، می‌باشد.

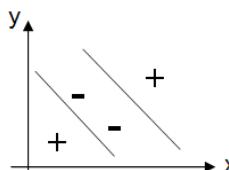
به عنوان مثال برای یک مجموعه داده با سه نقطه، خط  $H$  در هر حالت ممکن می‌تواند این سه نقطه با هر برچسب مثبت یا منفی را جدا کند. بنابراین ظرفیت  $VC$  یک فرضیه خط راست و برابر با ۳ می‌باشد که با  $H$  نشان داده می‌شود.

چند نمونه از حالات سه نقطه که با خط راست جدا می‌شوند؛ در شکل زیر آورده شده است.



شکل (۴-۴): جداسازی سه داده برچسبدار در حالات قرارگیری مختلف با خط فرضیه راست.

ولی مانند شکل زیر یک خط راست ممکن است قادر به جداسازی چهار نقطه با هر حالت ممکن را نداشته باشد.



شکل (۴-۵): نمایی از ظرفیت با بعد  $VC$  که عدم جدایی چهار نقطه توسط فرضیه یک خط تنها را نشان می‌دهد.

روش  $VC$  یک روش فرمال برای تعیین ظرفیت یک فرضیه می‌باشد که هرچقدر این فرضیه شکل پیچیده‌تری داشته باشد و انعطاف‌پذیرتر باشد قاعده‌تاً ظرفیت آن نیز بیشتر است. در حالت حدسی بدون استفاده از روش‌های فرمال مانند  $VC$  هرچقدر تعداد پارامترهای یک فرضیه بیشتر باشد (مانند تعداد  $\theta$ ‌ها در رگرسیون) درجات بالاتر یک فرضیه چندجمله‌ای به مفهوم ظرفیت بیشتر است یعنی ظرفیت یک فرضیه چندجمله‌ای با درجه ۴ بالاتر از ظرفیت یک فرضیه با درجه ۳ است.

## ۴-۵- ایجاد مدل مناسب برای الگوریتم یادگیری

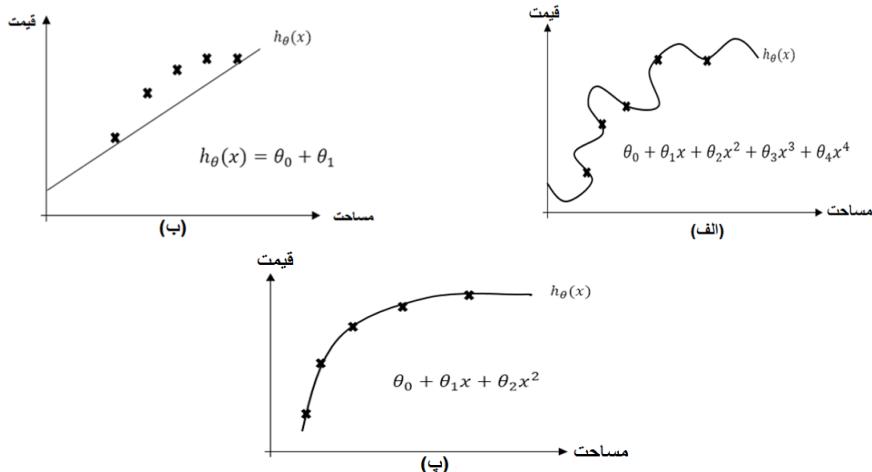
ایجاد یک مدل مناسب برای الگوریتم یادگیری مانند تنظیم مناسب پارامترها برای محاسبهتابع هزینه و یا الگوریتم گرادیان نزولی، چنانچه در فصل‌های قبل توضیحاتی در اینباره داده شد همواره بسیار مهم بوده است. اما نکته برجسته‌تر این است که وقتی ما درباره مدل‌های پیش‌بینی‌کننده صحبت می‌کنیم، خطای پیش‌بینی وابستگی بسیاری به دو موضوع مهم دارد. یکی مربوط به خطایی است که به دلیل بایاس بالا<sup>۱</sup> ایجاد می‌شود و دیگری خطای ناشی از واریانس بالا<sup>۲</sup>. برای اینکه خطای الگوریتم را پایین آوریم باید بتوانیم بایاس و واریانس را به میزان مناسب کاهش دهیم که این امر باعث می‌شود تا الگوریتم از بیش برازش<sup>۳</sup> و یا کم برازش<sup>۴</sup> اجتناب کند.

- |                  |                  |
|------------------|------------------|
| 1. High bias     | 3. Over fitting  |
| 2. High variance | 4. Under fitting |

## ۴-۶-۴- مفهوم بیش برآزش، بایاس و واریانس

با توجه به توضیحات داده شده درباره تأثیر دقیق بیش از حد و یا کمتر از حد مورد نیاز الگوریتم، با یک مثال درباره تخمین قیمت خانه با روش رگرسیون خطی، مفاهیم بایاس بالا و واریانس بالا، بیش برآزش و کم برآزش را معرفی می‌نماییم.

مثال) فرض کنید با رگرسیون خطی قصد داریم قیمت خانه را تخمین بزنیم. به نمودارهای شکل (۴-۳) توجه نمایید که سه حالت ممکن قرارگیری خط فرضیه را نشان می‌دهد.



شکل (۴-۳): حالات مختلف قرارگیری خط فرضیه در میان یک مجموعه داده مشترک.

## ۴-۶-۱- بیش برآزش و واریانس بالا

وقتی ما با شرایطی رو برو هستیم که در آن تعداد ویژگی‌ها بسیار بالا است ولی تعداد داده‌های ما برای آن کافی نیست و همچنین مدلی را ایجاد کرده‌ایم که از درجه چندجمله‌ای بالایی برخوردار است تا بتواند تمامی ویژگی‌ها را یاد بگیرد به دلیل خیلی زیاد بودن تعداد ویژگی‌ها مانند شکل (الف) که فرضیه ما از پیچیدگی بالایی برخوردار است و با دقیق بسیار بالایی تمام ویژگی‌ها را پوشش داده و  $h_{\theta}(x)$  با استفاده از چندجمله‌ای‌های درجه سوم و چهارم تلاش کرده تا تمامی حالات را در نظر گرفته و آنها را پوشش دهد، در این حالت پدیده بیش برآزش یا واریانس بالا اتفاق افتاده است. یعنی داده‌های آموزشی کاملاً یاد گرفته شده‌اند و خط نزدیک به صفر است که می‌تواند موجب بالا بودن خطای الگوریتم در مرحله آزمون شود.

از دیدگاه ظرفیت حدسی یک فرضیه نیز می‌توان موارد ذکر شده در فوق را در ارتباط با بایاس و واریانس فرضیه نیز مطرح کرد. به عبارتی هرچقدر خانواده فرضیه  $H$ ، غنی‌تر و انعطاف‌پذیرتر باشد ظرفیت آن بیشتر است. به عنوان مثال در رگرسیون خطی با توجه به خانواده  $H$  اگر از نوع یک خط مستقیم باشد یا اصطلاحاً  $H$  انعطاف‌پذیر نیست و ضعیف است و ظرفیت آن پایین است. در شکل (۴-۳)(ب) موضوع ظرفیت را می‌توان در قالب واریانس

و بایاس نیز مشاهده نمود یک فرضیه ضعیف مانند خط راست بایاس زیاد و واریانس کم دارد. و هرچه فرضیه منحنی‌تر و انعطاف‌پذیرتر باشد بایاس کمتر و واریانس آن بیشتر است.

در شکل (۳-۴) (الف) مصالحه در این حالت بین ظرفیت و بایاس و واریانس باید صورت گیرد. بحث مربوط به تغییرات بایاس و واریانس در همین فصل از دیدگاه‌های دیگر شده و خواهد شد. مصالحه بهینه بستگی به اندازه مجموعه داده (N) دارد و معمولاً هرچقدر اندازه مجموعه داده افزایش یابد (N بزرگتر) واریانس کاهش و بایاس هم نسبتاً کاهش می‌یابد. در حالی که ظرفیت افزایش می‌یابد؛ یعنی حجم داده هم نقش اساسی در این مصالحه دارد.

## ۴-۶-۲- کم برآش و بایاس بالا

اگر خط فرضیه به گونه‌ای قرار گیرد که نتواند ویژگی‌ها را با داده‌های کافی موجود برای آنها به اندازه مناسب و کافی پوشش دهد عملأ خط فرضیه در وضعیت نامطلوب قرار دارد، پس بر این اساس خطا بالا خواهد بود که به این حالت کم برآش گویند. در شکل (۳-۴)(ب) خط فرضیه  $h_{\theta}(x)$  یک خط راست می‌باشد که اصلاً نتوانسته درصد قابل قبولی از داده‌ها را پوشش دهد و به نظر می‌رسد که  $\theta_0 + \theta_1 x$  کافی نبوده و دقت خط فرضیه ما بسیار پایین می‌باشد و این حالت چنانچه گفته شد به کم برآش و یا بایاس بالا معروف است در نتیجه الگوریتم در مرحله آزمون می‌تواند با خطای بالای مواجه شود. به صورت کلی در پدیده کم برآش، درجه چندجمله‌ای معمولاً پایین‌تر از حد نیاز است و درصد خطا زیاد است.

## ۴-۶-۳- حالت مناسب از نظر بایاس و واریانس

در شکل (۳-۴)(پ) خط فرضیه ما به نظر مناسب می‌آید و چنانچه مشاهده می‌نمایید هم داده‌ها را پوشش می‌دهد و  $h_{\theta}(x)$  از پیچیدگی خیلی بالایی برخوردار نیست که این شرایط مناسبی را در مرحله آزمون الگوریتم رقم خواهد زد.

باید توجه داشته باشید که در طراحی مدل، باید خط  $h_{\theta}(x)$  طوری رسم شود که بتواند تعمیم‌پذیری را انجام دهد زیرا اگر در مرحله آموزش، داده‌ها با دقت خیلی بالا آموزش داده شده باشند اصطلاحاً الگوریتم داده‌های آموزشی را حفظ کرده<sup>1</sup> است و در این حالت وقتی در مرحله آزمایش با شرایطی که مقداری داده‌ها کاملاً متفاوت هستند یعنی الگوریتم این داده‌ها را در مرحله آموزش ندیده است دیگر ممکن است نتواند درست عمل کند و خطا در مرحله آزمایش بسیار بالا خواهد بود.

به عبارتی دیگر ممکن است داده‌های جدیدی در مرحله آزمایش به الگوریتم داده شود که در مرحله آموزش آنها را تجربه نکرده است و از آنجایی که در مرحله آموزش به مدل گفته شده که با دقت بسیار زیاد تمامی داده‌ها را یاد بگیرد و در مرحله آزمایش داده‌های جدید بسیاری وجود دارد که خط  $h_{\theta}(x)$  نمی‌تواند آنها را پوشش دهد پس درصد خطا بسیار بالا خواهد بود.

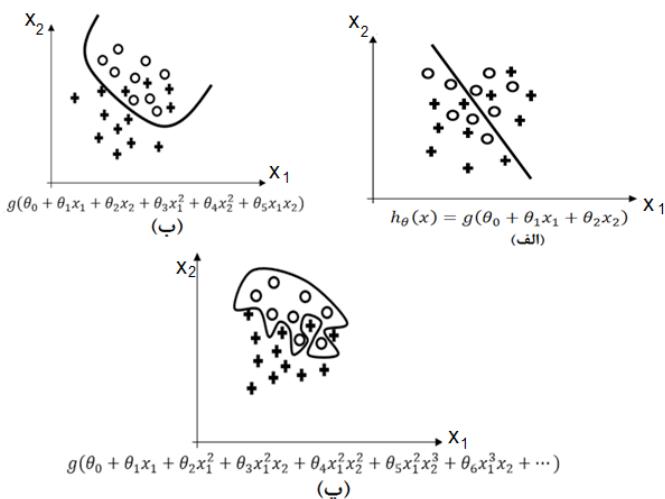
1. overtrain

## ۴-۷- نقش انتخاب ویژگی‌ها

در برخی از مسائل به دلیل پیچیدگی بالای الگوریتم نتایج مناسبی به دست نمی‌آید برای مقابله با این اتفاق می‌توان با حذف برخی از ویژگی‌ها که موجب کاهش تعداد ویژگی‌ها می‌شود از درجه پیچیدگی الگوریتم کم کرد و به حالت بینه رسانید. ولی توجه شود که در این کار ممکن است تعدادی از ویژگی‌ها که اطلاعات بسیار تأثیرگذاری برای بالا بردن صحت جوابمان را داشته‌اند، حذف شده و اثر منفی روی درستی نتایج الگوریتم بگذارد. در برخی مسائل ما ویژگی‌های موجود را نمی‌توانیم حذف کنیم در این حالت می‌توانیم بجای کم کردن تعداد ویژگی‌ها به تعداد داده‌های آموزشی اضافه کنیم یا در انتخاب ویژگی‌هایی که باید حذف شود با دقت زیادی عمل کنیم و همچنین از ابتدا ویژگی‌های مناسب را انتخاب کنیم و همچنین از الگوریتم‌هایی که به صورت خودکار عمل انتخاب ویژگی را انجام می‌دهند استفاده کنیم که برای این منظور با استفاده از مثال زیر تأثیر ویژگی‌ها در کم یا بیش برازش را معرفی نموده‌ایم.

مثال) تأثیر تعداد ویژگی‌ها در دسته‌بندی داده‌ها در دسته‌بندی‌های مانند رگرسیون لاجستیک.

در شکل (۴-۴) در نمودارهای (الف)، (ب) و (پ) می‌توانید شرایط مختلفی را ملاحظه نمایید که در آنها چگونگی تأثیر تعداد ویژگی و درجه چندجمله‌ای خط فرضیه در نحوه دسته‌بندی مجموعه داده‌ها نشان داده شده است.



شکل (۴-۴): حالت‌های مختلف خط فرضیه با توجه به تعداد ویژگی‌های مختلف و درجه چندجمله‌ای.

در نمودار (الف) از شکل (۴-۴) به دلیل بالا بودن تعداد ویژگی‌ها و پایین بودن درجه چندجمله‌ای خط فرضیه ا. خط فرضیه در حالت کم برازش قرار گرفته و ویژگی‌ها را پوشش نداده و در نتیجه دسته‌بندی نامناسبی اتفاق افتاده است. در نمودار (ب) از شکل (۴-۴) با توجه به تعداد ویژگی‌ها و داده‌های موجود، خط فرضیه از درجه چندجمله‌ای مناسبی استفاده کرده و بر این اساس خط تصمیم رسم شده نشان‌دهنده مناسب بودن وضعیت دسته‌بندی داده‌ها است. در نمودار (پ) از شکل (۴-۴) نیز به دلیل بالا بودن درجه چندجمله‌ای بیش از حد نیاز مسئله، الگوریتم بیش برازش شده است.

## ۴-۸- اعتبارسنجی تقاطعی<sup>۱</sup>

اعتبارسنجی، به این معنی است که نتایجی که پس از اجرای یک الگوریتم به دست می‌آید با نتایجی که هدف ساخت سیستم بوده مطابقت می‌کند. در الگوریتم‌های یادگیری اعتبارسنجی تقاطعی، یک روش در مرحله آموزش الگوریتم است که در هنگام آموزش الگوریتم شرایط موجود الگوریتم را از نظر رسیدن به وضعیت مطلوب بررسی می‌کند.

اعتبارسنجی تقاطعی روش‌های مختلفی دارد و در الگوریتم‌های یادگیری ماشین اعم از رگرسیون یا کلاس‌بندی؛ جهت بررسی اینکه عملیات آموزش به درستی انجام می‌گیرد یا خیر؛ می‌تواند مورد استفاده قرار گیرد. در روش عمومی جهت پیاده‌سازی این روش، داده‌ها، به ۳ مجموعه داده شامل آموزش، مجموعه آزمون و مجموعه اعتبارسنجی تقسیم می‌شوند. معمولاً ۶۰٪ جهت آموزش و ۲۰٪ درصد جهت اعتبارسنجی، ۲۰٪ جهت آزمون داده‌ها به صورت انفاقی تقسیم‌بندی می‌شوند. در حین کامل شدن یک دوره آموزش با استفاده از مجموعه آموزشی؛ مدل یادگیری شده با استفاده از مجموعه اعتبارسنجی که دارای برچسب می‌باشند؛ اعتبارسنجی می‌شوند. تا مشخص شود که مرحله آموزش به خوبی انجام گرفته است یا خیر. در صورتی که آموزش به درستی انجام نشده باشد مرحله آموزش مجددًا صورت گرفته و دوباره اعتبارسنجی صورت می‌گیرد. این مراحل تا زمانی که خطای اعتبارسنجی به یک مقدار موردنظر و یا در حالت ایده‌آل به کمترین مقدار خود (به مقدار کمینه) برسد؛ ادامه می‌یابد. از نظر مدل یادگیری در این مراحل می‌توان این گونه اظهار کرد که چنانچه آموزش مدل ساخته شده به اندازه‌ای باشد که خطای اعتبارسنجی بالا باشد؛ یا در آموزش مدل بیش برآذش انجام گرفته (در این حالت خطای آموزش خیلی کم می‌باشد) و مدل آموزش یافته دارای واریانس بالا می‌باشد و یا از طرفی زمانی می‌تواند خطای اعتبارسنجی بالا باشد که آموزش مدل کافی نبوده و شرایط کم برآذش رخ داده است و مدل ایجاد شده دارای اصطلاحاً بایاس بالا می‌باشد. به کمک اعتبارسنجی و با بررسی نتایج حاصل از مجموعه داده اعتبارسنجی می‌توان زمانی را که با آموزش کافی به مدل رسیدیم که در آن خطای ناشی از اعتبارسنجی به حداقل خود رسیده و مدل حاصله به خوبی ایجاد شده است را بیابیم. گرچه در این حالت غالباً خطای حاصله از مجموعه آموزشی کمینه نیست.

توجه به این نکته ضروری است که روش اعتبارسنجی تقاطعی که برخی موقع برآورد چرخش و یا تخمین نیز نامیده می‌شود یک روش ارزیابی مدل است و نه یک روش ایجاد مدل، که هدفش بررسی چگونگی تعمیم نتایج یک تجربه و تحلیل آماری به یک مجموعه داده مستقل است.

## ۴-۹- اعتبارسنجی تقاطعی K گروهی<sup>۲</sup>

مانند اعتبارسنجی تقاطعی در این روش مجموعه داده به K مجموعه داده مجزا مساوی تقسیم می‌گردد. این روش نیز انواع مختلفی دارد. در روش عمومی یکی از این k مجموعه داده به عنوان اعتبارسنجی و k-1 دیگر به عنوان مجموعه آموزش استفاده می‌شوند. سپس یکی دیگر از این k مجموعه داده به عنوان اعتبارسنجی و k-1 دیگر به عنوان مجموعه آموزش به کار می‌رond. بنابراین k زوج شامل k-1 مجموعه داده آموزشی و یک مجموعه اعتبارسنجی

1. Cross validation

2. K-fold cross validation

مختلف ایجاد می‌شود. در کاربردهای عملی معمولاً  $K$  برابر با  $10$  می‌باشد. و پس از  $k$  بار تکرار؛ در عملیات نهایی خروجی در کاربردهای مانند رگرسیون؛ نتیجه حاصل از این  $K$  نتیجه ایجاد شده متوسط‌گیری می‌شود.

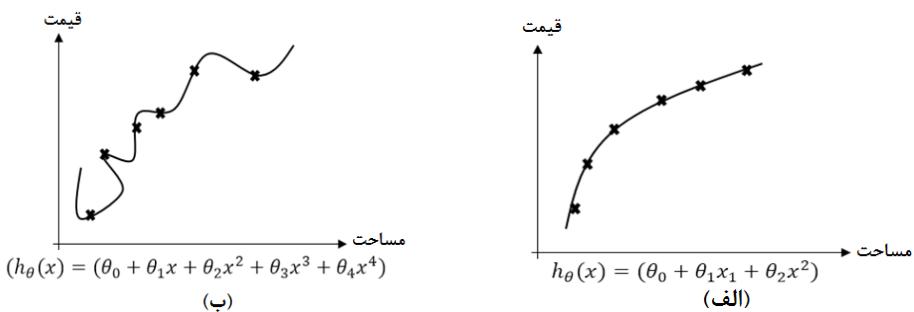
#### ۴-۱۰- تنظیم<sup>۱</sup>

به طور کلی قوانین مربوط به تنظیم‌کننده تلاش دارد تا الگوریتم مورد نظر را با تنظیم کردن از یک وضعیت نامناسب به یک حالت بهینه برساند مانند تنظیم پارامترهای تأثیرگذار در اجرای الگوریتم که در مطلب قبلی اشاره شد.

فرض کنید برای یک مسئله رگرسیون می‌خواهیم عمل بهینه‌سازی را انجام دهیم همانگونه که گفته شد می‌توان با تغییر درجه چندجمله‌ای یک الگوریتم را به حالت بهینه رساند که برای این کار باید تابع هزینه را مقداری تغییر داد. برای از بین بردن بیش برآش همانطور که می‌دانیم می‌توان با کم کردن تعداد ویژگی‌ها به انعطاف‌پذیری و تعمیم‌پذیری مدلمان کمک کنیم.

#### ۴-۱۱- تنظیم‌کننده برای رگرسیون خطی

قوانین کلی مربوط به تنظیم‌کننده برای بیشتر مسائلی که به بهینه‌سازی نیاز دارند صادق است ولی در ادامه مباحثت ما این قوانین را ابتدا با رگرسیون خطی و سپس با رگرسیون لاجستیک به عنوان مثال‌هایی برای فهم بهتر ارائه کرده‌ایم. فرض کنید چنانچه در شکل (۴-۵) مشاهده می‌نمایید دو مدل متفاوت از درجه چندجمله‌ای را داشته باشیم.



شکل (۴-۵): خط فرضیه‌هایی برای دو حالت مختلف الگوریتم از نظر پیچیدگی.

در شکل (۴-۵) (الف) خط فرضیه  $h_\theta(x)$  بسیار مناسب بوده ولی برای شکل (۴-۵) (ب) خط فرضیه  $h_\theta(x)$  بسیار پیچیده‌تر از حد مورد نیاز است. در اینجا ما با استفاده از روشی می‌خواهیم مقدار  $\theta_3$  و  $\theta_4$  را تا حد ممکن به کمینه یا حتی به صفر برسانیم که تأثیر دو جمله آخر در شکل (۴-۵) (ب) به حداقل برسد و بر این اساس خط فرضیه ما به وضعیتی مانند شکل (۴-۵) (الف) تبدیل شود که در این صورت هم آموزش بهتر صورت گرفته و هم تعمیم‌پذیری بهتری خواهیم داشت.

برای این کار به تابع هزینه که بصورت  $\min_{\theta} \frac{1}{2m} \sum_{i=1}^m h_{\theta}(x^{(i)}) - y^{(i)}_j$  است و در مباحث قبلی با آن آشنا شده‌ایم، دو جمله زیر را اضافه می‌کنیم.

$$x^3\theta_3^2 + x^4\theta_4^2$$

که در مجموع رابطه زیر را خواهیم داشت.

$$\left[ \min_{\theta} \frac{1}{2m} \sum_{i=1}^m h_{\theta}(x^{(i)}) - y^{(i)}_j \right]^2 + x^3\theta_3^2 + x^4\theta_4^2$$

حال اگر  $\theta_3$  و  $\theta_4$  در هر تکرار که برای  $J(\theta)$  اتفاق می‌افتد به میزان قابل ملاحظه‌ای کاهش یابد تا جایی که مقدارش به صفر برسد و در عمل وجود  $\theta_3$  و  $\theta_4$  در  $h_{\theta}(x)$  بی‌تأثیر شود چندجمله‌ای موجود که در شکل (۴-۵) (ب) وجود دارد را به مانند چندجمله‌ای موجود در شکل (۴-۵)(الف) که به صورت زیر است تبدیل خواهیم کرد که مدل مناسب‌تری است.

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

به صورت کلی می‌توان گفت که هر چه مقادیر  $\theta_n$  کوچک باشد  $h_{\theta}(x)$  ساده‌تر و بهتر خواهد بود و مدلمان تمایل کمتری به بیش برازش خواهد داشت.

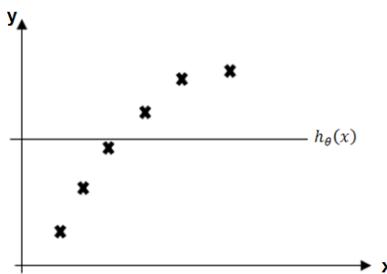
#### ۴-۱۲- رابطه تنظیم‌کننده برای تابع هزینه

در مثال قبل ما تنها دو پارامتر  $\theta_3$  و  $\theta_4$  را مدنظر قرار دادیم ولی اگر بخواهیم تمامی پارامترها را در تابع هزینه بگذاریم و کاهش دهیم از رابطه (۱) استفاده می‌کنیم.

$$J(\theta) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}_j)^2 + \lambda \sum_{j=1}^n \theta_j^2 \right] \quad \text{رابطه (۱)}$$

بخش آخر رابطه ۱ که به صورت  $\lambda \sum_{j=1}^n \theta_j^2$  می‌باشد رابطه اصلی تنظیم‌کننده و علامت  $\lambda$  یا لاندا به پارامتر تنظیم‌کننده معروف است که یک مقدار ثابت می‌گیرد و در هر تکرار به یکی از پارامترها ضرب می‌شود و بدین ترتیب تمامی پارامترها را به اندازه مورد نیازی که باعث به کمینه رسیدن تابع هزینه شود کاهش می‌دهد.

توجه داشته باشید مقداری که به  $\lambda$  داده می‌شود اگر از حد مورد نیاز مسئله بزرگ‌تر باشد نه تنها از بیش برازش جلوگیری می‌کند بلکه به قدری میزان پارامترها را کاهش می‌دهد که ممکن است از یک چندجمله‌ای تنها  $\theta_0$  باقی بماند و این بار مشکل کم برازش به وجود آید مانند خط فرضیه شکل (۶-۴) که در آن بدترین حالت قرارگیری خط فرضیه را مشاهده می‌نمایید.



شکل (۶-۴): قرارگیری خط فرضیه در وضعیت نادرست و کم برازش.

#### ۴-۱۳- انتخاب مدل<sup>۱</sup> در گرادیان نزولی

به الگوریتم زیر توجه کنید که در آن نحوه استفاده شدن رابطه تنظیم‌کننده نشان داده شده که برای گرادیان نزولی می‌باشد.

تکرار حلقه برنامه {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} - \frac{\lambda}{m} \theta_j$$

(j=1,2,3 ,...,n)

}

همان‌طور که در الگوریتم بالا ملاحظه می‌نمایید در ردیف اول مربوط به پارامترها برای  $\theta_0$  به صورت جداگانه محاسبات انجام می‌شود و نیازی به کم کردن مقدار پارامتر  $\theta_0$  نمی‌باشد و  $\theta_j$  دوباره به صورت ( $j=1, 2, 3, \dots, n$ ) کاهش خواهد داد.

همچنانی می‌توان رابطه  $\theta_j$  را به صورت زیر هم نوشت.

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^m h_\theta(x^{(i)}) - y^{(i)} x_j^{(i)}$$

که در آن  $\alpha \frac{\lambda}{m}$  کوچک‌تر از یک خواهد رسید.

$$(1 - \alpha \frac{\lambda}{m}) < 1$$

توجه شود که اگر فرض کنید  $\alpha = \frac{0.99}{m} - 1$  باشد با توجه به این فرض وقتی  $\theta_j$  در یک عددی کوچک‌تر از یک ضرب شود در هر تکرار مقدارش کاهش خواهد یافت.

چنانچه مشاهده نمودید ما با استفاده از قوانین تنظیم‌کننده در الگوریتم گرادیان نزولی نشان دادیم که چگونه و از چه روابطی می‌توان برای جلوگیری از پدیده بیش برآذش استفاده کرد.

#### ۱۴- تنظیم‌کننده برای رگرسیون لاجستیک

فرض کنید رابطه  $h_{\theta}(x)$  زیر را برای رگرسیون لاجستیک داریم که به عنوان مثالی از یک خط فرضیه با درجه چندجمله‌ای خیلی بالا است و موجب ایجاد حالت بیش برآذش شده و ما می‌خواهیم آن را بهینه کنیم.

$$h_{\theta}(x) = (\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

چنانچه در مباحث گذشته گفته شد، رابطه تابع هزینه برای رگرسیون لاجستیک به صورت زیر می‌باشد.

$$J(\theta) = [\frac{1}{m} \sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)}))]$$

در اینجا نیز باید برای انجام فرایند تنظیمسازی از رابطه زیر استفاده کنیم که به رابطه بالا اضافه می‌شود.

$$\frac{\lambda}{2m} \sum_{i=1}^n \theta_j^2$$

توجه شود که حتی اگر درجه چندجمله‌ای بسیار بالایی داشته باشیم با اضافه کردن تنظیم‌کننده می‌توان مقادیر  $\theta$  را در سطح پایین نگه داشت و از بیش برآذش جلوگیری کرد.

#### ۱۵- الگوریتم گرادیان نزولی با استفاده از تنظیم‌کننده

به طریقه پیاده‌سازی الگوریتم کلی به استفاده از تنظیم‌کننده توجه نمایید.

الگوریتم گرادیان نزولی:

تکرار الگوریتم }

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[ \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

[j=1, 2,3, .....n]

{

چنانچه در بخش توضیحات رگرسیون لاجستیک گفته شد مقادیر  $h_\theta(x^{(i)})$  در درونتابع سیگوئید قرار داده میشود و مقدار آن با مقداری که  $h_\theta(x^{(i)})$  در رگرسیون خطی تولید میکند، متفاوت میباشد. به هر حال میتوان با استفاده از رابطه مربوط به تنظیمکننده، مقادیر پارامترها را به وضعیت مناسب تغییر داده و از بیش برآذش یا کم برآذش جلوگیری کرد که این یکی از موضوعات اساسی در بهینهسازی الگوریتمهای یادگیرنده میباشد.



فصل

## شبکه عصبی

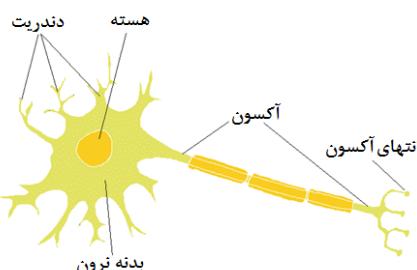
## ۱-۵- شبکه عصبی<sup>۱</sup>

با شنیدن واژه شبکه عصبی در سیستم‌های محاسباتی چیزی که به ذهن می‌رسد ماشینی است که تلاش می‌کند تا مانند مغز انسان عمل کند. گمان می‌رود که مغز انسان از تعداد بسیار زیادی نرون<sup>۲</sup> تشکیل شده باشد که هر نرون با تعداد بسیار زیادی نرون دیگر در ارتباط است. با اینکه سرعت سوئیچینگ نرون‌ها در مقایسه با کامپیوترها بسیار ناچیز می‌نماید، با این وجود آدمی قادر است در ۰.۱ ثانیه تصویر یک انسان را بشناسد. این قدرت فوق العاده گمان می‌رود از پردازش موازی توزیع شده<sup>۳</sup> با تعداد زیادی از نرون‌ها حاصل شده باشد.

### ۱-۱- ساختار نرون مغزی

نرون‌ها سلول‌هایی هستند که وظیفه انتقال اطلاعات عصبی را بر عهده دارند. نرون‌ها از طریق زائدۀ‌هایی بنام دندربیت<sup>۴</sup> اطلاعات را دریافت کرده در سوما<sup>۵</sup> که شامل هسته و جسم سلولی می‌باشد پردازش را انجام داده و پس از پردازش، از طریق زائدۀ‌های دیگری بنام آکسون<sup>۶</sup> اطلاعات را به سلول بعدی منتقل می‌کنند.

شکل (۱-۵) بخش‌های مختلف یک نرون مغز انسان را نشان می‌دهد.



شکل (۱-۵): نرون عصبی طبیعی و بخش‌های تشکیل‌دهنده آن.

## ۲-۵- شبکه عصبی مصنوعی<sup>۷</sup>

شبکه‌های عصبی مصنوعی، سیستم‌ها و روش‌های محاسباتی برای یادگیری ماشینی، نمایش دانش و در انتها اعمال دانش به دست آمده در جهت پیش‌بینی پاسخ‌های خروجی از سامانه‌های پیچیده هستند. ایده اصلی این گونه شبکه‌ها برگرفته از نحوه عملکرد سیستم عصبی زیستی است که هدف آنها پردازش داده‌ها برای یادگیری در راستای ایجاد دانش براساس داده‌های موجود است. این سیستم‌ها از تعداد زیادی عناصر پردازشی بهم پیوسته با نام نرون تشکیل شده است که برای حل یک مسئله تعداد زیادی از این سلول‌ها به صورت هماهنگ و با نظم خاصی با هم همکاری می‌کنند.

- |                                    |                              |
|------------------------------------|------------------------------|
| 1. Neural network                  | 5. Soma                      |
| 2. Neuron                          | 6. Axon                      |
| 3. Distributed parallel processing | 7. Artificial neural network |
| 4. Dendrite                        |                              |

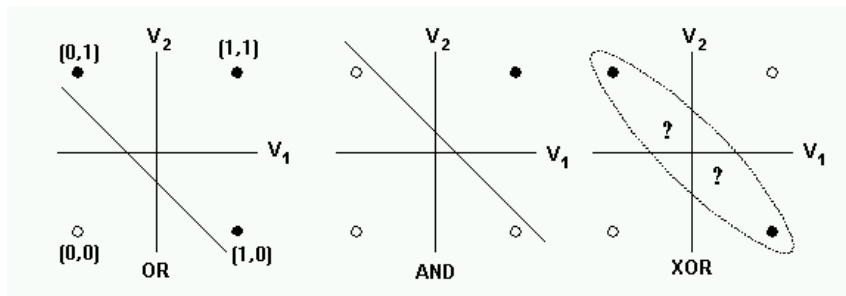
## ۱-۲-۵- تاریخچه شبکه عصبی مصنوعی

اولین نظریه شبکه عصبی مصنوعی توسط دو ریاضی دان با نامهای مک کلاک<sup>۱</sup> و پیت<sup>۲</sup> در سال ۱۹۴۳ مطرح شد. شبکه عصبی مطرح شده توسط آنها چند کanal ورودی و یک کanal خروجی داشت که از ورودی ها مقادیر صفر یا یک دریافت می شد و پس از پردازش مقدار صفر یا یک در خروجی این شبکه تولید می شد، صفر به معنای غیرفعال یا نادرست<sup>۳</sup> و یک به معنای فعال یا درست<sup>۴</sup> می باشد. این شبکه به دلیل نقایصی با شکست رو به رو شد اما نقطه شروعی برای محققانی مانند هب<sup>۵</sup> شد، هب روانشناسی بود که با ارائه نظریه ای کارایی این نوع شبکه را ارتقاء داد و سرانجام در سال ۱۹۵۸ شخصی به نام روزنبلت<sup>۶</sup> یک مدل کامل شده ای را ارائه داد که به قانون پرسپترون معروف است و پس از آن نیز تحقیقات فراوانی در این زمینه انجام شده که باعث پیدایش مدل های پیشرفته و متنوعی از شبکه های عصبی شده. این مدل ها برای حل مسائل در دامنه های مختلف مورد استفاده قرار می گیرند، امروزه با مطرح شدن شبکه های عصبی عمیق تحول چشم گیری در این حیطه اتفاق افتاده است.

## ۵- پرسپترون ساده<sup>۷</sup>

پرسپترون یک مدل اولیه از شبکه عصبی می باشد که به صورت یادگیری با نظارت عمل می کند و یک نوع تفکیک کننده خطی و دودویی است و در آن خروجی فقط به سمت جلو حرکت می کند که جهت تفکیک مقادیر به دو دسته صفرها یا یک ها استفاده می شود و فقط قادر است مثال هایی را یاد بگیرد که بصورت خطی جدا یابنده باشند مانند توابع بولی<sup>۸</sup> NAND, NOR, AND, OR را یاد می گیرد ولی برای XOR بدلیل اینکه به صورت خطی جدا یابنده نمی باشد کارایی ندارد.

به چند نمودار شکل (۲-۵) توجه نمایید.



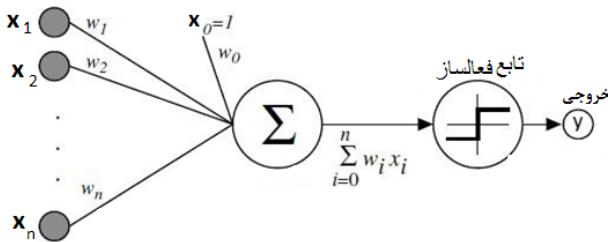
شکل (۲-۵): نمودارهای مربوط به توابع بولی خطی و غیر خطی.

در کل یک پرسپترون متشکل است از یک بخش ورودی که معمولاً آن را لایه به حساب نمی آورند، یک لایه پردازش که در اینجا تنها یک نرون می باشد (تابع تجمعی) و در انتهای یک لایه خروجی که شامل یک نرون می باشد و مقادیر

- 1. McCulloch
- 2. Pitts
- 3. False
- 4. True

- 5. Heb
- 6. Rosenbelt
- 7. Simple perceptron
- 8. Boolean

نهایی پردازش شده را ارائه می‌کند. در شکل زیر بخش‌های یک پرسپترون ساده را می‌توانید مشاهده نمایید.



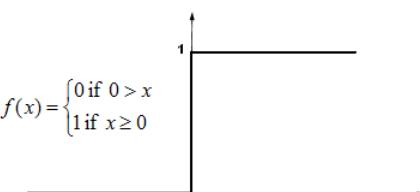
شکل (۵-۳): پرسپترون ساده با یک نرون که از تابع فعال‌ساز پله‌ای استفاده کرده است.

Xها یک بردار از مقادیر ورودی می‌باشند.

Wها یک بردار از وزن‌هایی با مقادیر حقیقی است که real در مقایسه با موهومی درست است ولی مقادیر واقعی<sup>۱</sup> مجموع وزن‌ها را محاسبه می‌کند. (توجه شود که در بخش‌های قبلی به جای پارامتر w از θ استفاده شده است).

(x) f(y) یا نتیجه پردازش شبکه می‌باشد که برای دسته‌بندی به نمونه مثبت یا منفی، در مسائل دسته‌بندی دودویی استفاده می‌شود.

نکته قابل توجه این است که هر نرون به خودی خود دارای یک وزنی می‌باشد که با  $w_0$  و یا b می‌توانیم نشان دهیم ولی وزن نرون‌های ورودی را با  $w(1,2,3,\dots,n)$  که وزن هر کدام از ورودی‌ها می‌باشد نشان می‌دهیم. در یک پرسپترون ورودی‌ها با وزن‌های خود وارد شبکه شده و توسط یک نرون محاسبه شده و توسط یک تابع تبدیل که به آن تابع حد آستانه<sup>۲</sup> یا تابع فعال‌ساز<sup>۳</sup> گفته می‌شود به یکی از دو کلاس صفرها یا یک‌ها اختصاص می‌یابد. تابع حد آستانه که در اینجا یک تابع پله‌ای<sup>۴</sup> است به صورت شکل (۵-۵) می‌باشد.



شکل (۵-۵): تابع پله و عبارات شرطی آن.

### ۵-۳-۱- محاسبه پرسپترون ساده

رابطه زیر برای محاسبه یک پرسپترون تک لایه با یک نرون مورد استفاده قرار می‌گیرد که برای محاسبه مسائلی مانند and و or و مسائلی از این قبیل که به صورت خطی جدایی‌پذیرند می‌تواند مورد استفاده قرار گیرد، ولی بیشتر مسائل در دنیای واقعی بسیار پیچیده‌تر از آن است که بتوان با یک پرسپترون ساده حل کرد و این مدل

1. real
2. Threshold
3. Activation function
4. step function

ساده برای درک راحت‌تر از نحوه عملکرد شبکه عصبی در اینجا ارائه گردیده است.

$$y = \text{step}[w_0 + \sum_{i=1}^n w_i x_i] \quad (1)$$

رابطه (1) را به صورت زیر می‌توان در نظر گرفت.

$$y = \text{step}[w_0 + \sum_{i=1}^n w_i x_i] = w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

### ۳-۵ - شبیه‌سازی گیت OR با پرسپترون ساده

ساخت هر شبکه عصبی شامل ایجاد مدل و ارزیابی مدل و در مرحله بعد شامل آزمون مدل می‌باشد. در واقع پس از ساخت مدل با استفاده از داده‌های آموزشی شبکه را آموزش می‌دهیم و در صورت مطلوب بودن شبکه در مرحله ارزیابی، با استفاده از داده‌های آزمون شبکه را آزمایش می‌کنیم. اگر در این مرحله شبکه با موفقیت به کار خود پایان داد این شبکه برای حل مسائل مربوطه در محیط واقعی آماده می‌باشد. در اینجا نیز برای ایجاد گیت OR همین مراحل را انجام می‌دهیم. به جدول (۱-۵) توجه نمایید.

جدول (۱-۵): مقادیر دودویی مربوط به OR منطقی.

$x_1$	$x_2$	t	Y
0	0	0	?
0	1	1	?
1	0	1	?
1	1	1	?

t همان حواب مورد انتظار ماست یعنی حواب  $x_1$  OR  $x_2$  در خروجی به صورت t می‌باشد و قبلاً توسط دانشمندان به اثبات رسیده است. y همان جوابی می‌باشد که شبکه عصبی ساخته شده توسط ما محاسبه کرده و به خروجی خواهد فرستاد. اگر بتوانیم شبکه‌ای بسازیم و آموزش دهیم که بتواند در خروجی yهایی تولید کند که نظیر به نظیر با هماهنگ باشد و یا حداقل تفاوت را داشته باشد، شبکه عصبی ایجاد شده با موفقیت پیاده‌سازی گردیده است. با توجه به مقادیر جدول (۲-۵) و توضیحاتی که در ادامه برای آن ارائه نموده‌ایم توجه کنید که در سه نسل یا دور<sup>۱</sup> آموزشی گیت OR را شبیه‌سازی کرده است.

برای ایجاد شبکه‌ای که بتواند گیت OR را شبیه‌سازی کند ابتدا باید الگوریتم ایجاد شده را به گونه‌ای مدل کنیم که بتواند به مانند گیت OR خروجی مورد نظر را تولید کنید و برای این کار شبکه را ساخته و آن را به قدری آموزش می‌دهیم که موفق به انجام این کار شود.

جدول (۲-۵): مقادیر مربوط به دورهای آموزشی پرسپترون ساده برای OR.

ردیف epoch	دور آموزشی	دروی اول	دروی دوم	دروی انتظار	خروجی مورد انتظار	خروجی محاسبه شده	خطای آموزشی	وزن اول	وزن دوم
		$x_1$	$x_2$	$t$		$y$	$e$	$w_1$	$w_2$
۱	۱	۰	۰	۰	۰	۰	۰	-۰.۲	۰.۴
۲	۱	۰	۱	۱	۱	۱	۰	-۰.۲	۰.۴
۳	۱	۱	۰	۱	۰	۰	۱	۰	۰.۴
۴	۱	۱	۱	۱	۱	۱	۰	۰	۰.۴
۵	۲	۰	۰	۰	۰	۰	۰	۰	۰.۴
۶	۲	۰	۱	۱	۱	۱	۰	۰	۰.۴
۷	۲	۱	۰	۱	۰	۰	۱	۰.۲	۰.۴
۸	۲	۱	۱	۱	۱	۱	۰	۰.۲	۰.۴
۹	۳	۰	۰	۰	۰	۰	۰	۰.۲	۰.۴
۱۰	۳	۰	۱	۱	۱	۱	۰	۰.۲	۰.۴
۱۱	۳	۱	۰	۱	۱	۱	۰	۰.۲	۰.۴
۱۲	۳	۱	۱	۱	۱	۱	۰	۰.۲	۰.۴

مانند تمام روش‌های مبتنی بر یادگیری در اینجا نیز ابتدا الگوریتم را آموزش داده و سپس آن را آزمایش می‌کنیم. اما ما در ادامه این مطلب تنها مرحله ایجاد مدل و آموزش را به شما معرفی می‌کنیم تا فهم آن آسانتر و مطالب ساده بماند. محاسبات پایین برای به دست آوردن یهای چهار ردیف اول جدول (۲-۵) می‌باشد.

ابتدا مقدار  $0 = x_1 = x_2$  بر اساس سطر اول جدول (۲-۵) به پرسپترون که تابع تجمعی آن را در رابطه (۱) مشاهده نموده‌اید، می‌دهیم و خروجی  $y$  آن را محاسبه می‌کنیم.

$$Y = \text{step}(-0.2*0+0.4*0) = \text{step}(0) = 0$$

$Y = 0$  یعنی خروجی محاسبه شده همانند خروجی مورد انتظار ( $t=0$ ) مساوی صفر می‌باشد. در نتیجه خطای شبکه در این مرحله صفر است ( $e=0$ ) که با رابطه زیر محاسبه می‌شود.

$$\text{Error} = e = t - y \quad \text{رابطه (۲)}$$

به دلیل اینکه خطای ردیف اول ( $e=0=0$ ) صفر می‌باشد بدون تغییر وزن‌ها به ردیف دوم می‌رویم.

$$y = \text{step}(-0.2*0+0.4*1) = \text{step}(0.4) = 1$$

$$e = 1 - 1 = 0$$

برای محاسبه  $y$  در هر ردیف جدید از وزن‌های ردیف قبلی استفاده می‌شود و در صورت عدم وجود خطای تغییر وزن انجام نمی‌شود.

$$y = \text{step}(-0.2*1+0.4*0) = \text{step}(-0.2) = 0$$

$$e = 1 - 0 = 1$$

در این مرحله به دلیل وجود خطای تغییر یابد و برای تغییر وزن‌ها اگر خروجی محاسبه شده کمتر از خروجی مورد انتظار است و  $e = t - y > 0$  باید وزن‌ها بیشتر شوند و اگر خروجی محاسبه شده بیشتر از خروجی مورد

انتظار است  $e=t-y < 0$  باید وزن‌ها کمتر شوند که با رابطه (۳) تغییر وزن‌ها انجام می‌شود.

$$W_i(\text{new}) = W_i(\text{old}) + \alpha * x_i * e \quad (3)$$

در معادله بالا نرخ یادگیری می‌باشد همان‌گونه که در فصول گذشته توضیح داده شد نرخ یادگیری مقداردهی اولیه می‌شود و باید مقدار مناسب به آن اختصاص داده شود  $\alpha$  معمولاً بین صفر و یک ( $0 < \alpha < 1$ ) انتخاب می‌شود که هرچه مقدارش بیشتر باشد سرعت یادگیری بیشتر می‌باشد و بالعکس. ما در این مثال نرخ یادگیری را به صورت اولیه مساوی ۰.۲ قرار می‌دهیم و در ادامه برای ردیف سوم که دارای خطأ بود به صورت زیر وزن‌ها را تغییر می‌دهیم.

$$W_1 W_1 = W_1 W_1 + \alpha * X_1 X_1 * e = -0.2 + 0.2 * 1 * 1 = 0$$

$$W_2 W_2 = W_2 W_2 + \alpha * X_2 X_2 * e = 0.4 + 0.2 * 0 * 1 = 0.4$$

همان‌گونه که مشاهده می‌شود  $W_2$  بدون تغییر است و  $W_1$  که دلیل وجود خطأ بوده تغییر کرده است که از این به بعد برای محاسبه ردیف‌های بعدی از وزن‌های جدید استفاده می‌شود.

$$y = \text{step}(0 * 1 + 0.4 * 1) = \text{step}(0.4) = 1$$

$$e = 1 - 1 = 0$$

در این ردیف خطای وجود نداشت و تغییر وزن‌ها نیاز نیست ولی با توجه به جدول (۲-۶) به دلیل وجود خطأ در دور آموزشی<sup>۱</sup> اول که در ردیف سوم آن اتفاق افتاد همین جریان به صورت دورهای آموزشی بعدی تا کاهش خطای شبکه به حد مطلوب، ادامه می‌یابد و همان‌طور که مشاهده می‌نمایید در ردیف سوم از دور آموزشی دوم خطأ مجددًا اتفاق افتاده و وزن‌ها تغییر کرده است ولی در دور آموزشی سوم هیچ خطای نیست و آموزش شبکه پایان یافته است.

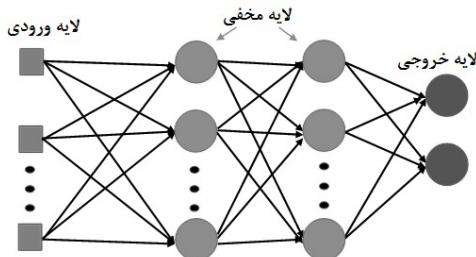
## ۴-۵- پرسپترون چند لایه<sup>۲</sup>

پرسپترون چند لایه یک مدل شبکه عصبی مصنوعی پیشخور<sup>۳</sup> است به این معنی که در آن مجموعه‌ای از داده‌ها از یک لایه به عنوان لایه ورودی وارد شبکه شده و پس از پردازش در هر لایه، به صورت حرکت مستقیم به سمت جلو به لایه آخر که همان لایه خروجی می‌باشد هدایت می‌شوند. شامل چندین لایه از گره‌های یک گراف جهتدار می‌باشد که هر گره به عنوان یک نرون در نظر گرفته می‌شود و نرون‌های هر لایه به تمامی نرون‌های لایه بعدی متصل می‌شود ولی نرون‌های موجود در یک لایه به هم متصل نیستند. تعداد لایه‌های مخفی و نرون‌های موجود در هر لایه از شبکه عصبی براساس شرایط مسئله و توسط طراح شبکه تعیین می‌شود که هرچه تعداد لایه‌ها و نرون‌های شبکه افزایش یابد می‌تواند مسائل پیچیده‌تری را حل کند اما حجم محاسباتی بالاتری خواهد داشت و بار پردازشی بالاتر را به سیستم تحمیل خواهد کرد از این رو پیاده‌سازی مدلی که هم از پیچیدگی کمتری برخوردار باشد و هم به تمامی نیازهای مسئله پاسخ دهد نیاز به تعمل و دقت بسیار دارد.

1. Epoch

2. Multi layer perceptron

3. Feed-forward



شکل (۵-۵): پرسپترون چندلایه با دو لایه مخفی و دو نرون خروجی.

## ۴-۵-۱- الگوریتم آموزش پرسپترون چند لایه

چنانکه در بخش قبل توضیحات مختصراً ارائه شد در روش‌های مختلف یادگیری ماشین و شبکه عصبی رویکرد اصلی ایجاد سیستمی می‌باشد که آن را آموزش دهیم و سپس براساس آموزشی که داده شده این سیستم بتواند در کاربرد واقعی برای ورودی‌های جدید، جواب مناسب ارائه کند. مثلاً شبکه عصبی را مدل کنیم که بتواند پس از آموزش مناسب برای مسئله AND در مرحله آزمون جواب درست را ارائه کند.

## ۴-۵-۲- پس انتشار خطأ و گرادیان نزولی<sup>۱</sup>

در این روش داده‌ها از لایه ورودی وارد شبکه شده و در لایه‌های میانی پردازش شده و با هدف تولید خروجی مورد انتظار ما به نرون یا نرون‌های موجود در لایه خروجی ارسال می‌شود که در اصل همان عمل پیش خور اتفاق افتاده است در ادامه اگر نتیجه محاسبات شبکه برابر خروجی مورد انتظار نبود یعنی خطأ رخ داده و برای برطرف کردن خطأ، داده‌ها در شبکه در جهت عکس ارسال می‌شوند تا با تغییر وزن‌ها، داده‌ها دوباره محاسبه شده و به لایه خروجی ارسال شوند. این عمل پس انتشار تا زمانی که خطای شبکه به اندازه قابل قبولی کاهش پیدا کند ادامه می‌یابد.

## ۴-۵-۳- مراحل الگوریتم پرسپترون با روش پس انتشار خطأ

یکی از الگوریتم‌های اساسی در فرایند آموزش شبکه‌های عصبی، الگوریتم پس انتشار خطأ می‌باشد. فرایند عملکرد این الگوریتم را می‌توان به صورت کلی به چهار بخش تقسیم نمود.

۱. فرایند پیشخور
۲. پس انتشار خطأ به لایه خروجی
۳. پس انتشار خطأ به لایه‌های مخفی
۴. به روزرسانی وزن‌ها

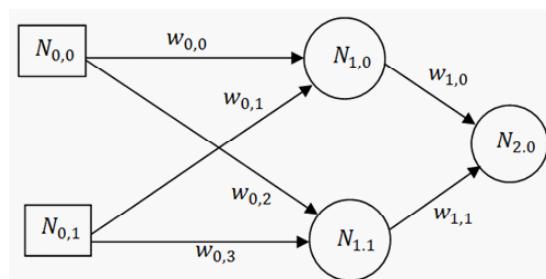
---

1. Error back propagation  
2. Gradient decent

یکی از دلایل خاتمه کار در الگوریتم پس انتشار خطای این است که در آن تابع خطای به کوچکترین میزان قابل قبول کاهش می‌یابد.

#### ۴-۴-۵ - فرایند اجرای الگوریتم پرسپترون چندلایه با روش پس انتشار خطای

اگر به شکل (۶-۵) توجه نمایید مدل شبکه عصبی برای حل معادله AND به صورت دو نرون ورودی ( $N_{0,0}$  و  $N_{0,1}$ )، دو نرون در لایه مخفی<sup>۱</sup> ( $N_{1,1}$  و  $N_{1,0}$ )، و یک نرون در لایه خروجی ( $N_{2,0}$ ) می‌باشد.



شکل (۶-۵): پرسپترون چندلایه با یک لایه مخفی برای محاسبه AND در پس انتشار خطای.

نرون‌های لایه ورودی براساس وزن‌های  $w_{0,0}$  تا  $w_{0,3}$  به نرون‌های لایه مخفی متصل شده‌اند و همچنین نرون‌های لایه مخفی براساس وزن‌های  $w_{1,0}$  و  $w_{1,1}$  به نرون لایه خروجی متصل شده‌اند. توجه به این نکته مهم است که مقادیر اولیه اختصاصی به وزن‌ها به صورت تصادفی انجام شده و این مقادیر در هر دور آغازشی در فرایند پس انتشار خطای به صورت مناسب در راستای رسیدن به مقدار بهینه خود تغییر می‌کنند و مقدار جدید به خود می‌گیرند.

جدول (۳-۵): مقادیر منطقی برای AND

row	$n_{0,0}$	$n_{0,1}$	$n_{2,0}$ (Output)
1	1	1	1
2	1	0	0
3	0	1	0
4	0	0	0

چنانکه در جدول (۳-۵) مشخص است مقادیر نرون‌های ورودی و خروجی مورد انتظار مشخص می‌باشند و چنانکه در بخش قبل اشاره شد خروجی محاسبه شده نهایی باید مانند خروجی مورد انتظار باشد. مقدار نرخ یادگیری  $\alpha$  برابر با ۰.۴۵ و مقدار مومنتوم  $\mu$  را برابر با ۰.۹ قرار می‌دهیم.

همچنین تابع  $f(x) = 1.0 / (1.0 + \exp(-x))$  را که تابع سیگموئید است به عنوان تابع حد آستانه‌گیری مورد استفاده قرار می‌دهیم.

جدول(۴-۵): مقادیر اولیه وزن‌ها.

$w_{0,0} = 0.4$	$w_{0,2} = -0.1$	$w_{1,0} = 0.06$
$w_{0,1} = 0.1$	$w_{0,3} = -0.1$	$w_{1,1} = -0.4$

در ادامه ما تنها از ردیف اول جدول(۳-۵) که به صورت ۱ و ۱ می‌باشد برای محاسبات استفاده می‌نماییم. البته این محاسبات برای هر سه ردیف دیگر جدول نیز باید به صورت مشابه انجام گردد.

#### ۵-۱-۴-۴-۱- مومنتوم<sup>(۱)</sup>

روشی برای بالا بردن سرعت رسیدن شبکه به کمینه سراسری و اجتناب از گیر کردن در کمینه محلی می‌باشد. از این روش در هنگام تغییر وزن‌ها استفاده می‌شود به این صورت که برای تغییر وزن‌ها در لحظه بعد ( $t+1$ ) با در نظر گرفتن لحظه فعلی ( $t$ ) و لحظه قبل ( $t-1$ ), بر نرخ تغییرات یا همان  $\alpha$  اثر می‌گذارد تا الگوریتم بتواند با سرعت مناسبی برای رسیدن به نقطه کمینه حرکت کند.

#### ۵-۲-۴-۴-۲- محاسبات پیشخور

محاسبات فرایند پیشخور به صورت دو مرحله‌ای می‌باشد که در آن ابتدا مقادیر نرون‌های لایه مخفی محاسبه شده و سپس در مرحله بعد مقدار نرون لایه آخر بر اساس آن نرون‌ها محاسبه می‌شود. به عبارتی دیگر مقادیر ورودی نرون‌های  $N_{0,0}$  و  $N_{0,1}$  از طریق نرون‌های موجود در لایه مخفی که همان نرون‌های  $N_{1,0}$  و  $N_{1,1}$  می‌باشند به سمت جلو در شبکه پیشخور می‌شوند که با مقادیر وزن‌های نرون‌های متصل به هم و همچنین مقادیر نرون‌های لایه مخفی ضرب می‌شوند که این فرایند را در زیر مشاهده می‌نمایید. البته به خاطر داشته باشیم که مقادیر ورودی یا همان مقادیر مربوط به  $N_{1,0}$  و  $N_{1,1}$  به ترتیب برابر با ۱ و ۱ می‌باشد.

$$N_{1,0} = f(x1) = f(w_{0,0} \times N_{0,0} + w_{0,1} \times N_{0,1}) = f(0.4 + 0.1) = f(0.5) = 0.622459$$

$$N_{1,1} = f(x2) = f(w_{0,2} \times N_{0,0} + w_{0,3} \times N_{0,1}) = f(-0.1 - 0.1) = f(-0.2) = 0.450166$$

با محاسبه شدن مقادیر نرون لایه مخفی، شبکه به سمت لایه مخفی محاسبات را ادامه می‌دهد و مقادیر به دست آمده را به نرون آخر  $N_{2,0}$  انتقال می‌دهد تا نرون آخر با استفاده از آن مقدارش محاسبه شود که این بخش دوم فرایند پیشخور است که به صورت زیر می‌باشد.

$$N_{2,0} = f(x3) = f(w_{1,0} \times N_{1,0} + w_{1,1} \times N_{1,1}) =$$

$$f(0.06 \times 0.622459 + (-0.4) \times 0.450166) = f(-0.1427188) = 0.464381$$

### ۴-۳-۵- پس انتشار خطا به لایه خروجی

پس از محاسبه خروجی در فرایند پیشخور، حالا زمان انجام عمل پس انتشار خطا است که برای این منظور باید مقدار خطای نرون لایه آخر  $N_{2,0}$  محاسبه شود. بر اساس جدول (۳-۵) مقدار به دست آمده نهایی خروجی باید به صورت ۱ شود. در اصل اگر خروجی محاسبه شده نهایی شبکه برابر یا بزرگتر از ۰,۵ باشد خروجی را ۱ و در غیر این صورت خروجی را ۰ در نظر می‌گیریم که مقدار تخمین زده شده برای نرون خروجی در محاسبات انجام شده تا این مرحله برابر با ۰.۴۶۴۳۸۱ می‌باشد و با توجه به این که هنوز به مقدار ۰,۵ نرسیده پس باید خطای شبکه محاسبه شود و محاسبات خطا نیز به صورت زیر انجام می‌شود.

$$\begin{aligned} N_{2,0}\text{Error} &= N_{2,0} \times (1 - N_{2,0}) \times (N_{2,0}\text{Desired} - N_{2,0}) = \\ &0.464381 \times (1 - 0.464381) \times (1 - 0.464381) = 0.133225 \end{aligned}$$

حالا با مشخص شدن مقدار خطای این مقدار در فرایند پس انتشار خطا در راستای بهینه ساختن خطا مورد استفاده قرار می‌گیرد که این بخش نیز دارای دو مرحله می‌باشد. ابتدا خطا از لایه آخر به لایه مخفی پس انتشار می‌شود که برای این عمل الگوریتم از نرخ یادگیری و همچنین از مومنتوم استفاده می‌نماید.

در این مرحله وزن‌های  $w_{1,0}$  و  $w_{1,1}$  به روزسانی می‌شوند که قبل از به روزسانی وزن‌ها باید ابتدا نرخ تغییر آن به دست آید که برای این عمل باید مقدار مربوط به نرخ یادگیری، مقدار خطا و مقدار مربوط به نرون  $N_{2,0}$  در هم ضرب شوند.

$$\begin{aligned} \Delta w_{1,0} &= \alpha \times N_{2,0}\text{Error} \times N_{1,0} = 0.45 \times 0.133225 \times 0.622459 = 0.037317 \\ \Delta w_{1,1} &= \alpha \times N_{2,0}\text{Error} \times N_{1,1} = 0.45 \times 0.33225 \times 0.450166 = 0.026988 \end{aligned}$$

حال در ادامه با توجه به مقادیر به دست آمده به عنوان نرخ تغییرات وزن‌ها، وزن جدید برای  $w_{1,0}$  و  $w_{1,1}$  قابل محاسبه است.

$$\begin{aligned} w_{1,0}\text{New} &= w_{1,0}\text{Old} + \Delta w_{1,0} + (\mu \times \Delta(t-1)) = \\ &0.06 + 0.037317 + 0.9 \times 0 = 0.097137 \end{aligned}$$

$$\begin{aligned} w_{1,1}\text{New} &= w_{1,1}\text{Old} + \Delta w_{1,1} + (\mu \times \Delta(t-1)) = \\ &-0.4 + 0.026988 = -0.373012 \end{aligned}$$

رابطه (۱- t)  $\Delta$  مشخص کننده تغییرات قبلی دلتای وزن مورد نظر می‌باشد و به دلیل اینکه در مثال ما تغییرات دلتای قبلی وجود ندارد پس مقدار آن را برابر ۰ در نظر می‌گیریم و اگر یک دور آموزشی به این محاسبات اضافه شود در این صورت می‌تواند یک مقداری را به دست آورد.

#### ۴-۴-۴-۵ پس انتشار به لایه مخفی

در این مرحله خطا باید از لایه مخفی به لایه ورودی پس انتشار شود که این عمل مقداری از پس انتشار خطا از لایه خروجی به لایه مخفی پیچیده‌تر است. در مرحله قبل مقدار خروجی نرون موجود در لایه خروجی  $N_{2,0}$  از قبل مشخص بود، اما در اینجا مقدار خروجی نرون‌های لایه مخفی یعنی نرون‌های  $N_{1,0}$  و  $N_{1,1}$  نامشخص می‌باشد.

در اینجا مقدار خطای  $N_{1,0}$  را محاسبه می‌نماییم. برای این منظور باید مقدار وزن جدید  $w_{1,0}$  را در خطای نرون  $N_{2,0}$  ضرب نماییم و مجدداً با همین روش خطای نرون  $N_{1,1}$  محاسبه می‌شود که محاسبه خطای هر دو نرون را در زیر مشاهده می‌نمایید.

$$N_{1,0} \text{ Error} = N_{2,0} \text{ Error} \times w_{1,0} \text{ New} = 0.133225 \times 0.097317 = 0.012965$$

$$N_{1,1} \text{ Error} = N_{2,0} \text{ Error} \times w_{1,1} \text{ New} = 0.133225 \times (-0.373012) = -0.049706$$

حال با توجه به مشخص شدن خطای نرون‌های لایه مخفی، وزن‌های بین لایه مخفی و لایه ورودی قابل بهروزرسانی می‌باشد. ابتدا باید نرخ تغییرات برای هر وزن محاسبه گردد.

$$\Delta w_{0,0} = \alpha \times N_{1,0} \text{ Error} \times N_{0,0} = 0.45 \times 0.012965 = 0.005834$$

$$\Delta w_{0,1} = \alpha \times N_{1,0} \text{ Error} \times N_{0,1} = 0.45 \times 0.012965 \times 1 = 0.005834$$

$$\Delta w_{0,2} = \alpha \times N_{1,1} \text{ Error} \times N_{0,0} = 0.45 \times -0.049706 \times 1 = -0.022368$$

$$\Delta w_{0,3} = \alpha \times N_{1,1} \text{ Error} \times N_{0,1} = 0.45 \times -0.049706 \times 1 = -0.022368$$

در ادامه با توجه به نتایج به دست آمده برای تغییرات وزن‌ها، اقدام به محاسبه مقدار وزن جدید بین نرون‌های ورودی و نرون‌های لایه میانی می‌نماییم.

$$w_{0,0} \text{ New} = w_{0,0} \text{ Old} + \Delta w_{0,0} + (\mu \times \Delta(t-1))$$

$$0.4 + 0.005834 + 0.9 \times 0 = 0.405834$$

$$w_{0,1} \text{ New} = w_{0,1} \text{ Old} + \Delta w_{0,1} + (\mu \times \Delta(t-1)) =$$

$$0.1 + 0.005834 + 0 = 0.105384$$

$$w_{0,2} \text{ New} = w_{0,2} \text{ Old} + \Delta w_{0,2} + (\mu \times \Delta(t-1)) =$$

$$-0.1 + -0.022368 + 0 = -0.122368$$

$$w_{0,3} \text{New} = w_{0,3} \text{Old} + \Delta w_{0,3} + (\mu \times \Delta (t - 1)) = \\ -0.1 + -0.022368 + 0 = -0.122368$$

#### ۵-۴-۴-۵- به روزرسانی وزن‌ها

نکته قابل توجه این است که تا وقتی تمامی خطاهای محاسبه نشده‌اند باید وزن‌ها را به روزرسانی نمود و اگر در زمان محاسبات خطای وزن جدید مورد استفاده قرار گیرد نتیجه محاسبات مقدار نادرستی خواهد بود.

حال بر اساس این توضیحات چنانچه در روابط زیر مشاهده می‌نمایید از وزن‌های جدید برای محاسبه مقدار نرون‌ها استفاده می‌شود تا میزان خطای مجدد بررسی شود و امید این است که با استفاده از وزن‌های جدید خطای نرون کاهش یافته باشد که برای این عمل دوباره مانند دور آموزشی اول ما مجددًا فرایند پیشخور و محاسبه مقدار نرون خروجی و محاسبه خطای آن را انجام می‌دهیم.

$$N_{1,0} = f(x1) = f(w_{0,0} \times N_{0,0} + w_{0,1} \times N_{0,1}) = \\ f(0.406 + 0.1) = f(0.506) = 0.623868314$$

$$N_{0,1} = f(x2) = f(w_{0,2} \times N_{0,0} + w_{0,3} \times N_{0,1}) = \\ f(-0.122 - 0.122) = f(-0.244) = 0.43930085 \\ N_{2,0} = f(x3) = f(w_{1,0} \times N_{1,0} + w_{1,1} \times N_{1,1}) = \\ f(0.097 \times 0.623868314 + (-0.373) \times 0.43930085) = \\ f(-0.103343991) = 0.474186972$$

چنانچه در محاسبات بالا مشاهده می‌نمایید مقدار نرون  $N_{2,0}$  به دست آمده برابر با 0.474186972 است و مرحله پیشخور انجام گردیده است و مرحله بعد به صورت مجدد محاسبه خطای نرون آخر  $N_{2,0}$  می‌باشد که در محاسبات زیر مشاهده می‌نمایید.

$$N_{2,0} \text{Error} = N_{2,0} \times (1 - N_{2,0}) \times (N_{2,0} \text{Desired} - N_{2,0}) = \\ 0.474186972 \times (1 - 0.474186972) \times (1 - 0.474186972) = 0.131102901$$

در اولین محاسبه خطای نرون خروجی مقدار خطای برابر با 0.133225 بود که پس از تغییر وزن‌ها و محاسبه مجدد خطای نرون خروجی مقدار به دست آمده برابر با 0.131102 می‌باشد و چنانچه مشاهده می‌نمایید مقدار خطای خروجی شبکه کاهش یافته است. اگرچه این میزان کاهش خطای بسیار ناجیز است اما با دورهای آموزشی زیاد خطای به تدریج به میزان قابل ملاحظه‌ای کاهش می‌یابد و در نهایت با کاهش خطای شبکه و رسیدن خطای به حداقل خود، شبکه قادر به تولید خروجی مورد انتظار خواهد بود.

## ۵-۵-مثال عملی برای دسته‌بندی براساس شبکه عصبی مصنوعی با زبان R

الگوریتم استفاده شده در این برنامه یک شبکه عصبی پرسپترون چند لایه است که برای دسته‌بندی مجموعه داده معروف iris مورد استفاده قرار گرفته است که این مجموعه داده در بخش رگرسیون معرفی گردید. به صورت خلاصه می‌توان گفت که مجموعه داده iris شامل ۵۰ نمونه از هر یک از سه گونه گل زنبق است که از هر نمونه چهار ویژگی بررسی گردید، این ویژگی‌ها به ترتیب مربوط به طول و عرض کاسبرگ و گلبرگ می‌باشند.

ابتدا بسته‌های نرم‌افزاری و مجموعه داده‌ها باید بارگذاری شوند که با تابع library() انجام می‌شود. بسته نرم‌افزاری مورد استفاده برای دسته‌بندی neuralnet است و خط دوم برنامه مربوط به بارگذاری بسته datasets است که شامل تعداد زیادی مجموعه داده است.

بسته نرم‌افزاری شبکه عصبی و مجموعه داده با دستور زیر بارگذاری می‌شود.

```
> library('neuralnet')
> library(datasets)
> data("iris")
```

با استفاده از دستور زیر می‌توانیم برچسب ستون‌ها که مربوط به ویژگی‌ها است را برای مجموعه داده iris مشخص نماییم. در اصل براساس خط برنامه زیر مجموعه داده iris با نام ستون‌های مشخص شده بارگذاری شده و در بخش environment قرار می‌گیرد.

```
> colnames(iris) = c("Sep.Length", "Sep.Width", "Petal.Length", "Petal.Width", "Class")
```

حال می‌توانیم شبکه عصبی را پیاده‌سازی کنیم، ولی ابتدا نیاز به تعیین تعداد لایه مخفی و تعداد نمونه‌های هر لایه داریم که خط برنامه زیر مشخص کننده دو لایه مخفی است که اولی ۳ نمون و دومی ۲ نمون درون خود جای داده است.

```
> hidd = c(3,2)
```

داده‌های مورد استفاده در مرحله آموزش به صورت زیر انتخاب می‌شود و به صورتی که مشاهده می‌نمایید از طریق تابع sample() تعداد ۱۰۰ نمونه از کل ۱۵۰ نمونه موجود در مجموعه داده iris، برای مرحله آموزش نمونه‌برداری و اختصاص داده شده و باقی نمونه‌ها برای مرحله آزمون مورد استفاده قرار می‌گیرد.

```
> samp = sample(1:150, 100)
> train<-iris[samp,]
> test<-iris[-samp,]
```

در خط زیر اسامی دسته‌ها و هر کدام از ویژگی‌های موجود در مجموعه داده‌های آموزشی را با یک متغیر نام‌گذاری می‌کنیم.

```
> a1<-train$Sep.Length;b<-train$Sep.Width;c<-train$Petal.Length;d<-train$Petal.Width
```

در سه خط برنامه زیر برچسب دسته‌ها در مجموعه داده آموزشی مشخص می‌شود.

```
> train$setosa = c(train$Class == "Iris-setosa")
> train$versicolor = c(train$Class == "Iris-versicolor")
> train$virginica = c(train$Class == "Iris-virginica")
```

حالا با توجه به اینکه شرایط برای اجرای شبکه عصبی آماده است با استفاده از خط برنامه زیر شبکه را آموزش می‌دهیم.

```
> net = neuralnet(setosa+versicolor+virginica~a1+b+c+d,train, hidden=hidd, lifesign="full")
```

اطلاعات اجرای الگوریتم به صورت زیر خواهد بود که اطلاعاتی از قبیل زمان اجرای الگوریتم، تعداد لایه‌های مخفی، میزان خطای الگوریتم در مرحله آموزش و غیره ارائه می‌نماید.

```
hidden: 3, 2    thresh: 0.01    rep: 1/1    steps: 38    error: 0    time: 0.02 secs
```

با استفاده از تابع `plot()` می‌توانید نتیجه نهایی الگوریتم را در مرحله آموزش و ایجاد مدل مشاهده نمایید.

```
> plot(net)
```

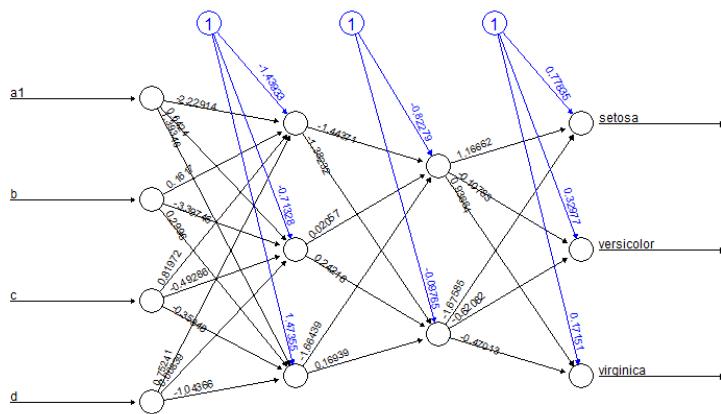
چنانچه در نمودار ایجاد شده توسط تابع `plot()` در زیر مشاهده می‌نمایید تعداد لایه‌های مخفی و تعداد نرون‌های آنها و تعداد نرون‌های خروجی که براساس تعداد دسته‌ها مشخص می‌شود و همچنین مقادیر وزن‌ها و بایاس‌ها و غیره مشخص گردیده است.

پس از آموزش شبکه باید اقدام به آزمون شبکه نماییم که برای اجرای این مرحله می‌توانیم از تابع `compute()` استفاده نماییم که باید از نمونه‌های آزمون که در متغیر `test` قرار دارد استفاده نماییم. توسط خط زیر تابع `compute()` با استفاده از داده‌های آزمون، عمل آزمون الگوریتم را انجام می‌دهد.

```
> comp <- compute(net, test[-5])
```

می‌توانیم با تایپ کلمه زیر نتیجه اجرای الگوریتم را برای ارزیابی عملکرد شبکه عصبی ایجاد شده مشاهده نماییم.

```
> comp
```



در زیر می‌توانید تعدادی از ردیف‌های هر بخش از خروجی مربوط به \$neurons و \$net.result به لایه‌های شبکه عصبی بوده و با اجرای تابع compute() ایجاد شده است را مشاهده نمایید.

\$neurons

\$neurons[[1]]

	1	Sep.Len	Sep.Wid	Pet.Len	Pet.Wid
3	1	4.7	3.2	1.3	0.2
10	1	4.9	3.1	1.5	0.1
13	1	4.8	3.0	1.4	0.1
17	1	5.4	3.9	1.3	0.4
18	1	5.1	3.5	1.4	0.3
20	1	5.1	3.8	1.5	0.3

....

\$neurons[[2]]

	[,1]	[,2]	[,3]	[,4]
3	1	0.000037816623801	0.000100537501287	0.9997532147
10	1	0.000026035622584	0.000145372933818	0.9998137335
13	1	0.000029495398757	0.000201147278280	0.9997871455
17	1	0.000010340330890	0.000014640012191	0.9999070309
18	1	0.000019045581840	0.000044695152262	0.9998513537
20	1	0.000021700042494	0.000015353934481	0.9998591709
21	1	0.000011388520536	0.000065602612775	0.9998988384

....

\$neurons[[3]]

	[,1]	[,2]	[,3]
3	1	0.07678744530	0.5179086149
10	1	0.07678157600	0.5179179514
13	1	0.07678444021	0.5179190053
17	1	0.07677198475	0.5179194093
18	1	0.07677770613	0.5179158674
20	1	0.07677646947	0.5179135077

....

\$net.result

	[,1]	[,2]	[,3]
3	-0.0000078787088017	-0.000035449600845	-0.0000534897972245
10	-0.0000303726032760	-0.000040613022447	-0.0000633765579193
13	-0.0000287973495686	-0.000041576165885	-0.0000611893050971
17	-0.0000440050634709	-0.000040483824206	-0.0000730454431999
18	-0.0000313947179498	-0.000038901903133	-0.0000660214508893
20	-0.0000288829343148	-0.000037303599951	-0.0000660703977663

....

در انتهای این برنامه می‌توانید برای مشاهده نتیجه دسته‌بندی مجموعه داده iris براساس داده‌های آزمون و مشاهده وضعیت دسته‌بندی از خطوط زیر استفاده نمایید.

```
> pred.weights <- comp$net.result
> idx <- apply(pred.weights, 1, which.max)
> pred <- c('setosa','versicolor','virginica')[idx]
> table(pred,test$Class)
```

جدول زیر مشخص کننده نتیجه دسته‌بندی برای داده‌های آزمون است که در آن ۱۰ نمونه در دسته setosa، ۲ نمونه در دسته versicolor و ۱ نمونه در دسته virginica به درستی قرار گرفته است. توجه به این نکته ضروری است که به دلیل نمونه‌برداری تصادفی از داده‌ها از مجموعه iris و اختصاص آنها به مجموعه داده‌های آموزشی و آزمون، ممکن است که در نمونه‌برداری هیچ داده‌ای که مربوط به یکی از دسته‌ها است انتخاب نشده باشد که در مرحله آزمون الگوریتم در دسته مربوطه قرار گیرد و همچنین ممکن است در هر بار اجرای این الگوریتم با خروجی‌های متفاوتی مواجه شوید. به این نکته ضروری توجه کنید که خروجی زیر، یک ماتریس است که براساس آن می‌توان کیفیت عملکرد الگوریتم را نیز ارزیابی نمود. برای اطلاع از نحوه انجام این کار باید به فصل ۱۳ مراجعه نمایید.

classify	setosa	versicolor	virginica
setosa	10	13	17
versicolor	6	2	0
virginica	0	1	1

## ۶-۵- مثال عملی برای دسته‌بندی براساس شبکه عصبی مصنوعی با زبان پایتون

در چهار خط اول کتابخانه‌های مورد استفاده مشخص می‌شوند.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
```

در سه خط زیر مجموعه داده iris به برنامه افزوده می‌شود که تنها دو ویژگی که همان septal-width و septal-lenth می‌باشد در اینجا مدنظر قرار دارد.

```
iris = datasets.load_iris()
```

```
X = iris.data[:, :2]
```

```
Y = iris.target
```

$h = .02$

در خط زیر پارامترهای مشخص کننده دسته بند مشخص می‌گردد که شیء Classifier را تولید نماید.

```
Classifier = MLPClassifier(activation='relu',
                           hidden_layer_sizes=(5, 2),
```

```
learning_rate_init=0.001,
max_iter=200, momentum=0.9,
random_state=1,
solver='lbfgs')
```

خط زیر با استفاده از Classifier دسته بند را برای دریافت داده و دسته بندی آنها ایجاد می نماید تا مدل آموزش دیده را ایجاد نماید.

`Classifier.fit(X, Y)`

در زیر تخمین نهایی دسته داده های آزمایشی با استفاده از Classifier.predict() انجام می شود و نمودار خط تصمیم ترسیم می شود. برای این منظور برای هر نقطه داده موجود در نمودار یک رنگ مشخص کننده دسته مورد نظر اختصاص داده می شود.

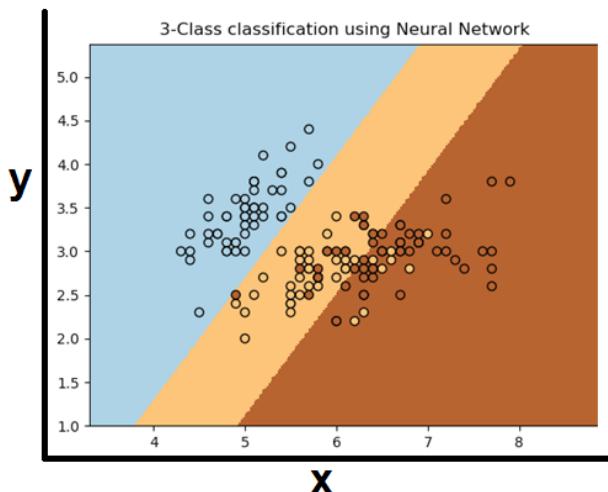
```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = Classifier.predict(np.c_[xx.ravel(), yy.ravel()])
```

خطوط زیر نقاط داده ای را به صورت رنگی در می آورند.

```
Z = Z.reshape(xx.shape)
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
```

خطوط زیر نمودار نهایی را در خروجی چاپ می نمایند که خط اول کار ترسیم بلوک های رنگی در نمودار را به عهده دارد.

```
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired, edgecolors='k')
plt.title('3-Class classification using Neural Network')
plt.axis('tight')
plt.show()
```



۹

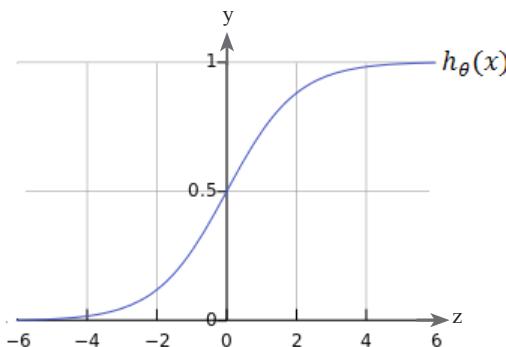
فصل

## ماشین بردار پشتیبان

## ۱-۶- ماشین بردار پشتیبان<sup>۱</sup>

ماشین بردار پشتیبان یکی از پرکاربردترین الگوریتم‌های با نظارت برای دسته‌بندی و پیش‌بینی می‌باشد که هم در صنعت و هم در مراکز تحقیقاتی به مقدار بسیار زیادی قابل قبول است و همچنین در مقایسه با شبکه عصبی و رگرسیون لاجستیک با توجه به صورت مسئله و شرایط موجود می‌تواند راه حل بسیار دقیق‌تر و بهتری را ارائه دهد. قبل از ورود به مباحث مربوط به ماشین بردار پشتیبان می‌خواهیم مروری درباره رگرسیون لاجستیک داشته باشیم زیرا به دلیل شباهت تابع هزینه این دو الگوریتم، با اندکی تغییر تابع هزینه رگرسیون لاجستیک به روابطی که تابع هزینه ماشین بردار پشتیبان را شکل می‌دهد خواهیم رسید.

برای رگرسیون لاجستیک چنانچه درگذشته با آن آشنا شدیم خط فرضیه به صورت  $h_{\theta}(x) = g(\theta^T X)$  می‌باشد و تابع حد آستانه<sup>۲</sup> آن یک تابع سیگموئید است که در شکل (۱-۶) مشاهده می‌نمایید.



شکل (۱-۶): تابع سیگموئید.

شرط زیر برای رگرسیون لاجستیک باید برقرار شود.

$$h_{\theta}(x) = \begin{cases} y = 1 & \text{if } \theta^T x \geq 0 \\ y = 0 & \text{if } \theta^T x < 0 \end{cases}$$

در صورتی که  $\theta^T x \geq 0$  باشد خط فرضیه به صورت  $y \approx 1$  خواهد بود به عبارتی هرچه  $Z = \theta^T x$  بزرگ‌تر از صفر باشد خروجی رگرسیون لاجستیک به یک نزدیک‌تر خواهد بود. و هرچه  $Z$  کوچک‌تر از صفر باشد خروجی رگرسیون لاجستیک به یک نزدیک‌تر خواهد شد.

براساس مطالب ارائه شده در فصل‌های قبل تابع هزینه رگرسیون لاجستیک به صورت رابطه (۱) می‌باشد.

$$\begin{aligned} J_{\theta} &= (y \log h_{\theta}(x) + (1-y) \log (1 - h_{\theta}(x))) = \\ &= -y \log \frac{1}{1 + e^{-\theta^T x}} - (1-y) \log \left( \frac{1}{1 + e^{-\theta^T x}} \right) \end{aligned} \quad (\text{رابطه } 1)$$

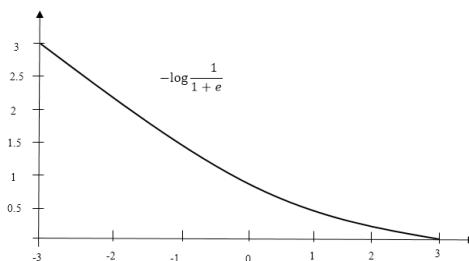
1. Support vector machine  
2. Threshold

حال در ادامه خواهیم دید که وقتی  $Y=1$  و در حالت عکس آن یعنی وقتی  $Y=0$  باشد روابطه مربوطه به چه صورت خواهد بود.

اگر  $Y=1$  باشد و ما عدد یک را در رابطه (۱) جایگذاری کنیم رابطه (۲) را خواهیم داشت.

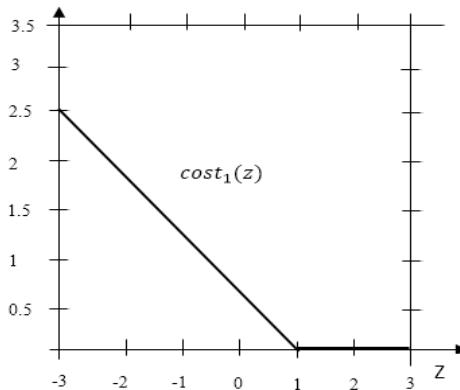
$$J_\theta = -1 \log \frac{1}{1+e^{-z}} - (1-1) \log \frac{1}{1+e^{-z}} = \log \frac{1}{1+e^{-z}} \quad \text{رابطه (۲)}$$

و هرچه  $Z$  از صفر بزرگ‌تر باشد تابع هزینه مقدار کوچک‌تری خواهد داشت که می‌توان نمودار آن را در شکل (۲-۷) مشاهده نمود.



شکل (۲-۶): نمودار تابع هزینه در رگرسیون لاجستیک در صورتی که  $y \approx 1$  باشد.

حال با توجه به نمودار بالا که برای رگرسیون لاجستیک بود، برای ماشین بردار پشتیبان تابع هزینه را مقداری تغییر می‌دهیم تا نمودار شکل (۲-۶) به صورتی که در نمودار شکل (۳-۶) مشاهده می‌کنید تغییر کند.

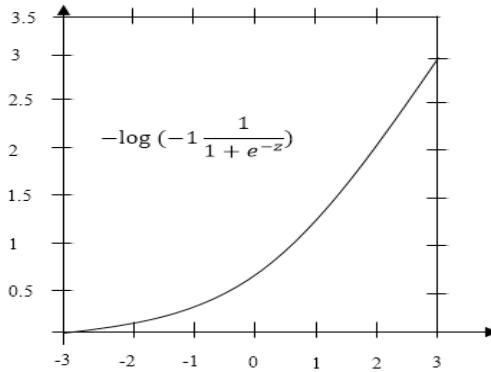


شکل (۶-۳): نمودار تابع هزینه ماشین بردار پشتیبان در شرایطی که  $y=1$  باشد.

همان‌طور که مشاهده می‌شود برای ماشین بردار پشتیبان تابع هزینه مانند یک خط خمیده نیست و از نقطه یک بر روی محور  $Z$  به بعد، مقدار تابع هزینه به صفر رسیده است.

حال برای حالتی که در رگرسیون لاجستیک  $Y=0$  در نظر گرفته شده باشد. رابطه (۳) را داریم و نمودار تابع هزینه به صورت شکل (۴-۶) خواهد بود.

$$= -y \log \frac{1}{1+e^{-z}} - (1-y) \log (1 - \frac{1}{1+e^{-z}}) \quad \text{رابطه (۳)}$$



شکل (۶-۴): تابع هزینه رگرسیون لاجستیک در شرایطی که  $y=0$  باشد.

تابع هزینه رگرسیون لاجستیک به صورت زیر است.

$$4) J_{\theta} = \min_{\theta} \frac{1}{m} [\sum_{i=1}^m y^{(i)} (-\log h_{\theta}(x^{(i)})) + (1-y^{(i)}) (1 - \log (1 - h_{\theta}(x^{(i)})))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

اگر رابطه (۴) را به دو بخش A و B که در زیر مشاهده می‌نمایید تقسیم کنیم می‌توانیم آن را به صورت رابطه (۵) فرض کنیم.

$$A = \frac{1}{m} [\sum_{i=1}^m y^{(i)} (-\log h_{\theta}(x^{(i)})) + (1-y^{(i)}) (1 - \log (1 - h_{\theta}(x^{(i)})))]$$

$$B = \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

$$J_{\theta} = A + B \quad \text{رابطه (۵)}$$

چنانچه در رگرسیون لاجستیک مشاهده نمودید قسمت A مربوط به تابع هزینه و قسمت B مربوط به رابطه تنظیم<sup>۱</sup> می‌باشد که در آن با تغییر مقدار  $\lambda$  می‌توان به مقدار مناسب  $\theta$  رسید.

حال براساس توضیحات داده شده به رابطه (۶) که برای به دست آوردن مقدار بهینه  $\theta$  در ماشین بردار پشتیبان است توجه نمایید.

## ۶ - ۲ - تابع هزینه ماشین بردار پشتیبان

رابطه تابع هزینه ماشین بردار پشتیبان به صورت زیر می‌باشد.

$$j_{\theta} = \min_{\theta} C \sum_{i=1}^m [y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T x^{(i)})] + \frac{1}{2} \sum_{j=1}^n \theta_j^2 \quad \text{رابطه (۶)}$$

اگر رابطه (۶) را به دو بخش A و B که در زیر مشاهده می‌نمایید تقسیم کنیم می‌توانیم آن را به صورت رابطه (۷) فرض کنیم.

$$A = \sum_{i=1}^m [y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)})]$$

$$B = \frac{1}{2} \sum_{i=1}^n \theta_j^2$$

$$J_{\theta} = CA + B \quad \text{رابطه (۷)}$$

همان‌طور که در رگرسیون لاجستیک با اختصاص مقدار مناسب به  $\lambda$  مقدار B را که مربوط به تنظیم‌کننده است به مقدار کوچکتری می‌رسانیم تا بتوانیم مقدار A را کاهش دهیم در بخش ماشین بردار پشتیبان با اختصاص مقدار مناسب به C این کار را انجام می‌دهیم.

اختصاص مقدار مناسب به C باعث می‌شود تا بتوانیم مقدار مناسب برای  $J(\theta)$  به دست آوریم. هر چه مقدار C را بیشتر کنیم در مرحله آموزش با خطای آموزشی کمتری روبرو می‌شویم و دقت الگوریتم در دسته‌بندی داده‌های آموزشی بالا می‌رود که بیش از حد بالا رفتن دقت الگوریتم درست نیست و همچنین بالا بودن مقدار C زمان اجرای الگوریتم را نیز در مرحله آموزش بالا می‌برد. اما اگر مقدار C را بیشتر از حد مناسب کم کنیم دقت الگوریتم برای دسته‌بندی داده‌های آموزشی را پایین می‌آوریم.

#### ۴-۳- دسته‌بندی بر مبنای بیشترین حاشیه<sup>۱</sup>

بعضی وقت‌ها به ماشین بردار پشتیبان عنوان دسته‌بندی کننده از روی بیشترین حد فاصله گفته می‌شود که در ادامه مفهوم آن را شرح خواهیم داد.

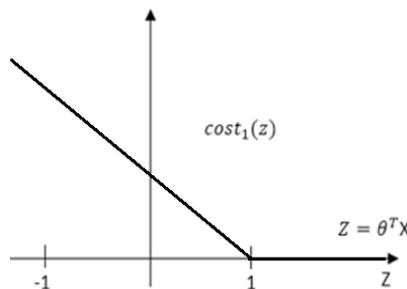
#### ۴-۴- نمودار تابع هزینه در ماشین بردار پشتیبان

با توجه به رابطه (۶) که برای ماشین بردار پشتیبان است رابطه زیر مفروض است.

$$cost(\theta^T x^{(i)}) = cost(z)$$

حال برای نمونه‌های مثبت نمودار تابع هزینه به صورت شکل (۵-۶) می‌باشد.

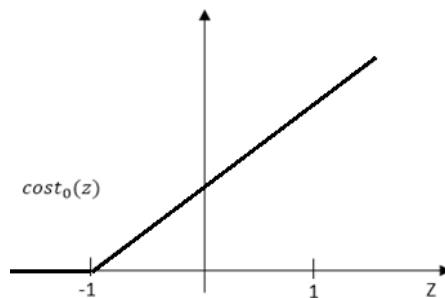
1. Large margin



شکل (۶-۵): تابع هزینه برای نمونه‌های مثبت در ماشین بردار پشتیبان.

اگر  $y=1$  باشد  $\theta^T x \geq 1$  خواهد بود یعنی برای  $\theta^T x$  مقدار  $cost_1(z)$  بزرگ‌تر از یک می‌باشد.

همچنین برای نمونه‌های منفی نمودار تابع هزینه به صورت شکل (۶-۶) می‌باشد اگر  $y=0$  باشد  $\theta^T x \leq -1$  خواهد بود یعنی برای  $\theta^T x$  مقدار  $cost_0(z)$  کوچک‌تر از منفی یک خواهد بود.



شکل (۶-۶): تابع هزینه برای نمونه‌های منفی در ماشین بردار پشتیبان.

به صورت کلی می‌توان گفت که اگر  $y=0$  باشد  $\theta^T x \leq -1$  خواهد بود یعنی برای  $\theta^T x$  مقدار  $cost_0(z)$  کوچک‌تر از منفی یک خواهد بود.

در اصل شرط‌های  $\theta^T x \geq 1$  و  $\theta^T x \leq -1$  به این دلیل برقرار است که هدف ما در ماشین بردار پشتیبان ایجاد حد فاصله بیشتر و مناسب برای حاشیه بین خط تصمیم و داده‌های کلاس‌هایی می‌باشد که می‌خواهیم از هم جدا کنیم و براساس شرط‌های  $\theta^T x \geq 0$  و  $\theta^T x < 0$  نمی‌توان حاشیه مناسب ایجاد کرد.

فرض کنید  $C$  مقدارش زیاد باشد یعنی حد فاصله حاشیه را عدد بالایی در نظر بگیریم در این حالت باید تلاش کنیم تا قسمت اول تابع هزینه که به صورت زیر می‌باشد مساوی صفر شود که در این شرایط  $J_{\theta}$  و  $y^{(i)}$  به صورت رابطه (۸) می‌شود.

$$\min_{\theta} \sum_{i=1}^m [y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)})] = 0$$

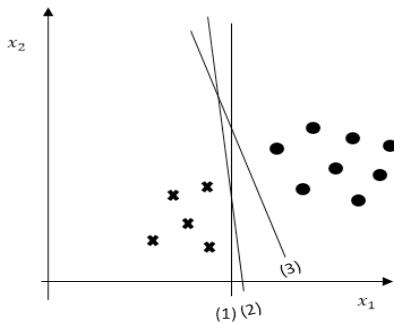
$$J_{\theta} = C \times 0 + \frac{1}{2} \sum_{j=1}^m \theta_j^2 = \frac{1}{2} \sum_{j=1}^m \theta_j^2 \quad (8)$$

$$J_{\theta} = \frac{1}{2} \sum_{j=1}^m \theta_j^2$$

$$y^{(i)} = \begin{cases} 1 & \text{if } \theta^T x^{(i)} \geq 1 \\ 0 & \text{if } \theta^T x^{(i)} \leq -1 \end{cases}$$

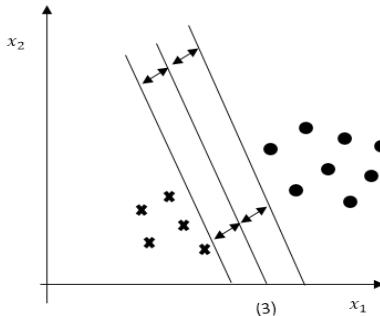
## ۶-۵- ایجاد خط تصمیم برای تفکیک دسته‌ها

فرض کنید دو نوع مختلف نمونه‌های آموزشی برچسبدار را چنانچه در شکل (۷-۶) مشاهده می‌نمایید داشته باشیم که می‌خواهیم از هم جدا کنیم و در دسته‌های متفاوت قرار دهیم.



شکل (۷-۶): نمونه‌های آموزشی با سه حالت مختلف قرارگیری خط تصمیم برای دسته‌بندی.

همان‌طور که در شکل (۷-۶) مشاهده می‌نمایید خط تصمیم (۱) به درستی عمل می‌کند و همچنین خط تصمیم شماره (۲) هم می‌تواند مناسب‌تر باشد، ولی برای ماشین بردار پشتیبان بهترین حالت قرارگیری خط تصمیم می‌تواند خط شماره (۳) باشد به این دلیل که خط (۳) دارای حاشیه زیادی از تمامی داده‌های هر دو کلاس نسبت به خطوط دیگر است که می‌توانید خط تصمیم شماره (۳) را با حاشیه مربوط در شکل (۸-۶) مشاهده نمایید.



شکل (۸-۶): خط تصمیم ماشین بردار پشتیبان با حاشیه مناسب.

در اصل اندازه فاصله حاشیه خط تصمیم که مورد بحث ما است همان فاصله بین خط<sup>(۳)</sup> و خطوط کناری آن است. در نهایت می‌توان گفت ماشین بردار پشتیبان تلاش می‌کند تا بیشترین حد فاصله موجود بین دو دسته را به عنوان خط تصمیم در نظر بگیرد البته این نکته را نیز باید دانست که ماشین بردار پشتیبان پیچیده‌تر و حساب‌شده‌تر از آن است که آن را فقط یک دسته‌بند بر مبنای فاصله حاشیه زیاد در نظر گرفت و اگر ما فقط فاصله حاشیه زیاد را در نظر بگیریم در این حالت این الگوریتم به داده‌های پرت<sup>۱</sup> حساسیت بیشتری نشان می‌دهد.

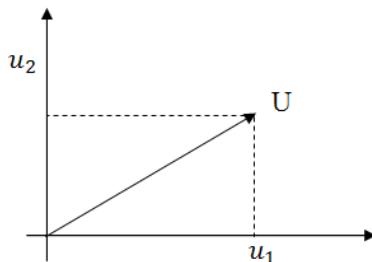
در نظر داشته باشید که ترسیم و قرارگیری خط تصمیم بایستی بر این اساس باشد که الگوریتم هم بتواند بیشترین حاشیه ممکن را برای تفکیک دسته‌ها داشته باشد و هم دقیق‌تر الگوریتم را بیش از حد فدای بالا بودن حاشیه خط تصمیم نکند.

#### ۶-۴- مباحث ریاضی مربوط به دسته‌بندی بر اساس حاشیه زیاد

برای اینکه بتوانید رابطه مربوط به دسته‌بندی بر اساس حد فاصله حاشیه خط تصمیم را متوجه شوید ابتدا باید با برخی از خصوصیات برداری آشنا شوید.

فرض کنید دو بردار  $V = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$  و  $U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$  داریم که هر کدام یک بردار دو بعدی می‌باشد بر این اساس می‌خواهیم بینیم که مفهوم رابطه زیر چیست؟  $U^T V = ?$

می‌توان گفت که  $U^T V$  ضرب داخلی<sup>۲</sup> بردار  $U$  و  $V$  می‌باشد و نمودار بردار  $U$  که یک بردار دو بعدی است به صورت شکل (۹-۶) است.



شکل (۹-۶): نمودار دو بعدی بردار  $U$ .

چنانچه مشاهده می‌شود بردار  $U$  در دو بعد  $u_1$  و  $u_2$  که با خطوط مقطع نشان داده شده است که در یک نقطه مشترک تلاقی دارند که تشکیل دهنده بردار  $U$  می‌باشند و می‌توانند دارای هر مقداری باشند. به عبارتی یک بردار دو بعدی از یک نقطه‌ای مانند صفر شروع و تا نقطه  $U$  امتداد دارد.

$$U = (u_1, u_2), \quad u \neq 0, \quad u = \mathbb{R}^2$$

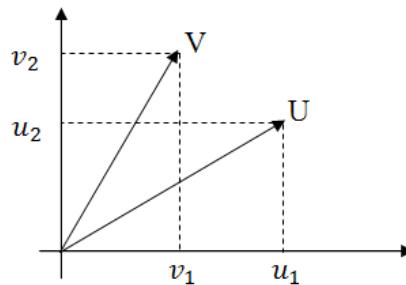
1. Out layer  
2. Inner product

## ۷-۶- مقیاس یک بردار

مقیاس یک بردار مانند  $U$  را به صورت  $\|U\|$  نشان می‌دهیم که مقدار آن با رابطه (۹) که به صورت زیر می‌باشد به دست می‌آید.

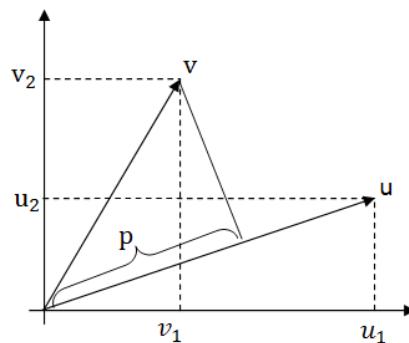
$$\|U\| = \sqrt{u_1^2 + u_2^2} \in R \quad \text{رابطه (۹)}$$

مثال) چنانچه در شکل (۱۰-۶) مشاهده می‌نمایید دو بردار  $U$  و  $V$  وجود دارد که می‌خواهیم ضرب داخلی آنها را به دست آوریم.



شکل (۱۰-۶): نمودار دو بردار دو بعدی  $U$  و  $V$ .

برای محاسبه ضرب داخلی چنانچه در شکل (۱۱-۶) مشاهده می‌نمایید از بردار  $V$  یک خط راست که به صورت عمود یا نود درجه به بردار دیگر یعنی  $U$  رسم می‌کنیم و فاصله بین مبدأ مختصات و نقطه اتصال خط عمود بر روی بردار  $U$  را محاسبه می‌کنیم که آن فاصله را با حرف  $p$  نشان می‌دهیم.



شکل (۱۱-۶): نمودار نحوه محاسبه ضرب داخلی دو بردار دو بعدی.

به عبارتی  $p$  همان اندازه افکنش<sup>۲</sup> بردار  $V$  بروی بردار  $U$  می‌باشد و بر آن اساس رابطه (۱) را به صورت زیر خواهیم داشت.

$$U^T V = P, \|U\|, P \in R \quad \text{رابطه (۱۰)}$$

1. Norm

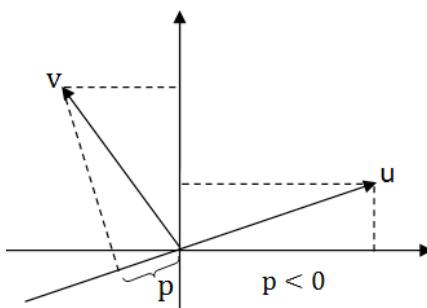
2. Projection

وهمچنین می‌توان با رابطه (۱۱) نیز به جواب مشابه رسید.

$$U^T V = u_1 v_1 + u_2 v_2 \leftrightarrow [u_1 \ u_2] \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \quad \text{رابطه (۱۱)}$$

به این نکته توجه کنید که جواب  $U^T V$  دقیقاً با  $V^T U$  مشابه است پس می‌توان گفت که  $V^T U = P. \|V\|$  خواهد بود.

اگر زاویه بین بردار  $U$  و بردار  $V$  از  $90^\circ$  درجه بیشتر باشد و  $U$  در ناحیه مثبت قرار گرفته باشد  $P$  یک عدد منفی خواهد بود برای این موضوع به شکل (۱۲-۶) توجه نمایید.



شکل (۱۲-۶): نمودار ضرب داخلی وقتی زاویه دو بردار بیش از  $90^\circ$  درجه باشد.

بر اساس شکل (۱۲-۶) بردار  $V$  در منطقه منفی قرار دارد و در حالتی مانند این که زاویه بین نقطه آغازین دو بردار از  $90^\circ$  درجه بیشتر باشد برای انجام عمل افکنش بردار  $V$  به روی بردار  $U$ ، بردار  $U$  را در منطقه منفی امتداد می‌دهیم و دوباره خط عمود از  $V$  به امتداد  $U$  مانند شکل فوق رسم می‌کنیم و  $P$  را به دست می‌آوریم.

#### ۴-۱-۱- مباحث ریاضی خط تصمیمی

چنانچه در مباحث ابتدایی این فصل گفته شد با استفاده از رابطه (۸) می‌توانیم خط تصمیم را در ماشین بردار پشتیبان به وجود آوریم. در ادامه مطالب با توجه به توضیحاتی که درباره ضرب داخلی بردار ارائه شد، خواهید دید که چگونه یک خط تصمیمی، در یک فضای نمونه‌ها، با استفاده از رابطه مربوط به ماشین بردار پشتیبان ترسیم می‌شود.

در ادامه برای درک بهتر و راحت‌تر مباحث  $\theta_0 = 0$  قرار داده می‌شود که هر مثال آموزشی تنها دارای دو ویژگی  $\theta_1, \theta_2$  باشد تا بتوان آن را به صورت نمودار به راحتی به تصویر کشید. بر این اساس رابطه زیر را خواهیم داشت.

$$\min_{\theta} \frac{1}{2} \sum_{i=1}^n \theta_j^2 = \frac{1}{2} (\theta_1^2 + \theta_2^2) = \frac{1}{2} (\sqrt{\theta_1^2 + \theta_2^2})^2 = \frac{1}{2} \|\theta\|^2$$

از آن جایی که  $\theta_0 = 0$  باشد در نهایت بردار نهایی به صورت  $\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$  برابر با  $\sqrt{\theta_1^2 + \theta_2^2}$  است و

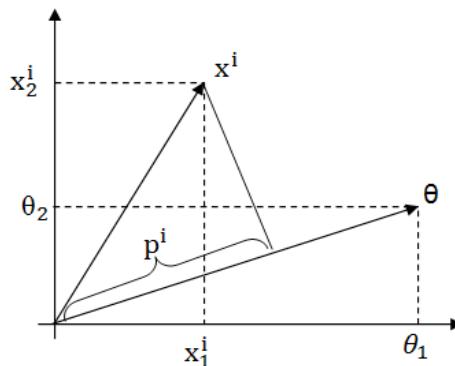
آمده است و با توجه به  $\theta_0 = 0$ , محل عبور خط تصمیم از مبدأ مختصات است.

در ادامه به تشریح مفهوم  $\theta^T X^{(i)}$  که در شرط رابطه (۸) مشاهده نمودید، خواهیم پرداخت.

$$\theta^T X^{(i)} = ?$$

همان‌گونه که برای  $U^T V$  نشان دادیم که چگونه می‌توان بردار  $V$  را بروی بردار  $U$  افکنش<sup>۱</sup> کرد برای نیز به همان صورت عمل می‌کنیم با این تفاوت که دو بردار ما  $\theta$  و  $x^{(i)}$  می‌باشند.

برای درک بهتر به شکل (۱۳-۶) زیر توجه شود.



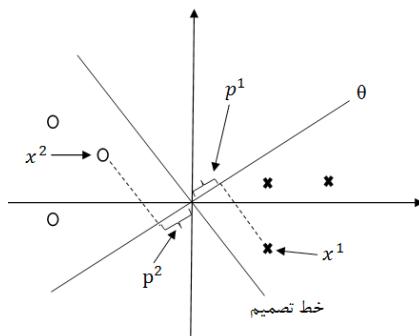
شکل (۱۳-۶): افکنش بردار  $x$  بر روی بردار  $\theta$ .

در اصل با افکنش  $x^{(i)}$  بر روی  $\theta$  مقدار  $P^{(i)}$  را به دست می‌آوریم که  $P^{(i)}$  نشان‌دهنده طول بخشی است که از افکنش  $x$  آمین نمونه آموزشی بر روی بردار  $\theta$  به دست آمده است یا به عبارتی دیگر اندازه حاشیه مثال‌های آموزشی مثبت و یا حاشیه مثال‌های منفی را مشخص می‌کند که در نهایت رابطه (۱۲) را خواهیم داشت.

$$\theta^T X^{(i)} = \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \quad \text{یا} \quad \theta^T X^{(i)} = p^{(i)} \cdot \|\theta\| \quad (12)$$

برای قرار گرفتن خط تصمیم در منطقه مناسب برای جدا کردن دو دسته در یک مسئله خطی به مثال زیر توجه نمایید.

مثال) چنانچه در شکل (۱۴-۶) مشاهده می‌نمایید فرض کنید دو مجموعه داده برچسب‌دار داریم که می‌خواهیم با ماشین بردار پشتیبان با ترسیم یک خط تصمیم، آنها را از هم جدا کنیم.



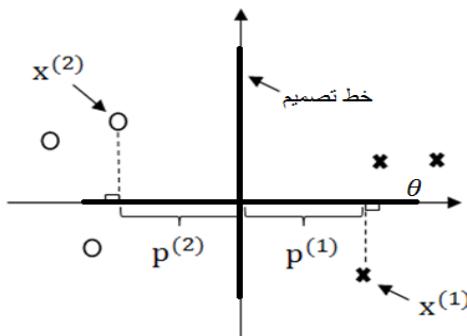
شکل (۱۴-۶): رسم خط تصمیم و حاشیه‌های مجاور در حالت نامناسب.

فرض بر این است که  $p^{(i)}$  یک افکنش از  $X$  بروی بردار  $\theta$  می‌باشد که در آن  $\theta_0$  برابر صفر است. همان‌گونه که در شکل (۱۴-۶) ملاحظه می‌شود  $x^1$  بر روی بردار  $\theta$  افکنش شده و به دلیل قرار گرفتن  $p^1$  در منطقه مثبت رابطه  $1 \geq \|p^1\| \|\theta\|$  برقرار است و به دلیل کوچک بودن  $p^1$  باید مقدار  $\|\theta\|$  یک عدد بزرگ باشد.

و با توجه به افکنش  $x^2$  بروی  $\theta$  و با توجه به قوانینی که در مطالب قبلی اشاره شد  $p^2$  در منطقه منفی قرار دارد و رابطه شرطی  $-1 \leq \|p^2\| \|\theta\|$  برای این مثال آموزشی برقرار است و در این حالت نیز به دلیل اینکه  $p^2$  یک مقدار منفی کوچک است پس  $\|\theta\|$  مقدارش بزرگ می‌باشد.

اما هدف ما ایجاد شرایطی است که بتوان مقدار بهینه را به دست آورد و برای همین پارامتر  $\|\theta\|$  باید مقدار کوچکی داشته باشد.

در ادامه می‌خواهیم برای مسئله قبلی با همان شرایط یکسان یک خط تصمیم متفاوت در جهت دیگری ترسیم کنیم که به شکل (۱۵-۶) توجه نمایید.



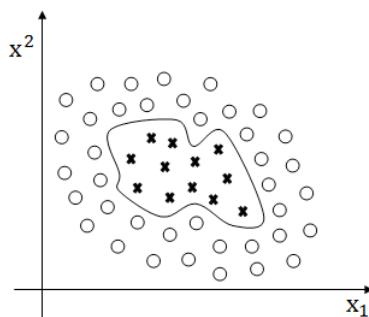
شکل (۱۵-۶): رسم خط تصمیم و حاشیه‌های مجاور. در حالت مناسب.

توجه به این نکته ضروری است که یک زاویه  $90^\circ$  درجه بین  $\theta$  و خط تصمیم وجود دارد و بردار  $\theta$  بر خط تصمیم همیشه عمود است.

در این مثال  $p^{(2)}$  در منطقه منفی قرار دارد پس شرط  $-1 \leq p^2 \cdot \|\theta\|$  برقرار است و به دلیل بزرگ بودن اندازه  $p^{(2)}$  مقدار  $\|\theta\|$  بسیار کوچک خواهد بود و برای  $p^{(1)}$  نیز  $0 \geq p^1 \cdot \|\theta\|$  برقرار است و پارامتر  $\|\theta\|$  بسیار کوچک و  $p^{(1)}$  مقدار بزرگی دارد و نشان‌دهنده آن است که حد فاصل بین خط تصمیم و نمونه‌های آموزشی براساس اندازه  $p^{(1)}$  و  $p^{(2)}$  زیاد است و در ماشین بردار پشتیبان این یک خط تصمیم مناسبی می‌باشد. بنابراین  $p^{(1)}$  به عنوان اندازه حاشیه مثال‌های آموزشی مثبت و  $p^{(2)}$  به عنوان اندازه حاشیه مثال‌های آموزشی منفی که در دو طرف خط تصمیم به صورت موازی با آن قرار دارند دو کلاس منفی و مثبت را با دقت بالایی از هم جدا می‌کند.

## ۹-۶- تابع هسته<sup>۱</sup> ماشین بردار پشتیبان

در مباحث قبل در مورد چگونگی جدا کردن دو دسته با خط تصمیم، که به صورت خطی جدایی‌پذیر بودند بحث شد اما بیشتر مسائل واقعی مانند شکل (۱۶-۶) به صورت خطی جدایی‌پذیر نمی‌باشند و ماشین بردار پشتیبان با استفاده از روش‌هایی قادر به انجام این تفکیک غیرخطی به بهترین نحو می‌باشد.



شکل (۱۶-۶): نمودار نمونه‌های آموزشی مربوط به دو کلاس که به صورت خطی جدایی‌پذیر نمی‌باشند.

شکل (۱۶-۶) نشان می‌دهد که چگونه دو گروه متفاوت از داده‌های برچسبدار با استفاده از یک مرز تصمیم غیرخطی از هم جدا شده‌اند که رابطه شرطی آن هم به صورت فرضی به شکل زیر می‌باشد که یک رابطه چندجمله‌ای با درجه بالاست.

$$y = \begin{cases} 1 & \text{if } (h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 \dots) \geq 0 \\ 0 & \text{if } (h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 \dots) < 0 \end{cases}$$

توجه داشته باشید که برای راحتی کار و درک راحت‌تر در ادامه برای ویژگی بجای استفاده از  $x$  در رابطه چندجمله‌ای از  $f_i$  استفاده می‌شود یعنی ویژگی‌ها را به صورت زیر نشان می‌دهیم:

$$f_1 = x_1, \quad f_2 = x_2, \quad f_3 = x_1x_2, \quad f_4 = x_1^2, \dots$$

$$h_{\theta}(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_1 f_2 \quad \text{مثال)$$

چنانچه بیان شد در بیشتر مواقع برای دسته‌بندی به روش‌های غیرخطی نیاز است که برای این کار می‌توان از تابع کرنل<sup>۱</sup> استفاده کرد.

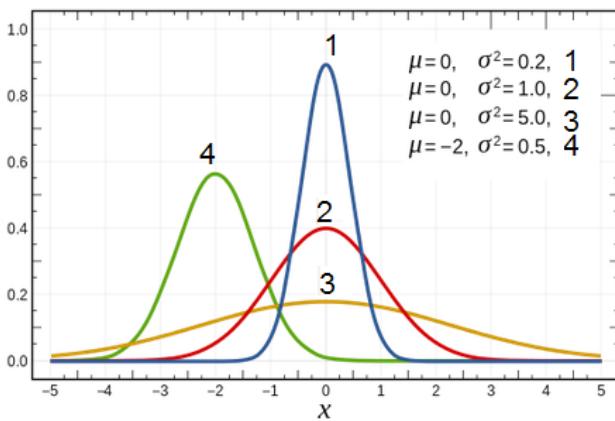
#### ۶-۹-۱- معرفی تابع هسته و عملکرد آنها

توابع هسته مختلفی برای ماشین بردار پشتیبان وجود دارد. وقتی ماشین بردار پشتیبان از هسته استفاده نمی‌کند به آن ماشین بردار پشتیبان با هسته خطی<sup>۲</sup> گویند ولی در شرایطی که مسئله نیاز به یک جداینده غیرخطی داشته باشد در این صورت از توابع هسته‌ای مانند هسته گوسی<sup>۳</sup> که به تابع *similarity* () یا تابع شباهت نیز معروف می‌باشد استفاده می‌کند.

#### ۶-۹-۲- تابع هسته گوسی<sup>۴</sup>

تابع گوسی یک تابع نمایی است که رابطه ریاضی آن به صورت رابطه (۱۳) تعریف می‌شود که در آن  $a$  و  $b$  و  $c$  ضرایب ثابت حقیقی و  $e$  عدد اوپلر است. ثابت  $a$  تعیین‌کننده ارتفاع قله منحنی،  $b$  تعیین‌کننده محل مرکز قله و  $c$  انحراف معیار که میزان کشیدگی یا پهن شدگی زنگوله را نشان می‌دهد که شکل این تابع به صورت شکل

$$F(x) = a e^{-\frac{(x-b)^2}{2c^2}} \quad \text{رابطه (۱۳) می‌باشد.}$$

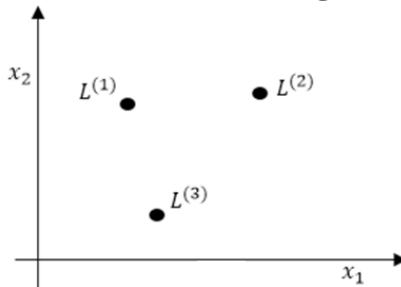


شکل (۱۷-۶): تابع گوسی.

- 
1. Kernel function
  2. Linear kernel
  3. Gaussian kernel
  4. Gaussian function

## ۶-۹-۱-۲- تابع گوسی در ماشین بردار پشتیبان

با توجه به توضیحات ریاضی تابع گوسی، رابطه (۱۴) برای ایجاد کرنل گوسی برای ماشین بردار پشتیبان می‌باشد که بر آن اساس شباهت هر نمونه آموزشی با نشانه<sup>۱</sup> مورد نظر مشخص می‌شود تا بتوانیم ویژگی‌های مناسب را استفاده کنیم. به شکل (۱۸-۶) که نشان‌دهنده یک فضای ویژگی‌ها است توجه کنید، سه نقطه به عنوان نقطه نشانه در آن وجود دارد، در حال حاضر فرض بر آن است که این نشانه‌ها براساس روش صحیح انتخاب شده‌اند. در ادامه مبحث روش قرار دادن نشانه نیز معروفی شده است.



شکل (۱۸-۶): فضای ویژگی‌ها با سه نشانه.

با توجه به رابطه زیر که برای محاسبه شباهت‌ها است به نحوه پیدا کردن ویژگی‌های مناسب براساس نشانه‌ها توجه نمایید.

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{i=1}^n (x_i - l_i^{(1)})^2}{2\sigma^2}\right) \quad \text{رابطه (۱۴)}$$

هدف اصلی از یافتن شباهت‌ها این است که می‌خواهیم هر داده آموزشی را با هر کدام از نشانه‌ها ارزیابی کنیم و اگر داده آموزشی از نظر شباهت به یک نشانه نزدیک بود در درون دسته مورد نظر قرار گیرد و در غیر این صورت درون دسته مورد نظر جای نگیرد.

## ۶-۹-۳- محاسبه شباهت<sup>۲</sup>

اگر  $x^1 \approx l^1$  برقرار باشد در این صورت رابطه  $f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$  را خواهیم داشت، یعنی اگر داده  $x^i$  به  $l^i$  از نظر شباهت نزدیک باشد  $f_1 \approx 1$  خواهد بود که عدد یک نشانگر شباهت است. به عبارتی عدد صفر موجود در صورت کسر گویای این است که فاصله اقلیدسی به صفر رسیده است و خروجی تابع تقریباً یک خواهد بود.

اگر  $x^1 \approx l^1$  برقرار نباشد در این صورت رابطه  $f_1 \approx \exp\left(-\frac{(\text{عدد بزرگ})^2}{2\sigma^2}\right) \approx 0$  را خواهیم داشت. یعنی اگر صورت کسر که همان فاصله اقلیدسی ویژگی با نشانه مورد نظر است عددی بزرگ باشد نشان‌دهنده فاصله زیاد شباهت این دو است و خروجی تابع به صفر نزدیک‌تر خواهد بود.

1. Land mark

2. Similarity computation

چنانچه در زیر مشاهده می‌نمایید برای هر نشانه  $L$  یک ویژگی  $f$  وجود دارد.

$$l^{(1)} \rightarrow f_1$$

$$l^{(2)} \rightarrow f_2$$

$$l^{(3)} \rightarrow f_3$$

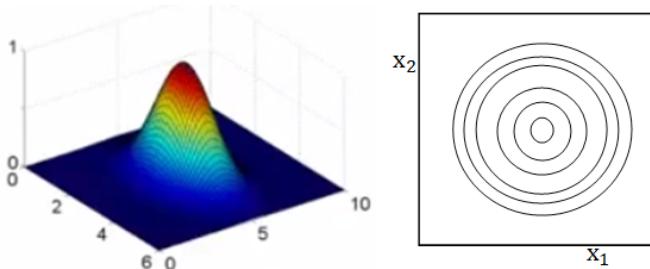
$$l_1 \rightarrow f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

$$l_2 \rightarrow f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$$

$$l_3 \rightarrow f_3 = \text{similarity}(x, l^{(3)}) = \exp\left(-\frac{\|x - l^{(3)}\|^2}{2\sigma^2}\right)$$

برای درک بهتر عملکرد این تابع به مثال‌هایی از نمودار کانتور با اعداد مختلف  $\sigma^2$  در زیر توجه نمایید.

مثال) فرض کنید  $l^{(1)} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$  باشد و  $\sigma^2 = 1$  باشد نمودار کانتور آن به صورت شکل (۱۹-۶) خواهد بود.



شکل (۱۹-۶): نمودار کانتور مربوط به تابع گوسی وقتی  $\sigma^2 = 1$  باشد.

به دلیل اینکه مقادیر  $X$  به صورت  $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$  می‌باشد پس نقطه مورد نظر برای  $l^{(1)}$  تقریباً در مرکز مکعب قرار دارد

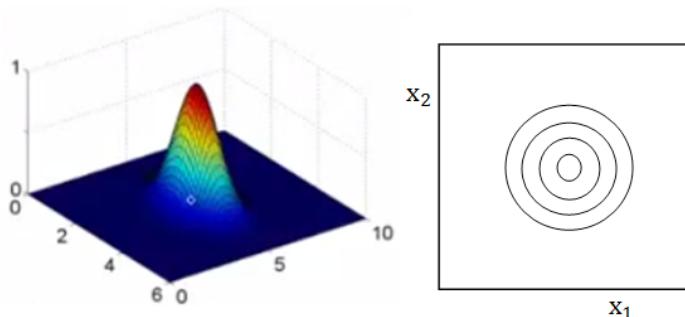
و به همین دلیل جواب  $f_1$  تقریباً به یک نزدیک است. ولی اگر مقادیر به گونه‌ای بود که نقطه تلاقی  $x_1$  و  $x_2$  از

مرکز دور بود مانند  $\begin{bmatrix} 6 \\ 8 \end{bmatrix}$  در این حالت  $f_1$  به صفر نزدیک‌تر می‌شد. توجه داشته باشید که مقدار اختصاص داده

شده به  $\sigma^2$  تعیین‌کننده پهنه‌ای ناحیه تپه مرکز مکعب می‌باشد و اگر آن را به ۰.۵ کاهش دهیم عرض تپه کم

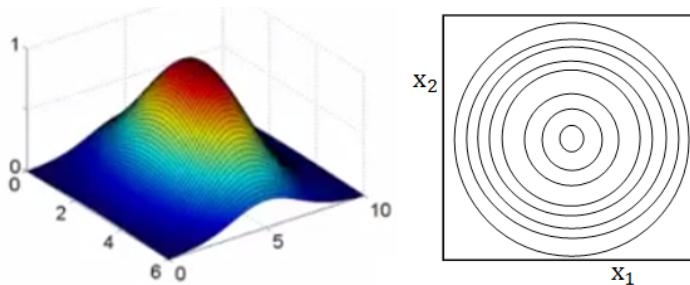
شده و حساسیت به قرار گرفتن نقطه تلاقی  $x_1$  و  $x_2$  در مرکز مکعب بالاتر می‌رود که آن را می‌توانید در شکل

(۲۰-۶) مشاهده نمایید.



شکل (۲۰-۶): نمودار کانتور مربوط به تابع گوسی وقتی  $\sigma^2 = 0.5$  می‌باشد.

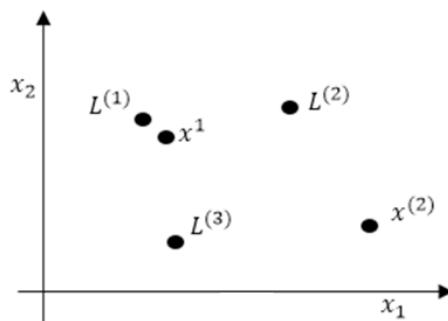
هر چه مقدار  $\sigma^2$  زیادتر شود مثلاً  $\sigma^2 = 3$  باشد عرض تپه‌ی مرکز مکعب بیشتر می‌شود و حساسیت شباهت کمتر می‌شود. به شکل (۲۱-۶) توجه کنید.



شکل (۲۱-۶): نمودار کانتور مربوط به تابع گوسی وقتی  $\sigma^2 = 3$  می‌باشد.

برای مشاهده نحوه عملکرد تابع گوسی برای ایجاد خط تصمیم غیرخطی و انجام عمل دسته‌بندی داده‌ها به صورت عملی به مثال زیر توجه نمایید.

مثال) در این مثال مقادیر  $\theta$ ‌ها در حالت فرضی به صورت زیر است و تعداد نشانه‌ها بر اساس شکل (۲۲-۶) خواهد بود که سه نشانه و دو نمونه آموزشی وجود دارد.



شکل (۲۲-۶): فضای ویژگی‌ها با نشانه‌های موجود.

بر اساس شکل (۲۲-۶) به دلیل نزدیک بودن  $\mathbf{l}^{(1)}$  به  $\mathbf{x}^1$  رابطه زیر برقرار است.

$$f_1 = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}^{(1)}\|^2}{2\sigma^2}\right) \approx 1 \rightarrow f_1 \approx 1, \quad f_2 \approx 0, \quad f_3 \approx 0$$

اگر  $f_1$  تقریباً برابر یک پس عملاً  $f_2$  و  $f_3$  باید تقریباً برابر صفر شوند. پس در ادامه مقادیر به دست آمده را در رابطه موجود جایگذاری می‌کنیم. اگر با استفاده ازتابع گوسی آهای زیر را داشته باشیم و همچنین مقادیر  $\theta$  نیز به صورت زیر باشد.

$$f_1 \approx 1, \quad f_2 \approx 0, \quad f_3 \approx 0$$

$$\theta_0 = -0.5, \quad \theta_1 = 1, \quad \theta_2 = 1, \quad \theta_3 = 0,$$

رابطه محاسبه  $h_{\theta}(x)$  به صورت زیر خواهد بود.

$$h_{\theta}(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3$$

$$h_{\theta}(x) = \theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0 = -0.5 + 1 = 0.5$$

$$h_{\theta}(x) \geq 0$$

به دلیل  $0 \geq h_{\theta}(x) \geq 0$  پس  $y=1$  می‌باشد. به عبارتی داده مربوطه در درون دسته مورد نظر خواهد بود، ولی اگر  $0 < h_{\theta}(x) < 0$  پس  $y=0$  بوده و داده مربوطه در درون دسته مورد نظر قرار نمی‌گرفت. برای این حالت به مثال زیر توجه نمایید.

مثال) برای یکی دیگر از  $x$ های موجود براساس شکل (۲۲-۶) فاصله نشانه‌ها را محاسبه می‌کنیم. بدليل اینکه فرض کردیم  $\mathbf{x}^2$  به هیچ  $L$  ای نزدیک نیست محاسبات زیر را خواهیم داشت.

$$f_1 \approx f_2 \approx f_3 \approx 0$$

$$\theta_0 = -0.5, \quad \theta_1, \theta_2 = 1, \quad \theta_3 = 0$$

با جایگذاری پارامترهای  $\theta$  و  $f$  در رابطه مربوطه نتیجه زیر را خواهیم داشت.

$$h_{\theta}(x) = -0.5$$

$$h_{\theta}(x) < 0$$

بنابراین  $0^2$  به هیچ  $L$  ای نزدیک نیست و  $y=0$  خواهد شد و داده مربوطه در درون دسته مورد نظر قرار نخواهد گرفت.

#### ۴-۹-۴ - نحوه تعیین نشانه

درمبخت توابع هسته ما برای توضیح بحث به حالت ساده‌تر  $L$ ها را به صورت دستی و فرضی در نمودار قرار دادیم ولی در مسائل واقعی طریقه انتخاب و تنظیم این نشانه‌ها براساس قوانین مشخصی انجام می‌شود.

یکی از روش‌هایی که برای پیدا کردن  $\textcolor{blue}{l}^{(i)}$  وجود دارد انتخاب خود مثال‌های آموزشی به عنوان  $\textcolor{blue}{l}$  بدون توجه به ماهیت آن می‌باشد،  $\textcolor{blue}{l}$ ها می‌تواند هر تعدادی باشد. توجه داشته باشید که انتخاب تعداد نقاط زیاد به عنوان  $\textcolor{blue}{l}$  باعث بالاتر رفتن پیچیدگی برنامه خواهد شد.

مثال) اگر نمونه‌های آموزشی داده شده به صورت زیر مفروض باشد.

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$$

محل دقیق نمونه آموزشی را به عنوان نشانه به صورت دستی انتخاب می‌کنیم.

$$\textcolor{blue}{l}^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$$

و برای هر نمونه آموزشی  $(x^{(i)}, y^{(i)})$  محاسبات زیر را انجام می‌دهیم.

$$f_1^{(i)} = \text{similarity}(x^{(i)}, l^{(1)})$$

$$f_2^{(i)} = \text{similarity}(x^{(i)}, l^{(2)})$$

...

$$f_m^{(i)} = \text{similarity}(x^{(i)}, l^{(m)})$$

و سپس تمامی  $f$ ها در درون یک بردار می‌ریزیم و به دلیل رابطه  $f_0 = 1$  به ابتدای بردار افزوده می‌شود.

$$F^{(i)} = \begin{bmatrix} F_0^{(i)} \\ F_1^{(i)} \\ F_2^{(i)} \\ \vdots \\ F_m^{(i)} \end{bmatrix}$$

در نهایت بردار  $F^{(i)}$  یک بردار شامل ویژگی‌های جدید است که نمونه‌های آموزشی را ارائه می‌کند.

#### ۶-۱۰- رابطه ماشین بردار پشتیبان براساس تابع هسته

برای محاسبه و دسته‌بندی با استفاده از هسته از رابطه (۱۵) استفاده می‌شود.

$$\min_{\theta} c \sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2 \quad \text{رابطه (۱۵)}$$

همان‌طور که مشاهده می‌شود در رابطه ماشین بردار پشتیبان به جای  $\theta^T x^{(i)}$   $\theta^T f^{(i)}$  استفاده کرد.

به دلیل مساوی بودن نمونه‌های آموزشی با ویژگی‌ها در قسمت رابطه بهینه‌سازی  $n=m$  در نظر گرفته می‌شود و

همچنین به دلیل شروع  $\theta$  از نقطه  $\theta_0$  برای  $i$  موجود در رابطه  $\frac{1}{2} \sum_{i=1}^n \theta_j^2$  این رابطه به صورت  $\frac{1}{2} \sum_{i=0}^n \theta_j^2$  می‌باشد.

تغییر می‌یابد و رابطه نهایی برای محاسبه ماشین بردار پشتیبان به صورت زیر است.

$$\min_{\theta} \sum_{i=1}^m y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2 \quad (16)$$

#### ۶-۱-۱- اهمیت مقدار دهی مناسب یارا مترها

اگر C بزرگ باشد دقت برنامه در مرحله آموزش بالا خواهد بود و انعطاف برنامه برای تحمل خطا پایین می‌آید و اگر C کوچک باشد دقت پایین و انعطاف برنامه بالا خواهد بود.

اگر  $\sigma^2$  مقدار بزرگ باشد قطر تپه موجود در مرکز مکعب بیشتر و انعطاف الگوریتم بالا و دقت الگوریتم پایین خواهد بود. اگر  $\sigma^2$  مقدارش کوچک باشد انعطاف الگوریتم پایین و دقت الگوریتم بالا می‌رود و قطر تپه درون مکعب ناکنتر می‌باشد.

در شرایطی که تعداد ویژگی‌های آموزشی کم و نمونه‌های آموزشی بسیار بزرگ باشد، در این مورد هم، می‌توان از ماشین بردار پشتیبان بدون کرنل استفاده کرد و استفاده از کرنل گوسی نتایج مناسبی را نسبت به روش بدون کرنل ارائه ندهد ولی برای حل مسائلی که در آن تعداد ویژگی‌های آموزشی کم و نمونه‌های آموزشی در حد متوسط می‌باشند نتیجه الگوریتم با کرنل گوسی، بهتر خواهد بود.

#### ۶-۱۱- مثالی برای دسته‌بندی به روش ماشین یردار پشتیبان با زبان R

در این برنامه هدف ایجاد یک دسته‌بندی کننده توسط ماشین بردار پشتیبان است که از مجموعه داده iris برای دسته‌بندی، استفاده شده است.

برای پیاده‌سازی ماشین بردار پشتیبان می‌توان از یک بسته نرم‌افزاری معروف به نام Iris با استفاده کرد که در خط اول برنامه بارگذاری شده است. همچنین چنانچه در خط دوم مشاهده می‌نمایید مجموعه داده Iris با استفاده از بسته برگذار شده است.

```
> library("e1071")
```

```
> library(datasets)
```

```
> head(iris,5)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa

5	5.0	3.6	1.4	0.2	setosa
---	-----	-----	-----	-----	--------

با استفاده از سه خط برنامه زیر نام گونه‌ها که همان species می‌باشد را به متغیر `y` و بقیه ویژگی‌ها با داده‌های آنها را به متغیر `x` اختصاص می‌دهیم. تابع `attach()` باعث می‌شود تا اشیاء درون یک data frame قابل جستجو و دسترسی باشد و تابع `subset()` می‌تواند بخشی از مجموعه داده را جدا نموده و در متغیر ذخیره کند.

```
> attach(iris)
> x <- subset(iris, select=-Species)
> y <- Species
```

خطوط زیر مدل الگوریتم ماشین بردار پشتیبان مورد نظر را ایجاد نموده و مشخصات مدل ایجاد شده را نمایش می‌دهد. براساس اطلاعات موجود در بخش parameters می‌توان مدل ایجاد شده را تفسیر نمود. در حرف `C` همان ثابت ظرفیت (capacity constant) و `radial` در بخش svm-kernel مشخص‌کننده نوع کرنل به کار رفته است، برای پارامتر `cost` هرچه مقدار کوچک‌تری نمایش داده شود نشان‌دهنده عملکرد بهتر برای بردارهای پشتیبان بوده و در نتیجه، دسته‌بندی دقیق‌تر انجام خواهد گرفت. پارامتر `gamma` تأثیرش بر نتیجه نهایی بسیار بالا است، اگر `gamma` عددی بیش از حد بالا باشد، تأثیر پارامتر `C` در جلوگیری از بیش برازش یا خیلی کم و یا کاملاً از بین می‌رود و تنها در بهینه‌سازی الگوریتم، خود بردار پشتیبان نقش ایفاء خواهد نمود.

```
> svm_model <- svm(Species ~ ., data=iris)
> summary(svm_model)
```

Call:

```
svm(formula = Species ~ ., data = iris)
```

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 1

gamma: 0.25

Number of Support Vectors: 51

( 8 22 21 )

Number of Classes: 3

Levels:

setosa versicolor virginica

همچنین می‌توان از کد زیر نیز برای ایجاد مدل استفاده کرد.

```
> svm_model1 <- svm(x,y)
> summary(svm_model1)
```

Call:

```
svm.default(x = x, y = y)
```

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 1

gamma: 0.25

Number of Support Vectors: 51

( 8 22 21 )

Number of Classes: 3

Levels:

setosa versicolor virginica

برای اجرای الگوریتم برای عمل دسته بندی از تابع predict() در برنامه به صورت زیر استفاده می‌نماییم.

```
> pred <- predict(svm_modell,x)
```

می‌توان از خط زیر برای محاسبه زمان اجرای الگوریتم استفاده نمود.

```
> system.time(pred <- predict(svm_modell,x))
```

user system elapsed

0 0 0

در نهایت با اجرای دستور زیر می‌توان یک ماتریس درهم ریختگی<sup>1</sup> که نشان‌دهنده نتایج دسته‌بندی است را ایجاد

کرد که ماتریس درهم ریختگی در فصل مربوط به ارزیابی الگوریتم‌های یادگیری ماشین توضیح داده شده است.

در کل برای دسته setosa به تعداد ۵۰ نمونه، برای versicolor به تعداد ۴۸ نمونه و برای virginica به تعداد ۴۸

نمونه به درستی دسته‌بندی شده است.

```
> table(pred,y)
```

	y		
pred	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	2	48

## ۶-۱۲- مثالی برای دسته‌بندی به روش ماشین بردار پشتیبان با زبان پایتون

برنامه زیر یک ماشین بردار پشتیبان را با استفاده از زبان پایتون پیاده‌سازی می‌نماید که در آن از مجموعه داده Iris استفاده شده است.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
```

در سه خط زیر مجموعه داده iris به برنامه افزوده می‌شود که تنها دو ویژگی اول موجود در مجموعه داده در اینجا مدنظر قرار دارد.

```
iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target
```

خط زیر با استفاده از svm.SVC() دسته بند را برای دریافت داده و دسته‌بندی آنها ایجاد می‌نماید تا مدل آموزش دیده را ایجاد نماید.

```
h = .02
clf = svm.SVC()
clf.fit(X, Y)
```

در زیر تخمین نهایی دسته داده‌های آزمایشی با استفاده از clf.predict() انجام می‌شود و نمودار خط تصمیم ترسیم می‌شود. برای این منظور برای هر نقطه داده موجود در نمودار یک رنگ مشخص کننده دسته موردنظر اختصاص داده می‌شود.

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

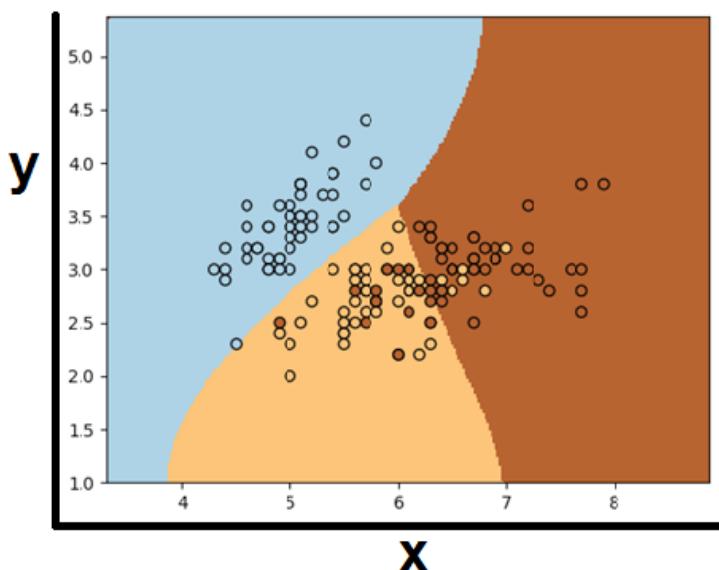
خطوط زیر نقاط داده‌ای را به صورت رنگی در می‌آورند.

```
Z = Z.reshape(xx.shape)
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
```

خطوط زیر نمودار نهایی را در خروجی چاپ می‌نمایند که خط اول کار ترسیم بلوک‌های رنگی در نمودار را به عهده دارد.

```
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired, edgecolors='k')
plt.title('3-Class classification using Support Vector Machine')
plt.axis('tight')
plt.show()
```

نتیجه دسته‌بندی در نمودار زیر قابل مشاهده می‌باشد.





فصل

## یادگیری بدون نظارت

## ۷-۱- یادگیری بدون نظارت

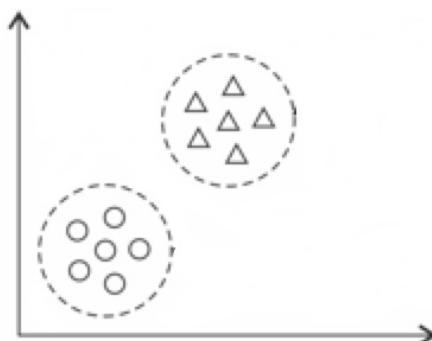
یادگیری بدون نظارت یک روش در یادگیری ماشین است که با انجام عمل استنتاج بر روی داده‌های بدون برچسب، ساختار مخفی داده‌ها و روابط آنها را کشف می‌کند. در یکی از روش‌های این نوع یادگیری به نام خوشه‌بندی؛ در طی یک فرآیند خودکار، داده‌ها به دسته‌هایی که اعضای آنها مشابه به یکدیگر می‌باشند تقسیم می‌گردند، که به این دسته‌ها خوشه<sup>۱</sup> گفته می‌شود. بنابراین خوشه مجموعه‌ای از داده‌ها می‌باشند که در آن داده‌ها با یکدیگر مشابه بوده و با داده‌های موجود در خوشه‌های دیگر غیرمشابه می‌باشند.

## ۷-۲- روش‌های خوشه‌بندی

روش‌های خوشه‌بندی به دو رده کلی تقسیم می‌شوند: ۱- خوشه‌بندی سخت. ۲- خوشه‌بندی نرم.<sup>۲</sup>

### ۷-۲-۱- خوشه‌بندی سخت

برای این الگوریتم اگر طبق شکل (۱-۷) خوشه‌هایی را داشته باشیم که به صورت کامل از هم جدا باشند و هیچ بخش مشترکی بین آنها نباشد خوشه‌بندی سخت گویند.

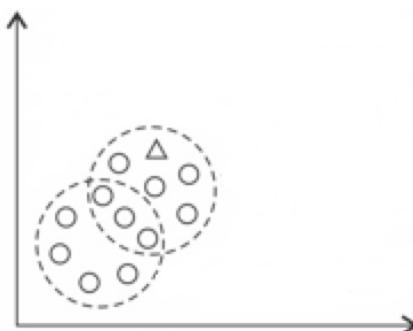


شکل (۱-۷): خوشه‌بندی سخت.

### ۷-۲-۲- خوشه‌بندی نرم

در این روش برخلاف روش خوشه‌بندی سخت چنانکه در شکل (۲-۷) مشاهده می‌نمایید حالتی وجود دارد که همیشه نمی‌توان مرز دقیقی برای خوشه‌ها در نظر گرفت و در آن معمولاً داده‌هایی وجود دارند که با درجه شباهت‌های متفاوت به هر چند خوشه تعلق دارند.

- 
1. Cluster
  2. Hard clustering
  3. Soft clustering



شکل (۲-۷): خوشبندی نرم.

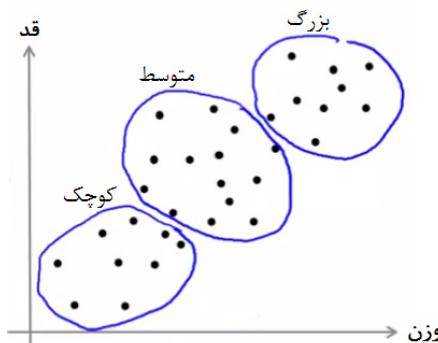
### ۷-۳- انواع مختلف الگوریتم خوشبندی

الگوریتم‌های مختلف مانند مدل‌های مبتنی بر نقطه مرکزی<sup>۱</sup>، مدل‌های مبتنی بر چگالی<sup>۲</sup>، مدل‌های مبتنی بر اتصالات<sup>۳</sup> برای عمل خوشبندی وجود دارد که همه آنها با مجموعه قوانین متفاوتی شباهت داده‌ها را تعریف می‌کنند. ما در این کتاب الگوریتم k-means که یکی از الگوریتم‌های مبتنی بر نقطه مرکزی است را به شما معرفی می‌کنیم تا بتوانید با مراحل یکی از مهمترین الگوریتم‌های خوشبندی آشنا شوید.

### ۷-۴- نحوه انتخاب تعداد خوشبندی

فرض کنید می‌خواهیم برای گروه‌بندی و تفکیک پیراهن‌ها در اندازه‌های مختلف راهکاری را ارائه کنیم. برای این کار می‌توان از روش‌های خوشبندی استفاده کرد که از روی ویژگی‌های پیراهن‌ها آنها را در خوشبندی مربوط به پیراهن‌هایی با اندازه کوچک، متوسط و بزرگ قرار دهد شکل (۳-۷) این خوشبندی را نشان می‌دهد. همچنین می‌توان داده‌های مربوطه براساس نیاز مسئله به خوشبندی بیشتری مانند خیلی کوچک، کوچک، متوسط، بزرگ و خیلی بزرگ را نیز خوشبندی کرد.

- 
1. Centroid models
  2. Distribution models
  3. Density models
  4. Connectivity models



شکل (۷-۳): خوشبندی لباس از روی اندازه آنها.

در اصل اگر بخواهیم پیراهن‌ها را در دسته‌های متفاوتی قرار دهیم باید این کار را براساس ویژگی‌های پیراهن‌ها انجام دهیم. همان‌گونه که در شکل (۷-۳) مشاهده می‌نمایید دو ویژگی قد و وزن می‌توانند انتخاب‌های مناسبی باشند. معمولاً انتخاب ویژگی براساس نیاز مسئله و رویکرد اصلی ما انجام می‌شود که براساس آن ویژگی‌ها، می‌توانیم داده‌های موجود را به سه خوش، پنج خوش و یا حتی بیشتر تقسیم کنیم.

## ۷-۵- ساختار کلی الگوریتم k-means

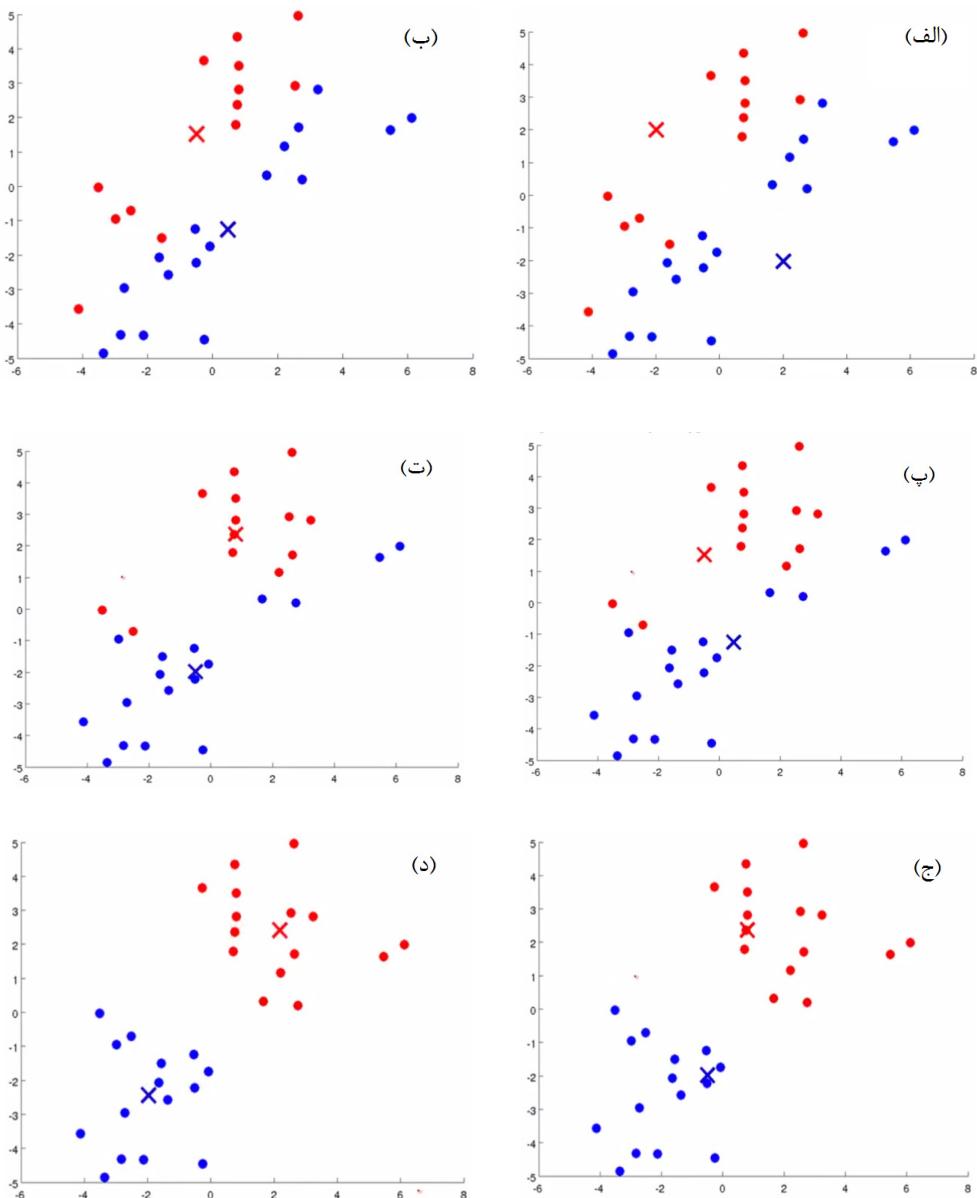
از این الگوریتم برای خوشبندی داده‌های بدون برچسب استفاده می‌شود که الگوریتم و یا نحوه عملکرد آن به صورت زیر می‌باشد:

۱. ابتدا چند نقطه مرکزی<sup>۱</sup> به عنوان مرکز خوش‌ها انتخاب می‌شود.
۲. فاصله هر کدام از نمونه‌های آموزشی با نقاط مرکزی محاسبه می‌شود که این نقاط مرکزی همان مراکز خوش‌ها می‌باشند.
- ۳- نزدیک‌ترین نقطه مرکزی برای هر نمونه آموزشی پیدا می‌شود.
- ۴- هر نمونه آموزشی به نزدیک‌ترین خوش‌های اختصاص داده می‌شود.
۵. نقطه مرکزی خوش‌ها دوباره به مرکزی‌ترین قسمت داده‌های نزدیک خود می‌رود.
۶. مرحله دو و سه تازمانی که خوش‌ها به حالت بهینه خود نرسیده‌اند، ادامه می‌پابد.

به عبارتی ابتدا نقطه‌ای به عنوان نقطه مرکزی در میان نمونه‌های آموزشی مشخص می‌شود که قرار است این نقطه‌ها با جمع آوری داده‌های مناسب در کنار خود، یک خوش از داده‌هایی با شباهت بیشتر به هم را پیدا آورند که خود، نقطه مرکزی این خوش‌ها خواهد بود و در ادامه فاصله هر کدام از نمونه‌های آموزشی با تک تک نقاط مرکزی محاسبه شده و هر کدام از این نمونه‌ها به نزدیک‌ترین خوش از نظر شباهت اختصاص می‌یابد و بعد از آن دوباره نقاط مرکزی به مرکز خوش نزدیک‌تر می‌شوند و این فرایند تکراری که نوبتی بین نمونه‌های آموزشی و

1. Centriod

نقاط مرکزی اتفاق می‌افتد، تا زمانی ادامه دارد که داده‌ها در بهترین وضعیت ممکن خوشبندی شوند. این فرایند در شکل (۴-۷) از (الف) تا (د) به ترتیب نمایش داده شده است.



شکل (۴-۷): فرایند خوشبندی در الگوریتم k-means

براساس شکل (۴-۷) فرض بر این است که می‌خواهیم دو خوش داشته باشیم و بر این اساس در شکل (۴-۷)(الف)

دو نقطه مرکزی با علامت ضریر انتخاب شده که یکی قرمز و دیگری آبی است و مابین داده‌ها قرار گرفته‌اند. در شکل (۴-۷)(ب) نقاط مرکزی کمی به سمت داده‌های هم رنگ خود تغییر مکان داده‌اند. در شکل (۴-۷)(پ) پس از محاسبه داده‌ها، تعدادی از داده‌هایی که با رنگ قرمز مشخص شده‌اند به سمت مرکز خوش قرمز نزدیک‌تر شده و همچنین داده‌های مشخص شده با رنگ آبی به مرکز خوش آبی نزدیک‌تر شده‌اند و داده‌های مجاور از نظر رنگ که منظور همان نوع یکسان است به یکنواختی بیشتری رسیدند ولی هنوز داده‌های کنار هم کاملاً شبیه به هم نیستند. در شکل (۴-۷)(ت) نقاط مرکزی پس از محاسبه فاصله‌ها جابجا شده و به مرکز خوش همنگ خود کمی نزدیک‌تر شده‌اند. در شکل (۴-۷)(ج) تمامی داده‌های قرمز و آبی کاملاً از هم تفکیک شده‌اند و به نقطه مرکزی همنگ خود نزدیک‌تر شده‌اند، ولی هنوز نقطه مرکزی در مرکزی‌ترین قسمت خوش خود قرار ندارد. در شکل (۴-۷)(د) مراکز خوش‌ها یکبار دیگر تغییر مکان داده و در مرکز خوش‌ها جای گرفته است.

بعد از این مرحله به دلیل اینکه داده‌ها کاملاً از هم جدا شده و نقاط مرکزی نیز در مرکزی‌ترین قسمت ممکن در خوش‌ها جای گرفته‌اند دیگر تغییرات قابل ملاحظه‌ای نخواهیم داشت و برنامه به اتمام رسیده است و تکرار بیش از این در نتیجه خوش‌بندی اثر چندانی نخواهد داشت. پس در نتیجه خوش‌بندی به بهترین وضعیت ممکن رسیده است.

## ۷-۵-۱- الگوریتم k-means و روابط محاسباتی آن

همان‌گونه که در مباحث فصل‌های قبل ملاحظه نمودید برای رساندن الگوریتم به حالت بهینه که بتواند بهترین نتیجه را برای ما تولید کند، کار بر روی پیاده‌سازی الگوریتم با شرایطی که در آن تابع هزینه به کمینه ممکن برسد مسئله اصلی ما خواهد بود.

برای انجام خوش‌بندی با درصد خطای پایین برای k-means دو موضوع اصلی را باید در نظر داشته باشیم. اولی پیاده‌سازی صحیح الگوریتم و برطرف‌سازی خطاهای برنامه است که باعث شود تا برنامه به صورت صحیح اجرا شده و خوش‌بندی را انجام دهد. دومی بهینه‌سازی آن است که باعث شود تا الگوریتم از افتادن به کمینه‌های محلی اجتناب کند و خوش‌بندی‌های بهتر و دقیق‌تری را داشته باشیم.  
۱. مقداردهی اولیه نقاط مرکزی به صورت تصادفی.

$$\mu_1, \mu_2, \dots, \mu \in R^n$$

۲. تکرار تا زمانی که به حالت بهینه برسد.

}

۱-۲- محاسبه محل بهینه برای نقاط مرکزی از نظر نزدیک‌ترین فاصله به میانگین داده‌های خوش مربوطه

$$\text{برای هر } i=1 \text{ تا } m \text{ محاسبه کن } C^i := \min_k x^i - \mu_k^2 \text{ را :}$$

۲-۲- محاسبه داده‌ها جهت اختصاص آنها به خوش مناسب.

$$\text{برای هر } k=1 \text{ تا } K \text{ محاسبه کن } \mu_k \text{ را :} \\ \left\{ \begin{array}{l} \mu_k := \frac{\sum_{k=1}^K \mathbf{1}_{\{c^i=k\}} \mathbf{x}^i}{\sum_{k=1}^K \mathbf{1}_{\{c^i=k\}}} \end{array} \right.$$

### ۷-۵-۲ - توضیحات الگوریتم k-means

توجه شود که در الگوریتم بالا  $K$  به تعداد کل خوشها اشاره دارد و  $\{1, 2, \dots, K\}$  نشان‌دهنده شماره خوشهاست کوچک می‌باشد.

رابطه  $\mathbf{x}^i - \mu_k^2$  نشان‌دهنده فاصله بین نمونه‌های آموزشی  $\mathbf{x}^i$  و مرکز خوشهاست  $\mu_k$  می‌باشد.

$\mathbf{c}^i$  نشان‌دهنده یک اندیس از نقاط مرکزی خوش که نزدیک‌ترین فاصله را به  $\mathbf{x}^i$  دارد.

$\mu_k$  میانگینی از نقطه‌های اختصاص داده شده به خوش  $k$  می‌باشد. به عنوان مثال برای نقطه مرکزی  $\mu_2$  اگر نمونه‌های آموزشی  $\mathbf{x}^{10}, \mathbf{x}^6, \mathbf{x}^5, \mathbf{x}^1$  را داشته باشیم تساوی زیر را خواهیم داشت.

$$C^i \Rightarrow C^1 = 2, C^5 = 2, C^6 = 2, C^{10} = 2$$

زیرا به دلیل وجود عدد ۲ به عنوان اندیس در  $\mu_2$  که در اصل همان  $\mu_2 = \mu_{C^i}$  می‌باشد هر کدام از  $C^i$ ‌ها مساوی ۲ خواهد بود.

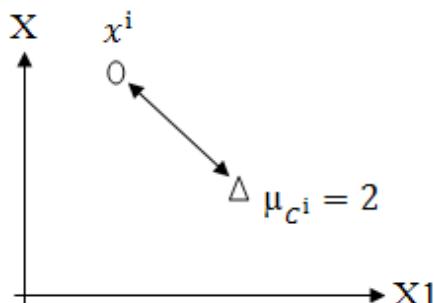
و برای به دست آوردن  $\mu_{C^i} = \mu_2$  رابطه زیر را خواهیم داشت.

$$\mu_{C^i} = \mu_2 = \frac{1}{4} [x^1 + x^5 + x^6 + x^{10}] \in \mathbb{R}$$

برای به حداقل رساندن تابع هزینه  $J(C^i, \mu_k)$  از رابطه زیر استفاده می‌شود.

$$J(C^i, \mu_k) = \frac{1}{m} [\mathbf{x}^i - \mu_{C^i}]^2$$

برای به دست آوردن فاصله بین  $\mathbf{x}^i$  و  $\mu_k$  از رابطه  $\mathbf{x}^i - \mu_k^2$  استفاده می‌کنیم. برای درک بهتر به شکل (۷-۶) توجه نمایید.



شکل (۷-۶): میزان فاصله شباهت دو نمونه.

در مرحله اول از بخش دوم الگوریتم ابتدا با ثابت نگه داشتن  $\mu_k$  تابع هزینه  $J(C^i, \mu_k)$  را بر اساس  $C^i$  به کمینه برسانیم و بعد از

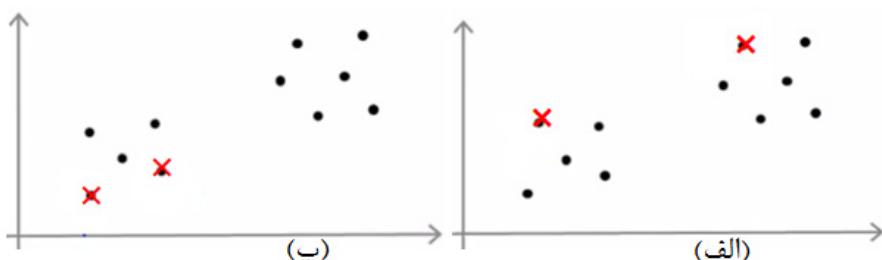
آن در مرحله دوم بخش دوم الگوریتم، هدف انتخاب مقدار صحیح برای  $\mu_k$  است که دوباره از این نقطه نظر تابع هزینه را به کمینه می‌رساند و این دو مرحله با تکرار نوبتی باعث به کمینه رسیدن کامل تابع هزینه  $J(C^k)$  می‌شود.

### ۷-۵-۳- مقداردهی اولیه الگوریتم k-means

در این الگوریتم مقداردهی اولیه نقاط مرکزی موضوع مهمی می‌باشد و مهم‌تر از آن به وجود آوردن شرایطی است که الگوریتم از گیر کردن در کمینه محلی اجتناب کند.

یکی از رویکردها برای انتخاب نقطه مرکزی انتخاب تصادفی است که برای درک چگونگی آن به مثال زیر توجه نمایید.

مثال) فرض کنیم مسئله ما به گونه‌ای می‌باشد که به دو خوش احتیاج است به عبارتی  $\mu_k \in R$  و  $k=2$  را داشته باشیم که نمودار مربوطه نیز به صورت شکل (۶-۷) می‌باشد.



شکل (۶-۷): دو نمودار برای نشان دادن وضعیت قرارگیری نقطه مرکزی.

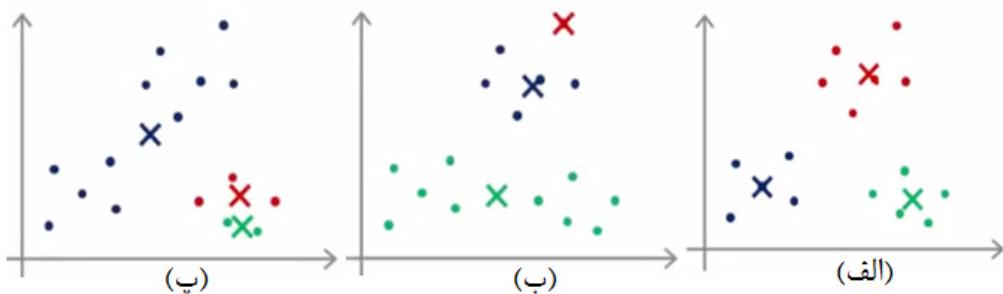
در مقداردهی تصادفی نقاط مرکزی می‌توانند در هر جایی از نمودار باشند و همان‌طور که در نمودار شکل (۶-۷) (الف) می‌بینید این دو نقطه به صورت اتفاقی در محل مناسب با فاصله مناسب از هم قرار دارند، ولی همان‌طور در شکل (۶-۷)(ب) مشاهده می‌کنید نقاط مرکزی در فاصله خیلی نزدیک به هم و در محل تجمع داده‌ها قرار گرفته‌اند که می‌تواند باعث گیر کردن الگوریتم در کمینه محلی شود و شکل‌گیری خوش‌های جدا از هم و مناسب امکان‌پذیر نباشد. برای درک بهتر این شرایط به یک مثال دیگر توجه نمایید.

مثال) فرض کنید می‌خواهیم سه خوش تشكیل دهیم که داده‌های آن به صورت زیر پراکنده شده‌اند.



شکل (۷-۷): داده‌های پراکنده در فضای برداری.

حال با مقدار دهی تصادفی مراکز خوش ممکن است حالات زیر اتفاق افتد.



شکل (۸-۷): مقادیر مرکزی مختلف به عنوان نقاط مرکزی خوش.

در شکل (۸-۷)(الف) مقادیر مرکزی در محل‌های بسیار مناسب بوده‌اند و به راحتی سه خوش را تشکیل خواهند داد. در شکل (۸-۷)(ب) نقطه مرکزی که با رنگ آبی مشخص شده در محل مناسبی قرار دارد، نقطه سیز به اشتباہ بین دو خوش قرار گرفته و نقطه قرمز تنها یک داده را به خود اختصاص داده و در بدترین جای ممکن قرار گرفته و هیچ کارایی ندارد. در شکل (۸-۷)(پ) دو نقطه مرکزی که با رنگ‌های قرمز و سبز نشان داده شده است تلاش به تفکیک داده‌هایی می‌کند که در حالت بهینه می‌باشد در یک خوش و واحد قرار می‌گرفتند و نقطه مرکزی دیگر که به رنگ آبی است در بین دو تجمع از داده‌هایی قرار گرفته که شباهت چندانی به هم ندارند و باید به دو خوش مجزا تفکیک شوند که این وضعیت نشان‌دهنده این است که الگوریتم در کمینه محلی گیر کرده است.

## ۷-۶- روش اجتناب از مقادیرهای اولیه نامناسب در خوشبندی

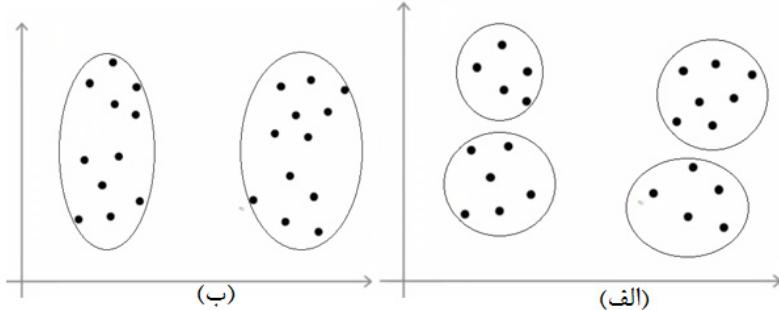
یکی از راه‌های اجتناب از گیر کردن در کمینه محلی این است که نقاط مرکزی را به دفعات مقادیرهای کنیم تا اینکه از قرارگیری آنها در محل مناسب اطمینان حاصل کنیم. برای این امر حلقه‌ای تشکیل می‌دهیم که مقادیرهای اولیه را انجام دهد، این حلقه مقادیر مختلفی را به نقاط مرکزی به صورت تصادفی اختصاص می‌دهد و چندین بار این مقادیرهای اولیه را تکرار می‌کند و پس از آنکه حلقه کار خود را به اتمام رساند تمامی حالات به دست آمده را با هم مقایسه کرده و بهترین حالت اتفاق افتاده را به عنوان نقاط مرکزی انتخاب می‌کند. بهترین حالت همانی می‌باشد که تابع هزینه آن پایین‌ترین مقدار را دارد.

فرض کنید برای حلقه خود  $i=1, \dots, 100$  را داشته باشیم، در هر اجرای حلقه  $\mu_k, \dots, \mu_1$  و  $C^i, C^m$  محاسبه می‌شود و وقتی حلقه به پایان می‌رسد حالتی که در آن برای  $(\mu_k, \dots, \mu_1)$  پایین‌ترین مقدار به دست آمد یک حالت بهینه برای الگوریتم است و بهترین حالت قرارگیری نقاط مرکزی را خواهیم داشت.

## ۷-۷- انتخاب تعداد خوش‌های بدون ناظارت

برای انتخاب تعداد خوش‌های می‌توان به صورت انتخاب دستی و یا انتخاب خودکار عمل کرد. در روش دستی با استفاده از تکنیک‌های مختلفی مانند مصورسازی و نگاه کردن به خروجی الگوریتم خوشبندی و مواردی دیگر می‌توان

تعداد خوشه هارا انتخاب کرد که در مسائل مختلف مورد استفاده قرار می‌گیرد. مثلاً در شکل (۹-۷) هر دو نمودار نشان‌دهنده یک وضعیت قرارگیری مانند هم برای داده‌ها است که به صورت دستی توسط کاربری به دو خوشه مانند شکل (۹-۷)(ب) و توسط کاربری دیگر به چهار خوشه مانند شکل (۹-۷)(الف) تقسیم شده است.



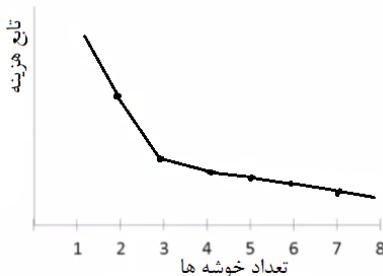
شکل (۹-۷): تقسیم داده‌ها به تعداد چهار خوشه در شکل (الف) و دو خوشه در شکل (ب).

### ۷-۷-۱- انتخاب تعداد خوشه به صورت خودکار

روشی دیگر وجود دارد که در این روش خوشه‌بندی به صورت خودکار انجام می‌شود. اگر بخواهیم روش دستی را با روش خودکار مقایسه کنیم نمی‌توان گفت که به صورت کلی کدام یک بهتر است ولی بسته به شرایط مسئله هر کدام از روش‌ها می‌تواند مفید باشد.

### ۷-۷-۱-۱- روش آرنج<sup>۱</sup> در انتخاب خودکار تعداد خوشه

یکی از روش‌های خوشه‌بندی خودکار می‌باشد که بر آن اساس می‌توان تعداد مناسب خوشه‌ها را تشخیص داد و خوشه‌ها را از هم جدا کرد. برای درک بهتر روش آرنج به مثال زیر توجه نمایید.



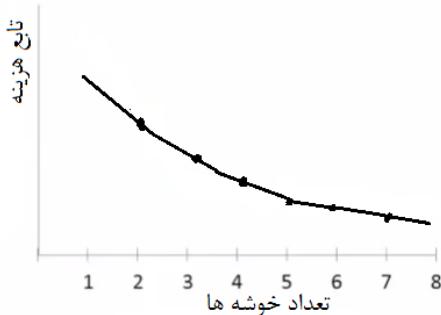
شکل (۷-۷): نمودار الگوریتم خودکار برای یافتن تعداد خوشه که در آن محل آرنج مشخص است.

همان‌گونه که در نمودار شکل (۷-۱۰) مشاهده می‌کنید خط به وجود آمده مانند بازوی انسان است که در نقطه ۳ مانند آرنج خم شده، معمولاً نقطه‌ای که مانند آرنج خم شده می‌تواند در هر نقطه‌ای مانند نقطه ۳، نقطه ۴ یا هر نقطه‌ای دیگر اتفاق افتد ولی تنها یک بار اتفاق می‌افتد.

1. Elbow method

در مثال ما نقطه ۳ جایی بوده که تابع هزینه ( $J$ ) با شیب زیادی کاهش یافته و بعد از آن کاهش چشم‌گیری نداشته است و به همین خاطر شکل آرنج بوجود آمده است. نقطه‌ای که در آن آرنج اتفاق می‌افتد مشخص‌کننده تعداد خوشه مناسبی است که برای مسئله باید در نظر گرفته شود.

حالات مختلفی در این روش ممکن است به دست آید و خط ترسیم شده در نمودار شرایطی مختلفی را نشان دهد که یکی از آنها وضعیتی است که در آن نتوان از روی خط تشخیص داد که کدام نقطه بیشتر شبیه به آرنج است و محل آرنج همان‌گونه که در شکل (۱۱-۷) مشاهده می‌شود خیلی واضح نباشد.



شکل (۱۱-۷): نمودار الگوریتم خودکار برای یافتن تعداد خوشه که در آن محل آرنج مشخص نیست.

در این حالت ممکن است نقطه ۳ یا ۴ و یا ۵ به نظر مناسب آید ولی نتایج خوشه‌بندی مناسبی ارائه ندهد به همین دلیل از روش آرنج مانند روش‌های دیگر نمی‌توان در تمامی حالات استفاده کرد.

## ۷-۷-۲ - انتخاب تعداد خوشه به صورت دستی

در بیشتر مسائل دنیای واقعی نیز تعداد خوشه‌ها بر حسب مسئله مورد نظر به صورت دستی انتخاب می‌شود و تعداد آن از پیش تعیین شده است مانند زمانی که می‌خواستیم تعداد خوشه‌ها را برای تقسیم‌بندی پیراهن‌ها انتخاب کنیم در این کار براساس نیاز مسئله که می‌خواهد پیراهن‌ها را در سه دسته کوچک، متوسط و بزرگ از هم جدا کند، باید برای الگوریتم خود سه خوشه در نظر بگیریم و نمی‌توانیم براساس روش آرنج به صورت خودکار تعداد خوشه‌ها را انتخاب کنیم زیرا ممکن است تعداد خوشه‌های انتخاب شده بیشتر یا کمتر از تعداد خوشه‌هایی باشد که ما نیاز داریم. همان‌گونه که مشاهده نمودید روش آرنج به صورت خودکار تعداد خوشه‌ها را مشخص می‌کند و ممکن است تعداد خوشه‌ها متناسب با نیاز مسئله نباشد، اما اگر تعداد خوشه‌ها از پیش مشخص نباشد می‌تواند راه حل مناسبی برای یافتن تعداد خوشه‌های موجود در مسئله باشد.

## ۷-۱-۱ - جرای الگوریتم k-means با استفاده از زبان R

در این الگوریتم بسته نرم‌افزاری مورد نیاز datasets می‌باشد که یک مجموعه داده با نام attitude را نیز به همراه

سایر مجموعه داده‌ها در خود جای داده است. مجموعه داده attitude مربوط به یک نظرسنجی از کارکنان دفتر یک سازمان بزرگ مالی است که داده‌های آن از پرسشنامه‌های حدود ۳۵ نفر از کارکنان تصادفی مربوط به حدود ۳۰ بخش مختلف سازمان جمع‌آوری گردیده است. اعداد موجود در مجموعه داده مربوط به امتیاز داده شده توسط پرکنندگان پرسشنامه است که می‌بایستی از ۱ تا ۱۰۰ عددی را به هر سؤال اختصاص می‌داده‌اند. مثلاً یکی از پرسش‌ها مربوط به مزايا (privileges) می‌باشد که اعداد بالاتر نشان‌دهنده رضایت بیشتر فرد از وضعیت مزايا در سازمان است.

چنانچه در توضیحات مربوط به فرایند دسته‌بندی بدون ناظر گفته شد هدف از خوشبندی قرار دادن داده‌هایی با شباهت بیشتر به هم در یک مجموعه مشترک است پس در این مسئله ما می‌خواهیم با در نظر گرفتن تنها دو ویژگی موجود در مجموعه داده که همان learning و privileges است داده‌ها را خوشبندی کنیم که از روی این خوشبندی بتوان وضعیت مزايا و یادگیری در سازمان را مورد بررسی قرار داد.

> library(datasets)

در خط برنامه زیر از طریق تابع str() که مجموعه داده attitude را در آرگومان خود دارد می‌توان محتوای مجموعه داده را به صورت یک data frame مشاهده نمود که در آن لیست کلاس‌های درون مجموعه داده در ستون اول مشخص است

> str(attitude)

'data.frame': 30 obs. of 7 variables:

\$ rating	: num 43 63 71 61 81 43 58 71 72 67 ...
\$ complaints	: num 51 64 70 63 78 55 67 75 82 61 ...
\$ privileges	: num 30 51 68 45 56 49 42 50 72 45 ...
\$ learning	: num 39 54 69 47 66 44 56 55 67 47 ...
\$ raises	: num 61 63 76 54 71 54 66 70 71 62 ...
\$ critical	: num 92 73 86 84 83 49 68 66 83 80 ...
\$ advance	: num 45 47 48 35 47 34 35 41 31 41 ...

با دستور زیر نیز می‌توانید درباره مجموعه داده مطالعه کنید که البته اجرای قسمت اصلی الگوریتم وابسته به این خط برنامه نیست و می‌توانید از نوشتمن آن اجتناب کنید.

> summary(attitude)

rating	complaints	privileges	learning
Min. :40.00000	Min. :37.0	Min. :30.00000	Min. :34.00000
1st Qu. :58.75000	1st Qu. :58.5	1st Qu. :45.00000	1st Qu. :47.00000
Median :65.50000	Median :65.0	Median :51.50000	Median :56.50000

Mean :64.63333	Mean :66.6	Mean :53.13333	Mean :56.36667
3rd Qu.:71.75000	3rd Qu.:77.0	3rd Qu.:62.50000	3rd Qu.:66.75000
Max. :85.00000	Max. :90.0	Max. :83.00000	Max. :75.00000
raises	critical	advance	
Min. :43.00000	Min. :49.00000	Min. :25.00000	
1st Qu.:58.25000	1st Qu.:69.25000	1st Qu.:35.00000	
Median :63.50000	Median :77.50000	Median :41.00000	
Mean :64.63333	Mean :74.76667	Mean :42.93333	
3rd Qu.:71.00000	3rd Qu.:80.00000	3rd Qu.:47.75000	
Max. :88.00000	Max. :92.00000	Max. :72.00000	

با استفاده از تابع kmeans() در خط زیر ۲ خوشه را از داده‌های موجود ایجاد می‌نماییم که آرگومان dat مربوط به attitude[] است که آرگومان درون آن یک بردار با دو عدد است که اولی ۳ و نشان‌دهنده سومین متغیر درون مجموعه داده و دومین عدد که همان ۴ است نشان‌دهنده چهارمین متغیر درون مجموعه داده است. عدد ۲ در درون آرگومان تابع kmeans() مشخص‌کننده تعداد خوشه‌های مورد نیاز است که باید ایجاد گردد.

```
> dat = attitude[,c(3,4)]
```

```
> km1 = kmeans(dat, 2, nstart=100)
```

نتیجه اجرای الگوریتم نیز با دستور زیر به صورت نمودار به نمایش در می‌آید. در تابع plot() آرگومان main مشخص‌کننده عنوان نمودار است، عدد مربوط به آرگومان pch مشخص‌کننده شکل هر داده در نمودار است و cex اندازه هر نقطه مشخص‌کننده داده را تعیین می‌کند. در نمودار خروجی داده‌ها مربوط به متغیرهای learning و privileges است که داده‌های آنها براساس شباهت، در دو خوشه سبز و قرمز قرار گرفته‌اند.

```
> plot(dat,col=(km1$cluster +1),main="K-Means result with 2 clusters",
      pch=20, cex=2)
```



## ۷-۹- جرای الگوریتم k-means با استفاده از زبان پایتون

در برنامه زیر نحوه خوشه‌بندی مجموعه داده Iris را با استفاده از k-means می‌توانید مشاهده نمایید. باید توجه نمود که این الگوریتم از روی برچسب داده‌ها آنها را خوشه‌بندی نمی‌کند و در اینجا از قوانین دسته‌بندی بدون ناظر استفاده می‌شود.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import Kmeans
from sklearn import datasets
```

در دو خط زیر مجموعه داده iris به برنامه افزوده می‌شود که در این برنامه تمامی ویژگی‌ها و داده‌های موجود در مجموعه داده مورد استفاده قرار می‌گیرد.

```
iris = datasets.load_iris()
```

```
X = iris.data
```

در خط زیر تعداد خوشه‌هایی که می‌خواهیم در خروجی داشته باشیم مشخص می‌گردد.

```
estimators = [('k_means_iris_3', Kmeans(n_clusters=3))]
```

دو خط زیر برچسب‌های نمودار را مشخص می‌کنند.

```
fignum = 1
```

```
titles = ['3 clusters']
```

خطوط زیر یک نمودار سه بعدی برای نشان دادن خوشه‌ها ایجاد می‌نماید.

```
for name, est in estimators:
```

```
    fig = plt.figure(fignum, figsize=(4, 3))
```

```
    ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
```

```
    est.fit(X)
```

خطوط زیر داده‌های خوشه‌بندی شده را در محل خود در نمودار به صورت رنگی چاپ می‌نماید.

```
labels = est.labels_
```

```
ax.scatter(X[:, 3], X[:, 0], X[:, 2],
```

```
c=labels.astype(np.float), edgecolor='k')
```

خطوط زیر تنها برچسب‌هایی که اسمای ویژگی‌ها هستند را در نمودار چاپ می‌نمایند.

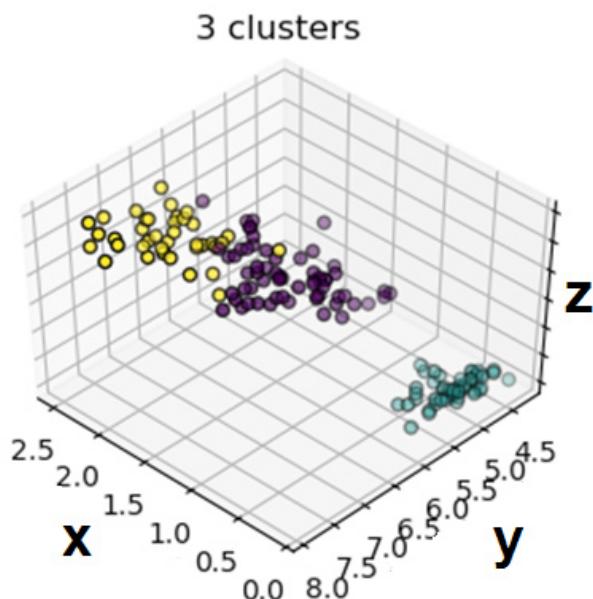
```
ax.set_xlabel('x')
```

```
ax.set_ylabel('y')
```

```
ax.set_zlabel('z')
```

```
fig.show()
```

نتیجه خوشبندی در نمودار زیر قابل مشاهده می‌باشد.



نمودار بالا در سه محور x، y و z ترسیم می‌شود که معرف ابعاد ویژگی‌ها می‌باشد.





فصل

## تحلیل مؤلفه‌های اصلی

## ۱-۱- تحلیل مؤلفه‌های اصلی<sup>۱</sup>

بررسی مجموعه داده‌ها، ساده‌سازی و یافتن الگوی حاکم بر متغیرها مهم‌ترین مرحله از فرآیند مدل‌سازی و حل مسئله است. تحلیل مؤلفه‌های اصلی یکی از انواع روش‌های تحلیل داده‌های چند متغیره است که هدف اصلی آن تقلیل بعد مسئله مورد مطالعه است. با استفاده از تحلیل مؤلفه‌های اصلی می‌توان تعداد زیادی متغیر مستقل همبسته را با تعداد محدودی متغیر مستقل جدید که مؤلفه‌های اصلی نامیده می‌شوند و ناهمبسته‌اند، جایگزین نمود.

## ۱-۲- کاربردهای تحلیل مؤلفه اصلی

به صورت کلی، کاهش ابعاد ویژگی‌های مسئله باعث بالاتر رفتن سرعت الگوریتم یادگیری می‌شود. در فشرده‌سازی اطلاعات با حذف ویژگی‌هایی با درجه اهمیت پایین‌تر، حجم داده‌ها و فضای ذخیره‌سازی کاهش یافته و علاوه بر طراحی سریع‌تر و ساده‌تر الگوریتم، می‌توان در مسائل مربوط به بهینه‌سازی، از آن برای جلوگیری از بیش‌برازش استفاده نمود. همچنین برای مصورسازی و به دست آوردن درک بهتر از داده‌ها می‌توان از تحلیل مؤلفه‌های اصلی استفاده نمود که این روش به خصوص در شرایطی که ابعاد داده‌ها و ترکیب ساختار آنها کاملاً مشخص نیست مفید است.

## ۱-۳- حذف ویژگی و کاهش ابعاد

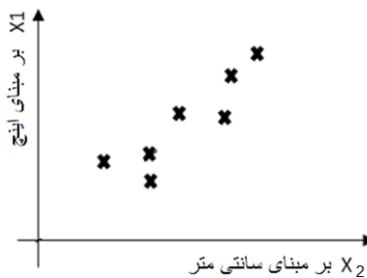
روش‌های کاهش بعد داده، یک فضای چند بعدی را به یک فضای با ابعاد کمتر نگاشت می‌کنند. در واقع با ترکیب مقادیر ویژگی‌های موجود، تعداد کمتری ویژگی بوجود می‌آورند بطوری که این ویژگی‌ها دارای تمام (یا بخش اعظمی) از اطلاعات موجود در ویژگی‌های اولیه باشند. این روش‌ها به دو دسته‌ی خطی و غیرخطی تقسیم می‌شوند که تحلیل مؤلفه‌های اصلی یکی از روش‌های خطی است. روش‌های خطی بدنیال یافتن یک زیرفضای تخت عمومی<sup>۲</sup> هستند.

مهم‌ترین نکته در کاهش ابعاد ویژگی‌ها مانند زمانی که ویژگی‌های ما متعلق به یک تصویر است حفظ کیفیت تصویر اولیه می‌باشد، هر چند در بسیاری از موارد حفظ کیفیت اولیه به صورت کامل امکان‌پذیر نمی‌باشد، لیکن از دست دادن حداقل اطلاعات به عنوان یکی از اهداف کاهش بعد، همواره مد نظر بوده است.

## ۱-۴- روش کاهش تعداد ویژگی

در این بخش برای آنکه مسئله کاهش ابعاد ویژگی را معرفی کنیم در قالب مثال‌هایی نحوه تبدیل یک فضای چندبعدی به ابعاد کمتر را معرفی می‌نماییم.

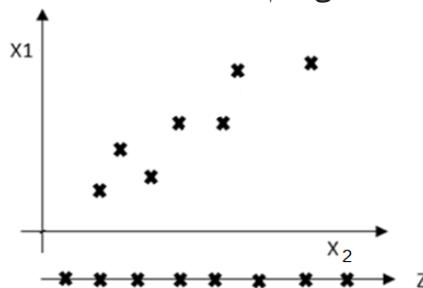
مثال) فرض کنید که دو ویژگی موجود را بخواهیم به یک ویژگی تبدیل کنیم و یا به عبارتی تعداد ویژگی‌ها را از دو به یک مورد تغییر دهیم که برای این کار به توضیحات زیر توجه نمایید.



شکل (۱-۸): نمودار فضای داده‌ای با دو ویژگی.

همان‌طور که در شکل (۱-۸) مشاهده می‌شود دو بردار موجود اندازه می‌باشد که این اندازه‌ها در یک نوع نیستند و براین اساس مورد مناسبی برای تبدیل این دو ویژگی به یک ویژگی می‌توانند باشند که همیشه برای کاهش بعد شرایط به این صورت نیست و بعضی وقت‌ها ویژگی‌ها می‌توانند از یک نوع نباشند و در حالتی دیگر نیز حتی ویژگی‌ها می‌توانند کاملاً مانند هم باشند یعنی حالت افرونگی ویژگی‌ها را داشته باشیم که می‌توانیم عمل کاهش بعد را بر آنها اعمال کنیم.

برای تبدیل دو بعد نمودار شکل (۱-۸) به یک بعد ابتدا باید خطی با نام Z را با مناسب‌ترین حالت در میان داده‌های این دو بعد به وجود آوریم و سپس داده‌های هر دو بعد را به آن نسبت دهیم که در اصل داده‌ها به این خط افکنش<sup>۱</sup> می‌شوند. در نهایت چنانچه در شکل (۲-۸) مشاهده می‌نمایید خط Z که همان محصول تلفیق داده‌های دو بعد اولیه بود به عنوان یک بردار با نام Z ایجاد می‌شود و به این صورت در محاسبات به جای شرکت دادن دو ویژگی قبلی از ویژگی Z استفاده می‌کنیم.

شکل (۲-۸): ایجاد یک بردار با نام Z از دو بردار  $X_1$  و  $X_2$ .

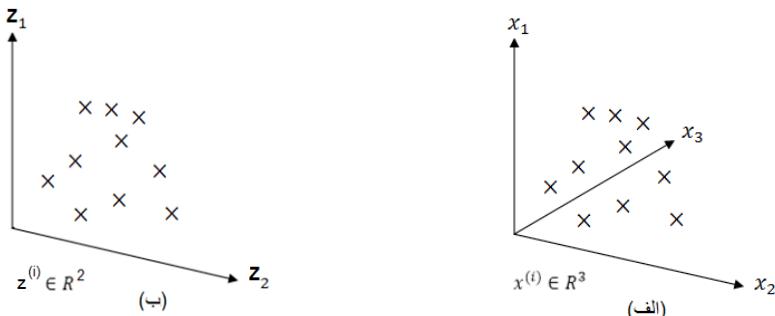
به صورت ریاضی عبارت زیر گویای توضیحات مسئله تبدیل دو بعد به یک بعد است که نشان می‌دهد دو بردار را با هم ترکیب کرده و یک بردار ایجاد می‌نماییم.

$$X_1, X_2 \in R^2 \rightarrow Z_1 \in R^1$$

مثال) تبدیل یک فضای ویژگی سه بعدی به یک فضای ویژگی دو بعدی.

فرض کنید که ما سه ویژگی داریم که می‌خواهیم تعداد آن را کاهش دهیم. در اینجا ما یک فضای سه بعدی  $X \in R^3$  داریم به عبارتی سه بردار با نام‌های  $x_1$  و  $x_2$  و  $x_3$  داریم که می‌خواهیم آنها را به دو بردار  $x_1$

و  $\text{z}_2$  تبدیل کنیم.



شکل (۳-۸): شکل (الف) یک فضای سه بعدی و شکل (ب) یک فضای دو بعدی از شکل (الف) است.

چنانچه در شکل (۳-۸)(الف) مشاهده می‌نمایید داده‌های موجود در یک فضای سه بعدی را به یک صفحه دو بعدی مانند شکل (۳-۸)(ب) افکنش می‌کنیم و یک فضای دو بعدی از ویژگی‌ها ایجاد کرده‌ایم که می‌توان آن را به صورت بردار  $z^{(i)}$  زیر نوشت.

$$x_1, x_2, x_3 \in R^3 \rightarrow z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

به بیان دیگر  $z^{(i)}$  یک افکنش از یک فضای سه بعدی است که در آن دو ویژگی وجود دارد و آن را به صورت  $z_1^{(i)}$  و  $z_2^{(i)}$  در نظر می‌گیریم و در کل  $z^{(i)}$  یک بردار با دو ویژگی است.

## ۴-۱- مصورسازی داده‌ها با استفاده از تحلیل مؤلفه‌های اصلی

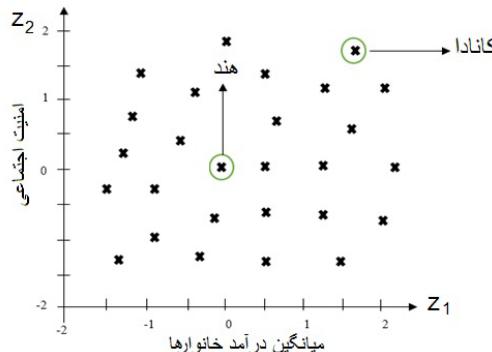
به تصویر کشیدن وضعیت داده‌ها به صورت نموداری نیز یکی از کاربردهای کاهش ابعاد ویژگی است. در برخی شرایط ما نیاز داریم که شرایط موجود یک مسئله را با به تصویر کشیدن و تحلیل داده‌های مربوط به ویژگی‌ها مورد بررسی قرار دهیم. باید توجه داشت که اگر یک مسئله دارای بیش از ۱۰۰ ویژگی باشد فضای برداری آن نیز دارای ۱۰۰ بعد است حال اگر آن را در یک نمودار بخواهیم نشان دهیم عملاً تحلیل آن به صورت دیداری غیرممکن است. زیرا ما معمولاً برای به تصویر کشیدن از یک، دو یا سه بعد استفاده می‌کنیم که برای ما قابل فهم است. پس چنانچه گفته شد می‌توان از قابلیت کاهش بعد برای ایجاد یک فضای نموداری قابل مشاهده استفاده کرد. به مثال زیر توجه نمایید.

مثال) فرض کنید جدول (۱-۸) زیر مربوط به وضعیت زندگی در کشورهای مختلف می‌باشد و چنانچه مشاهده می‌کنید تعداد ۵ ویژگی وجود دارد که می‌تواند بیش از ۱۰۰ ویژگی یا بیشتر نیز باشد.

جدول (۱-۸): شرایط زندگی در کشورهای مختلف.

کشور	میانگین درآمد خانوار	فقر	امنیت اجتماعی	نرخ رشد رفاه	نرخ تورم
کانادا	66.25	24.23	81.54	0.891	0.05
چین	9.26	35.02	72.66	0.459	0.8
هند	0.652	48.56	54.69	0.125	0.12
...	...	...	...	...	...

همان طور که می‌دانیم نشان دادن داده‌هایی با تعداد بالایی از ویژگی‌ها موضوع سختی می‌باشد که در این حالت می‌توانیم با تبدیل تمامی ویژگی‌ها به دو ویژگی اساسی، آنها را به صورت کلی به تصویر کشیده و مورد مطالعه قرار دهیم. به عبارتی  $\mathbf{x}^{(i)} \in R^2$  تبدیل می‌کنیم. در اصل به جای استفاده از تمامی ویژگی‌های هر کشور دو ویژگی مهم را به عنوان دو ویژگی برتر انتخاب و بقیه را حذف می‌کنیم. به نمودار شکل (۴-۸) توجه نمایید.



شکل (۴-۸): نمودار نشان‌دهنده کیفیت زندگی در کشورهای مختلف بر اساس دو ویژگی.

همان طور که در شکل (۴-۸) مشاهده می‌نمایید دو بعد یا دو ویژگی از تمامی ویژگی‌های موجود را نگه داشته و بقیه ویژگی‌ها را حذف کرده‌ایم. مثلاً برای داده مربوط به کشور کانادا می‌توان فهمید که از هر دو نقطه نظر در بالاترین وضعیت است ولی کشور هند از نقطه نظر این دو ویژگی مقداری در رتبه پایین‌تری قرار گرفته است.

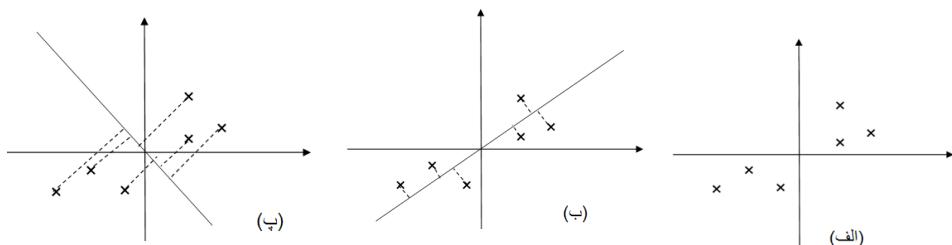
## ۱-۵-۱- نحوه عملکرد الگوریتم تحلیل مؤلفه اصلی

در تحلیل مؤلفه‌های اصلی تلاش بر این است که مؤلفه‌های اصلی<sup>۱</sup>، تلاش می‌کند تا بهترین سطحی را که بتوان داده‌ها را به آن افکنش کرد را پیدا کند. به این نکته توجه کنید که معمولاً قبل از انجام حذف ویژگی و کاهش بعد، عمل نرمال‌سازی ویژگی‌ها باید انجام شود.

مثال) فرض کنید نمودار شکل (۱-۸)(الف) را داریم و می‌خواهیم داده‌ها به یک بردار افکنش شود. برای این کار بهترین حالت قرارگیری بردار را در میان داده‌ها پیدا کرده و آن را ترسیم می‌کنیم و در امتداد آن خطی ترسیم

1. Principal component

می‌شود که خط مورد نظر ما برای افکنش کردن داده به آن است. اما مسئله این است که بهترین حالت قرارگیری برداری که می‌خواهیم بر آن افکنش انجام شود را چگونه انجام دهیم. به شکل زیر توجه نمایید.

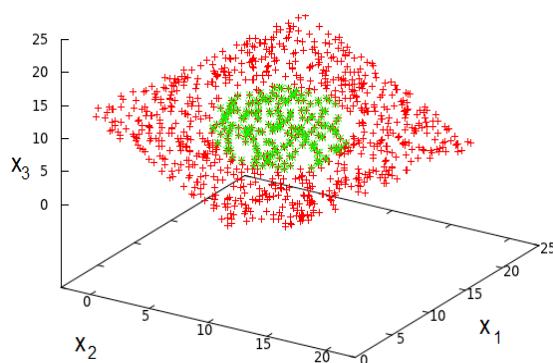


شکل (۸-۵): نمودارهای مربوط به حالات مختلف قرارگیری برداری که به آن افکنش داده می‌شود.

فرض کنید نمودار ما مانند شکل (۸-۵)(الف) باشد. چنانچه در نمودار (ب) مشاهده می‌نمایید یک بردار ترسیم شده، که فاصله کمی با هر کدام از داده‌ها دارد که گویای یک حالت مناسب برای قرارگیری برداری که می‌خواهیم افکنش را بر آن انجام دهیم می‌باشد. در حالی که در نمودار (پ) می‌توان دید که داده‌ها در فاصله زیادی با خط ترسیم شده قرار دارند پس الگوریتم در افکنش دارای خطای بالایی خواهد بود و راستای این خط برای عمل افکنش داده‌ها مناسب نمی‌باشد. به صورت کلی می‌توان گفت که هرچه برداری که داده‌ها را به آن افکنش می‌کنیم فاصله کمی با داده‌ها داشته باشد به این معنی است که خطای افکنش پایین‌تری داریم و در این حالت یک افکنش بهتری خواهیم داشت.

## ۱-۶- کاهش سه بعد به دو بعد

اگر بخواهیم از یک فضای سه بعدی یک سطح دو بعدی پیدا کرده و استخراج کنیم چنانچه در شکل (۶-۸) مشاهده می‌نمایید باید دو بردار در فضای سه بعدی ایجاد کنیم که هر کدام از بردارها دارای خطای افکنش پایینی باشند و همچنین از نقطه آغازین به هم متصل باشند. در این حالت یک صفحه دو بعدی به وجود می‌آید که از طریق آن صفحه دو بعدی می‌توان داده‌ایی که افکنش شده‌اند را استخراج کرد که بر این اساس یک فضایی دو بعدی از داده‌ای افکنش شده را خواهیم داشت.



شکل (۶-۸): افکنش داده‌ایی که در سه بعد قرار دارند بر روی یک صفحه دو بعدی.

توجه کنید که خطای افکنش در این فضای سه بعدی نیز همان فاصله داده‌ها از هر نقطه‌ای از فضای سه بعدی تا به صفحه دو بعدی است.

## ۷-۱-۱- مراحل اجرای الگوریتم تحلیل مؤلفه اصلی

در الگوریتم تحلیل مؤلفه اصلی اگر ویژگی‌ها را به صورت  $x^{(1)}, x^{(2)}, \dots, x^m$  داشته باشیم. قبل از اجرای الگوریتم باید مرحله پیش پردازش شامل مقیاس‌گذاری و نرمال‌سازی ویژگی‌ها را انجام دهیم که روش انجام آن در بخش مربوطه توضیح داده شده است.

الگوریتم تحلیل مؤلفه اصلی به صورت کلی تلاش می‌کند تا بردارهای  $u^{(i)}$  را در یک فضای ویژگی‌ها پیدا کند. پس از آن پردازش بر روی بردار یا فضای به دست آمده  $Z$  خواهد بود. حال در ادامه خواهیم دید که این پردازش‌ها و محاسبات برای به دست آوردن بردارها به صورت ریاضی چگونه انجام می‌شود.

**الگوریتم تحلیل مؤلفه اصلی:**

نرمالیزاسیون میانگین

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

هر کدام از مثال‌ها را از میانگین به دست آمده کم می‌کنیم.

$$x^{(i)} - \mu$$

از این پس  $x^{(i)}$ ‌ها همین نمونه‌هایی است که میانگین از آنها کم شده است.

محاسبه ماتریس کواریانس<sup>۱</sup>

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)}) (x^{(i)})^T$$

علامت  $\Sigma$  برای ماتریس کواریانس می‌باشد و با حرف کوچک سیگما که برای جمع مقادیر است فرق دارد. محاسبه بردار ویژه<sup>۲</sup> از ماتریس کواریانس ( $\Sigma$ ) این مرحله در ادامه تشریح می‌شود ولی با یکی از توابع زبان R که به صورت زیر می‌باشد این کار را می‌توان انجام داد.

eigen()

$$z^{(i)} = u_{reduce}^T x^{(i)} \quad \text{تبديل n به k بعد}$$

1. Covariance matrix
2. Eigenvector

در الگوریتم تحلیل مؤلفه اصلی ما  $n$  بعد داده را داریم که آن را به  $k$  بعد تبدیل می‌کنیم این تعداد  $k$  در اصل پارامتر الگوریتم می‌باشد و همچنین به نام تعداد مؤلفه‌های اصلی<sup>۱</sup> نیز شناخته می‌شود یا تعداد مؤلفه‌های اصلی که ما استخراج کرده‌ایم.

## ۱-۱- روشهای دست آوردن تعداد مؤلفه‌های اصلی (k)

برای به دست آوردن  $k$  باید میانگین مربعات خطای افکنش<sup>۲</sup> به دست آید که در اصل تحلیل مؤلفه اصلی قصد کاهش آن را دارد که با رابطه زیر انجام می‌شود.

ASPE → Average square projection error

$$\text{ASP Error: } \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2$$

در رابطه بالا  $x^{(i)}$  داده‌های اولیه و  $x_{approx}^{(i)}$  داده‌های افکنش شده می‌باشد و واریانس کل داده‌ها را نیز از رابطه زیر به دست می‌آوریم.

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$$

و در نهایت از طریق رابطه زیر برای بدست آوردن مقدار  $k$  استفاده می‌کنیم.

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2}$$

همان‌طور که می‌دانیم  $k$  تعداد مؤلفه‌های اصلی است که باید تعدادش درست انتخاب شود و برای بدست آوردن مقدار صحیح برای  $k$  الگوریتم زیر مورد استفاده قرار می‌گیرد.

الگوریتم محاسبه  $k$   
}

با  $k=1$  شروع کن.

محاسبه کن  $z^{(1)}, \dots, z^{(m)}$  و  $u_{reduce}, x_{approx}^{(1)}, \dots, x_{approx}^m$

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01$$

بررسی برقراری شرط

پایان

}

به عبارتی مقدار  $k$  ابتدا از یک عدد کوچکی مانند ۱ شروع شده و در رابطه مربوطه قرار می‌گیرد و اگر نتیجه رابطه

- 
1. Number of Principal component
  2. Average square projection error

موردنظر که همان تقسیم میانگین مربعات خطای افکنش بر واریانس کل داده‌ها است مقداری کوچک‌تر یا مساوی ۱۰۰٪ شود مقدار  $k$  همان یک انتخاب می‌شود و در غیر این صورت مقدار  $k$  تا زمانی که نتیجه تقسیم میانگین خطای افکنش به واریانس کل داده‌ها به مقدار مناسب نرسیده است مقادیر دیگری به  $k$  اختصاص داده می‌شود. توجه نمایید که مقدار ۱۰٪ یعنی ۹۹ درصد از واریانس حفظ شده که مقدار مناسبی است.

## ۹-۱- برگشت داده‌های حذف شده

تا به اینجا از تحلیل مؤلفه اصلی به عنوان روشی برای کاهش بعد صحت کردیم که باعث کاهش ابعاد مسئله می‌شد. همان‌طور که می‌دانیم داده‌های فشرده شده ممکن است نیاز به برگشتن به حالت اولیه پیدا کند که مراحل انجام این کار را در ادامه خواهیم دید.

## ۹-۱-۱- مراحل عکس فشرده‌سازی

باید بر عکس کارهایی که برای حذف ویژگی‌ها انجام می‌دادیم عمل کنیم تا بتوانیم ویژگی‌های حذف شده را بازسازی کنیم.

هنگام فشرده‌سازی ابتدا  $\mathbf{z} = \mathbf{U}_{\text{reduce}}^T \mathbf{x}$  به دست می‌آمد که  $\mathbf{z}$  کاهش یافته حالت چند بعدی بود.

به عبارتی  $\mathbf{z} \in \mathbb{R}^k$  بود که ما قصد تبدیل آن به حالت عکس یعنی  $\mathbf{x} \in \mathbb{R}^n$  را داریم که برای این کار به روابط زیر توجه نمایید.

$$\mathbf{x}_{\text{approx}}^{(i)} = \mathbf{U}_{\text{reduce}} \mathbf{z}^{(i)} + \boldsymbol{\mu}$$

با این رابطه به ازای هر بردار ورودی یک  $\mathbf{x}_{\text{approx}}^{(i)}$  تخمین زده شده با عنوان  $\mathbf{x}^{(i)}$  به دست می‌آید و خطای ناشی از افکنش داده‌ها با رابطه زیر محاسبه می‌شود.

$$\text{Projection error: } \frac{1}{m} \sum_{i=1}^m \|\mathbf{x}^{(i)} - \mathbf{x}_{\text{approx}}^{(i)}\|^2$$

همچنین مانند رابطه زیر اگر از ماتریس  $\mathbf{U}$  به جای ماتریس  $\mathbf{U}_{\text{reduce}}$  در تبدیل  $n$  بعد به  $k$  بعد استفاده کنیم یک فضای  $n$  بعدی به یک فضای  $k=n$  بعدی دیگر می‌رود یعنی هیچ اطلاعاتی را از دست نمی‌دهیم به عبارتی فقط یک تغییر مختصات بر روی داده‌ها اعمال می‌شود و در این حالت می‌توان تمامی داده‌ها را دوباره بازیابی کرد.

$$\mathbf{x}^{(i)} = \mathbf{U} \mathbf{z}^{(i)} + \boldsymbol{\mu}$$

توجه کنید که اگر در کاهش بعد به گونه‌ای عمل شود که صدرصد واریانس داده‌ها حفظ شود در این حالت داده‌های بازیابی شده برابر نسخه اصلی خواهد بود و در غیر این صورت نسخه برگردانده شده به اندازه واریانس حفظ شده شبیه نسخه اولیه خواهد بود.

## ۱۰-۱- مثال عملی برای تحلیل مؤلفه اصلی با زبان R

در این برنامه هدف تحلیل داده‌های آموزشی موجود در مجموعه داده iris است که می‌خواهیم ویژگی‌های موجود در آن را تحلیل نموده و مقادیر مربوط به مؤلفه‌های اصلی را بیابیم. دلیل آنکه در این بخش نیز از مجموعه داده iris استفاده می‌نماییم این است که شما در فصل‌های گذشته با ساختار آن آشنا شده‌اید و آشنایی با ساختار مجموعه داده در درک بهتر این الگوریتم اثر مثبت دارد.

ابتدا براساس خطوط زیر بسته‌های نرم‌افزاری و مجموعه داده iris را بارگذاری می‌نماییم و برای مشاهده شش ردیف از نمونه موجود در مجموعه داده از تابع head() استفاده می‌نماییم.

```
> install.packages("ggbioplot", "vqv")
> library(ggbioplot)
> library(devtools)
> data(iris)
> head(iris, 6)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Widt	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

از طریق دو خط برنامه زیر تمامی نمونه‌های موجود در مجموعه داده را به دو قسمت تقسیم می‌نماییم. قسمت اول مربوط به داده‌های مربوط به هر چهار ویژگی موجود در مجموعه داده می‌باشد که در log.iris ذخیره می‌نماییم و قسمت دوم فقط شامل نام گونه‌ها می‌باشد که در iris.species ذخیره می‌گردد تا بعداً برای مصورسازی مؤلفه‌های اصلی مورد استفاده قرار گیرد.

```
> log.iris <- log(iris[, 1:4])
> iris.species <- iris[, 5]
```

از طریق تابع prcomp() که برای تحلیل مؤلفه‌های اصلی از روش تجزیه مقادیر مجرد (SVD) استفاده می‌کند، مدل اصلی PCA را ایجاد می‌نماییم.

```
> iris.pca <- prcomp(log.iris, center = TRUE, scale. = TRUE)
```

خروجی تابع prcomp() نیز به صورت زیر قابل چاپ و مشاهده می‌باشد.

```
> print(iris.pca)
```

Standard deviations:

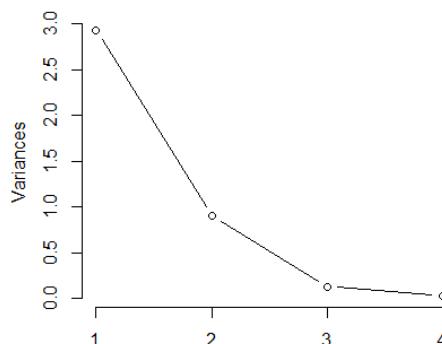
```
[1] 1.7124583 0.9523797 0.3647029 0.1656840
```

Rotation:

	PC1	PC2	PC3	PC4
Sepal.Length	0.5038236	-0.45499872	0.7088547	0.19147575
Sepal.Width	-0.3023682	-0.88914419	-0.3311628	-0.09125405
Petal.Length	0.5767881	-0.03378802	-0.2192793	-0.78618732
Petal.Width	0.5674952	-0.03545628	-0.5829003	0.58044745

در خروجی بالا مقادیر مربوط به انحراف‌معیار (standard deviations) مشخص گردیده و همچنین در بخش چرخش (rotation)، مقادیر مربوط به چهار مؤلفه اصلی نمایش داده شده است که ضرایبی از ترکیب خطی از متغیرهای پیوسته می‌باشند و نمودار خروجی به دست آمده به صورت زیر می‌باشد که مشخص‌کننده میزان واریانس برای هر مؤلفه اصلی است و چنانچه مشاهده می‌نمایید مؤلفه اصلی اول و مؤلفه اصلی دوم، واریانس بالاتری نسبت به مؤلفه سوم و چهارم دارد.

```
> plot(iris.pca, type = "l")
```



خروجی تابع زیر مشخص‌کننده اهمیت هر کدام از مؤلفه‌های اصلی است. در خط اول، نتیجه مقادیر انحراف‌معیار، در ردیف دوم نسبت واریانس و در ردیف سوم مقادیر مربوط به نسبت تجمعی مشخص گردیده است.

```
> summary(iris.pca)
```

Importance of components:

	PC1	PC2	PC3	PC4
Standard deviation	1.7125	0.9524	0.36470	0.16568
Proportion of Variance	0.7331	0.2268	0.03325	0.00686
Cumulative Proportion	0.7331	0.9599	0.99314	1.00000

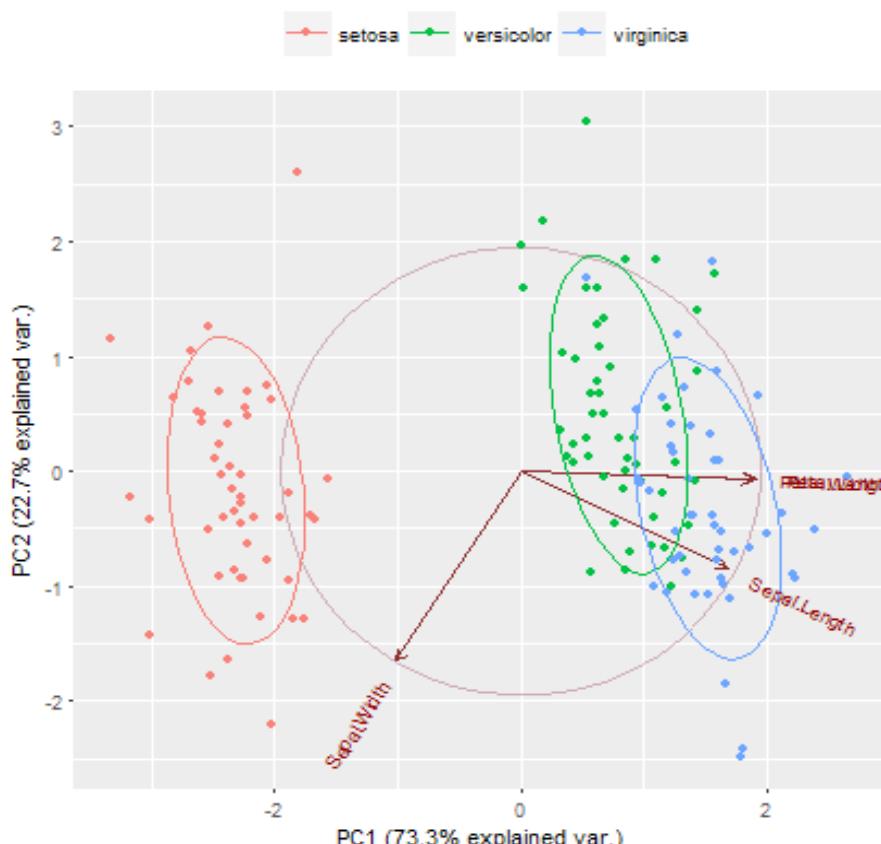
از طریق تابع زیر نیز می‌توان مقادیر مؤلفه اصلی را برای داده‌های جدید به دست آورد که ما در این مثال با استفاده از تابع tail() از دو ردیف آخر موجود در مجموعه داده iris به عنوان داده جدید فرضی استفاده نموده‌ایم.

```
> predict(iris.pca, newdata = tail(log.iris, 2))
```

	PC1	PC2	PC3	PC4
149	1.0809930	-1.01155751	-0.7082289	-0.06811063
150	0.9712116	-0.06158655	-0.5008674	-0.12411524

با استفاده از خطوط برنامه زیر نیز می توانید نمودار مولفه های اصلی را برای سه گونه موجود گیاه زنبق ترسیم نمایید که توابع استفاده شده برای این مرحله، مربوط به بسته ggbioplot می باشد که برای اطلاع از جزئیات توابع آن می توانید به مستندات مربوط به آن مراجعه نمایید.

```
> g <- ggbioplot(iris.pca, obs.scale = 1, var.scale = 1, groups = iris.species, ellipse = TRUE, circle = TRUE)
> g <- g + scale_color_discrete(name = "")
> g <- g + theme(legend.direction = 'horizontal', legend.position = 'top')
> print(g)
```



## ۱۱-۸- مثال عملی برای تحلیل مؤلفه اصلی با زبان پایتون

در این برنامه هدف اجرای الگوریتم PCA می‌باشد که مجموعه داده iris را که دارای ۴ بعد ویژگی می‌باشد. تحلیل نموده و آن را به ۳ بعد کاهش می‌دهد و در یک نمودار ۳ بعدی چاپ می‌نماید.

در پنج خط اول برنامه زیر بسته‌های نرم‌افزاری مورد نیاز و مجموعه داده به حافظه بارگذاری می‌شود.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from sklearn import decomposition
from sklearn import datasets
```

```
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

خطوط برنامه زیر با استفاده از تابع (`Axes3D()`) یک نمودار سه بعدی می‌سازد که داده‌ها را درون آن به تصویر کشید. باید توجه نمود که مجموعه داده موجود دارای ۴ بعد است که در اینجا باید به ۳ بعد کاهش می‌باید.

```
fig = plt.figure(1, figsize=(4, 3))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
plt.cla()
```

خطوط زیر عملیات اصلی کاهش ابعاد ویژگی را انجام می‌دهند و محصول نهایی تبدیل یافته را در `x` ذخیره می‌نمایند.

```
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)
```

داده‌های درون `x` با شرکت در برنامه زیر که یک حلقه `for` است برچسب نقاط را درون نمودار ترسیم مشخص می‌نماید.

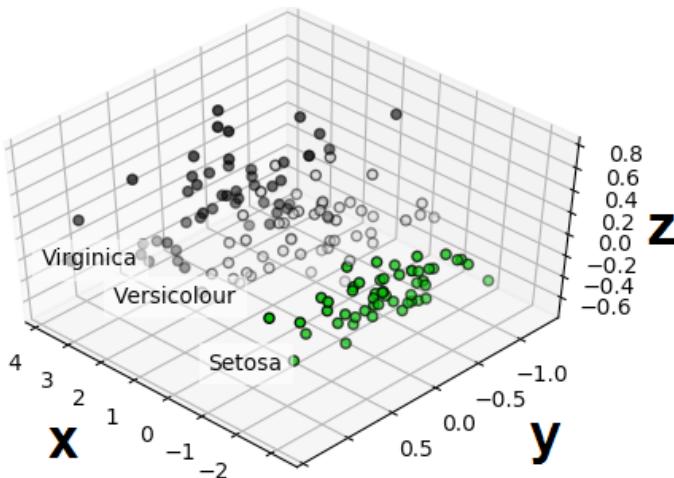
```
for name, label in [('Setosa', 0), ('Versicolour', 1), ('Virginica', 2)]:
    ax.text3D(X[y == label, 0].mean(),
              X[y == label, 1].mean() + 1.5,
              X[y == label, 2].mean(), name,
              horizontalalignment='center',
              bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))
```

خطوط برنامه زیر نیز نقاط داده‌ای را به صورت نمودار پراکندگی در نمودار چاپ می‌نماید که هر داده با رنگ مشخصی در بخش مربوط به خود قرار دارد.

```
y = np.choose(y, [1, 2, 0]).astype(np.float)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y, cmap=plt.cm.nipy_spectral,
           edgecolor='k')

plt.show()
```

نمودار زیر خروجی نهایی الگوریتم می‌باشد.



۹

فصل

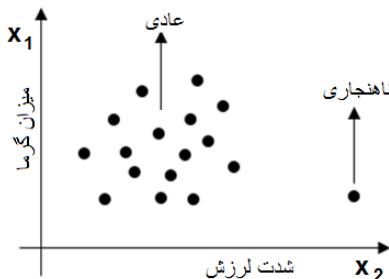
## تشخیص ناهمجاري

## ۱-۹- تشخیص ناهنجاری<sup>۱</sup>

تشخیص ناهنجاری روندی برای یافتن الگوهایی در یک مجموعه داده است که رفتار طبیعی مورد انتظار را ندارند. به این رفتارهای غیرمنتظره ناهنجاری گفته می‌شود. تشخیص ناهنجاری یکی از مسائل رایج یادگیری ماشین است. تشخیص ناهنجاری را می‌توان به سه بخش بدون نظارت، با نظارت و همچنین تشخیص ناهنجاری نیمه‌نظراتی دسته‌بندی کرد.

فرض کنید یک کارخانه تولید موتور هوایپیما برای آزمون نهایی موتورهای تولید شده از نظر مناسب بودن وضعیت کیفی، بخواهد آنها را براساس ویژگی‌های زیر مورد آزمون قرار دهد. نمودار مربوط به ویژگی‌های مطرح شده نیز به صورت شکل (۱-۹) است.

$$\begin{aligned} X_1 &= \text{گرمای تولید شده} \\ X_2 &= \text{شدت لرزش به وجود آمده} \\ &\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \end{aligned}$$

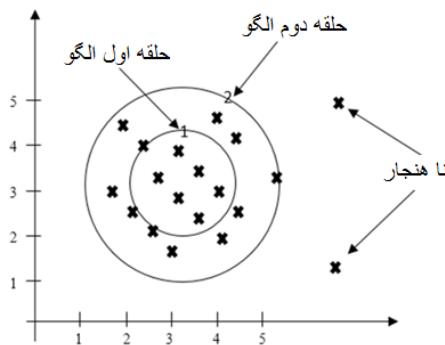


شکل (۱-۹): نمودار نشان دهنده داده‌های هنجار و ناهنجار.

به صورت کلی می‌توان گفت که با آزمون وضعیت موتورهای تولید شده که برای این موضوع ویژگی‌های مختلف موتور مورد بررسی قرار می‌گیرد مجموعه داده‌هایی جمع‌آوری می‌شود که با تحلیل آنها می‌توانیم ناهنجاری موجود در وضعیت موتورها را تشخیص دهیم. معمولاً ویژگی‌های موجود در هر مسئله مشخص است و بر آن اساس داده‌ها جمع‌آوری می‌شوند چنانچه در شکل (۱-۹) مشاهده می‌نمایید دو ویژگی میزان گرمای و شدت لرزش مدنظر قرار دارد و در فضای برداری این دو ویژگی، داده‌ای که از بقیه داده‌ها کمی فاصله دارد را یک ناهنجاری در نظر می‌گیریم. در اصل منطقه مناسب جایی است که داده‌ها در مرکز مدل ایجاد شده باشند و هرچه داده‌ها که براساس آزمون وضعیت موتور به دست آمده‌اند از مرکز مدل ما دورتر شوند احتمال نرمال بودن آن داده کمتر می‌شود و به احتمال ناهنجار بودنش می‌افزاید.

براساس شکل (۲-۹) حلقه اول درون نمودار بیانگر منطقه مربوط به داده‌هایی است که دلالت به هنجار بودن با درجه احتمال بالایی می‌باشند همچنین حلقه دوم نیز داده‌های هنجار را با درجه احتمال پایین‌تری شامل می‌شود

ولی داده‌های خارج از آن بیانگر ناهنجاری می‌باشد. به عبارتی داده‌ها در نمودار اگر از یک حد آستانه‌ای خارج شوند که در اینجا ما حلقه دوم را در مدل خود در نظر گرفته‌ایم، آن داده‌ها را ناهنجار اعلام می‌کنیم.



شکل (۲-۹): مدل مفهومی منطقه مشخص کننده داده‌های هنجار و داده‌های ناهنجار.

## ۲ - ۹ - کاربردهای مختلف تشخیص ناهنجاری

از تشخیص ناهنجاری در مسائل مختلفی می‌توان استفاده کرد که برخی از آنها عبارتند از تشخیص نفوذ سایبری<sup>۱</sup>، تشخیص تقلب<sup>۲</sup>، تشخیص ناهنجاری‌های پزشکی<sup>۳</sup>، تشخیص آسیب صنعتی<sup>۴</sup> و در زمینه‌های پردازش متن و پردازش تصویر نیز کاربرد گسترده‌ای دارد.

### ۲ - ۹ - ۱ - تشخیص تقلب

یکی از کاربردهای معروف تشخیص ناهنجاری، تقلب‌شناسی می‌باشد که از این روش برای تشخیص جعل عملیات بانکی و تشخیص رفتارهای غیرعادی کاربران اینترنتی و غیره می‌توان استفاده کرد.

فرض کنید یک سیستم نمایش وضعیت<sup>۵</sup> در یک مرکز داده<sup>۶</sup> داریم که کارش نظارت بر نحوه عملکرد ماشین‌ها و دستگاه‌های موجود در این مرکز است و یکی از کارهای آن نظارت بر ارتباطات برقرار شده به یک منبع داده خاص می‌باشد.

بر این اساس تمامی اتصالات بر مبنای یک الگوی مشخص شده انجام خواهد شد و اگر هرگونه اتصالی به این منبع داده که از الگوی مشخص شده پیروی نکند وجود داشته باشد آن را یک ناهنجاری در نظر می‌گیریم و عملاً وقتی یک تلاش موفقی برای دسترسی به این منابع داده انجام شود که به دور از الگوی مشخص شده و تعیین شده است، معرفی یک تقلب است و تشخیص این وضعیت را تشخیص تقلب گویند که براساس یک تشخیص ناهنجاری به دست آمده است.

1. Cyber-intrusion detection
2. Fraud detection
3. Medical fraud detection
4. Industrial damage detection
5. Monitoring
6. Data center

مثال) اگر در مسئله تشخیص ناهنجاری در مرکز داده، همان ویژگی‌های ماشین آم باشد که میزان استفاده از حافظه، تعداد دسترسی به دیسک ذخیره‌سازی در واحد ثانیه، استفاده از منابع پردازنده مرکزی<sup>۱</sup>، ترافیک شبکه باشد، به روند تشخیص ناهنجاری در ادامه مطالب توجه نمایید.

حال براساس این چهار ویژگی موجود از ماشین آم شرایط آن را از نظر عادی و یا غیرعادی بودن تحلیل می‌کنیم و اگر نتایج تحلیل نشان دهد که داده‌ها در خارج از الگوی استاندارد ایجاد شده هستند ما می‌توانیم آن را یک تقلب و یا ناهنجاری در نظر بگیریم. مثلاً بالا رفتن غیرعادی و خیلی زیاد حجم پردازش در پردازنده مرکزی به صورت لحظه‌ای می‌تواند ناشی از یک ایراد فنی و یا حتی ناشی از یک نوع حمله خرابکارانه به نام انکار سرویس<sup>۲</sup> نیز باشد که با بالا بردن خیلی زیاد تراکنش شبکه و منابع آن مرکز داده را غیرقابل استفاده می‌کند.

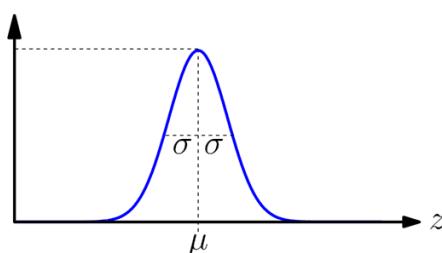
### ۹-۳-۱- مباحث ریاضی برای تعیین ناهنجاری

برای آنکه بتوانیم محدوده فوارگیری داده‌ها را در حالت عادی یا غیرعادی مشخص کنیم از یک سری از قوانین و توابع ریاضی استفاده می‌کنیم که یکی از این توابع چنانچه در بخش مربوط به ماشین بردار پشتیبان معرفی شد تابع گوسی است که به کاربر آن در تشخیص ناهنجاری در ادامه مطالب توجه نمایید.

### ۹-۳-۱-۱- تابع گوسی

در نظریه احتمالات یکی از مهم‌ترین توزیع‌های احتمالی پیوسته تابع گوسی نام دارد که این تابع نوسان‌های طبیعی و فیزیکی پیرامون یک مقدار ثابت و مقادیر حاصل از این توزیع را نشان می‌دهد. تابع احتمال این توزیع دارای دو پارامتر میو و سیگما می‌باشد که تعیین‌کننده مکان و تعیین‌کننده مقیاس توزیع است و میانگین توزیع با پارامتر مکان و پراکندگی آن با پارامتر مقیاس برابر است و منحنی تابع احتمال حول میانگین توزیع متقارن است. در حالت خاص اگر و باشد این حالت توزیع طبیعی نرمال یا عادی نامیده می‌شود.

شکل (۳-۹) نشان‌دهنده تابع گوسی می‌باشد.



شکل (۳-۹): منحنی تابع گوسی.

رابطه تابع توزیع احتمال که برای شناسایی و توزیع یک احتمال به کار بردہ می‌شود به صورت زیر می‌باشد.

$$f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x-\mu^2}{2\sigma^2}\right) \quad x \in \mathbb{R} \quad \text{رابطه ۱}$$

که در رابطه بالا معرف میانه یا مد و واریانس می‌باشد. تابع  $f(x)$  همان تابع متقارن حول محور  $x$  است و همچنین این نقطه میانگین مد و میانه می‌باشد و نقاط عطف این منحنی به صورت زیر می‌باشد.

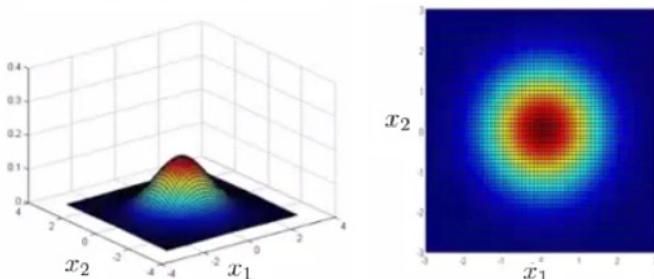
$$X = \mu - \sigma \quad \text{یا} \quad X = \mu + \sigma$$

حال اگر بخواهیم بگوییم که  $x$  عضو اعداد طبیعی و توزیعی از تابع گوسی نرمال با نقطه مطلق و انحراف معیار باشد آن را به صورت زیر می‌نویسیم.

$$X \sim N(\mu, \sigma^2) \quad \text{رابطه ۲}$$

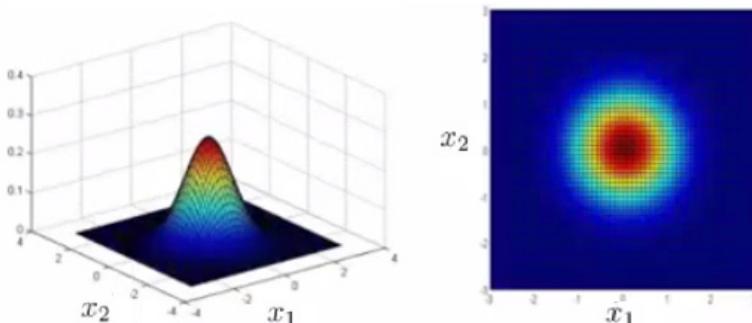
به چند مورد از نمودارهای تابع گوسی که در شکل (۴ و ۵ و ۶) آمده توجه نمایید.

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



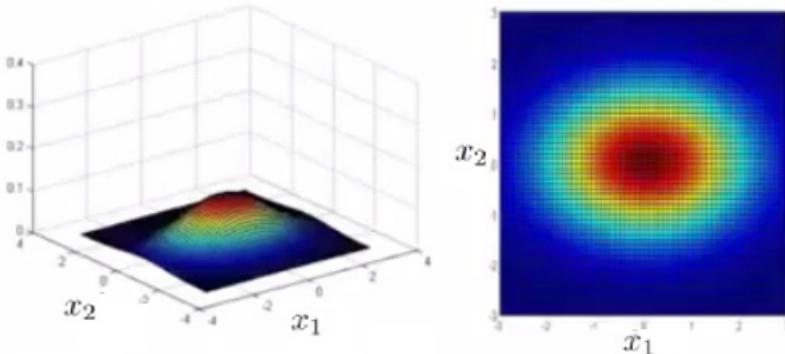
شکل (۴-۹): وضعیت توزیع طبیعی استاندارد برای تابع گوسی.

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$$



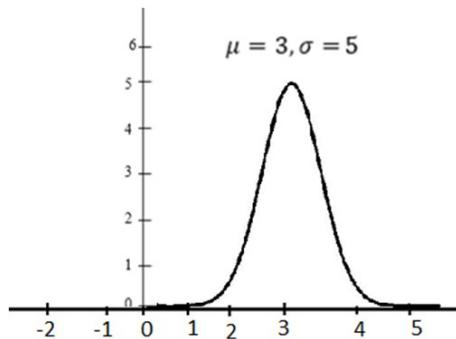
شکل (۵-۹): در این شکل سیگما کم و عرض منحنی نیز کم است و ارتفاع منحنی زیاد است.

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$



شکل (۶-۹): در این شکل سیگما زیاد و عرض منحنی زیاد است و ارتفاع منحنی کم است.

همان‌گونه که در سه نمودار فوق ملاحظه می‌کنید هرچه مقدار سیگما کم می‌شود عرض منحنی کمتر و ارتفاع آن بیشتر می‌شود و هرچه سیگما بیشتر می‌شود شرایط برعکس اتفاق می‌افتد و چنانچه در شکل (۷-۹) می‌بینید مشخص‌کننده محل قرارگیری مرکز تقارن منحنی است و اگر پارامتر را تغییر دهیم نقطه مرکزی منحنی بروی محور  $x$  تغییر خواهد کرد.



شکل (۷-۹): تابع گوسی وقتی که برابر ۵ و برابر ۳ می‌باشد.

### ۹-۳-۲- تخمین پارامترهای تابع گوسی

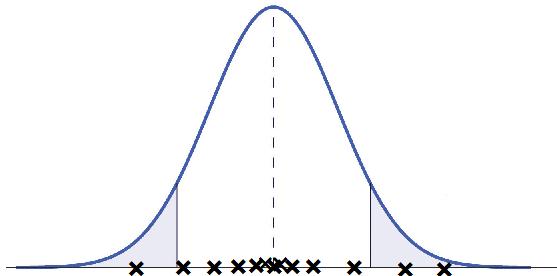
تخمین مقدار درست برای اختصاص دادن به پارامترهای  $\mu$  و  $\sigma^2$  یکی از موضوعات مهم در استفاده تابع گوسی برای حل مسائل می‌باشد. فرض کنید مجموعه داده زیر در دست است.

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$

حال فرض کنید رابطه زیر مفروض است.

$$x^{(i)} \sim N(\mu, \sigma^2)$$

به عبارتی  $x$  توزیعی ازتابع نرمال است ولی در این مسئله پارامترهای  $\mu$  و  $\sigma^2$  را نداریم و هدف تخمین این پارامترها و پیدا کردن بهترین حالت برای تابع است فرض کنید براساس مجموعه داده در دست نمودار شکل (۸-۹) را داریم.



شکل (۸-۹): تابع گوسی که در آن در محل تجمع داده‌ها است.

همان‌گونه که مشاهده می‌نمایید بهترین حالت می‌تواند قرار دادن مرکز منحنی در بخشی باشد که بیشترین تجمع داده را داریم و تخمین پارامترهای  $\mu$  و  $\sigma^2$  این عمل را برای ما انجام می‌دهد.

روابط تخمین پارامترهای تابع گوسی به صورت زیر می‌باشد.

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)} \quad \text{رابطه (۳)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)^2 \quad \text{رابطه (۴)}$$

### ۹-۳-۳- الگوریتم تشخیص ناهنجاری

با توجه به مطالعی که برای تابع گوسی معرفی شد حالا در ادامه مطالعه ملاحظه خواهید کرد که چگونه می‌توان یک الگوریتم تشخیص ناهنجاری را با استفاده از تابع گوسی به وجود آورد.

تخمین چگالی<sup>۱</sup>

یکی از موارد مورد نیاز الگوریتم تشخیص ناهنجاری، تخمین چگالی می‌باشد که مشخص کننده وضعیت مثال‌های آموزشی مورد مطالعه براساس قوانین تابع گوسی می‌باشد.

اگر مثال‌های آموزشی  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  که  $x \in R$  است را داشته باشیم حالا اگر بخواهیم  $p(x)$  را به دست آوریم باید از روابط زیر استفاده کنیم.

$$x_i \sim N(\mu_i, \sigma_i^2) \quad \text{رابطه (۵)}$$

$$p(x_1; \mu_1, \sigma_1^2) \dots p(x_n; \mu_n, \sigma_n^2) \quad \text{رابطه (۶)}$$

مثال) اگر سه مثال آموزشی  $x_1, x_2, x_3$  داشته باشیم روابط آن به صورت زیر خواهد بود.

$$x_1 \sim N(\mu_1, \sigma_1^2) \rightarrow p(x_1; \mu_1, \sigma_1^2)$$

$$x_2 \sim N(\mu_2, \sigma_2^2) \rightarrow p(x_2; \mu_2, \sigma_2^2)$$

$$x_3 \sim N(\mu_3, \sigma_3^2) \rightarrow p(x_3; \mu_3, \sigma_3^2)$$

$$P(x) = ?$$

$$P(x) = p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) p(x_3; \mu_3, \sigma_3^2)$$

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

در رابطه فوق از علامت  $\Pi$  برای ضرب استفاده می‌شود.

حال با افزودن این روابط و ترکیب آنها با هم الگوریتم اصلی را به وجود می‌آوریم.

#### ۴-۳-۴- مراحل الگوریتم تشخیص ناهنجاری

انتخاب ویژگی‌های ( $x_i$ ) مناسب که براساس مسئله آن ویژگی‌ها می‌توانند اشاره‌گر به داده‌هایی باشند که تعیین‌کننده شرایط ناهنجاری باشد.

تنظیم پارامتر برای  $\mu_1, \mu_2, \dots, \mu_n$  و  $\sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \sum_{i=1}^m x_j^{(i)} \quad \text{(الف)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2 \quad \text{(ب)}$$

براساس هر مثال آموزشی  $x$  جدید  $p(x)$  را محاسبه کن.

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^n \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right) \quad \text{(الف)}$$

تشخیص ناهنجاری اگر  $p(x_{test}^{(i)}) < p(x)$  باشد. به عبارتی اگر  $p(x_{test}^{(i)}) > p(x)$  را داشته باشیم ناهنجاری وجود دارد و اگر پایان.

اپسیلن ( $\epsilon$ ) پارامتری است که مقدارش تعیین‌کننده حد آستانه‌ای برای تعیین ناهنجاری است که باید به صورت مناسب مقداردهی شود.

#### ۴-۴- ارزیابی عملکرد الگوریتم تشخیص ناهنجاری

در یک الگوریتم یادگیری با نظارت برای ارزیابی الگوریتم همان‌طور که در مباحث آغازین اشاره شد ارزیابی

الگوریتم در حین اجرای مرحله آموزش توسط روشی به نام وارسی اعتبار<sup>۱</sup> انجام می‌شود که صحت الگوریتم را قبل از رسیدن به مرحله آزمون تقریب بزنده و در مرحله آزمون الگوریتم ارزیابی اصلی برای نحوه عملکرد کلی الگوریتم انجام می‌شود. در این الگوریتم نیز به همین صورت فرایند ارزیابی انجام می‌شود.

فرض کنید با مسئله‌ای روبه رو هستیم که در آن داده‌های ما دارای برچسب می‌باشند و داده‌های نرمال و داده‌های ناهنجار، با برچسب مشخص شده‌اند. اگر خروجی الگوریتم را با  $y$  نشان دهیم شرط زیر برقرار است.

$$Y = \begin{cases} 0 & \text{if non anomaly} \\ 1 & \text{if anomaly} \end{cases}$$

مثلاً اگر نمونه‌های آموزشی مانند  $x^{(1)}, x^{(2)}, \dots, x^{(m)}$  را که داده‌های نرمال هستند را داشته باشیم روابط مجموعه ارزیابی آن به صورت زیر خواهد بود که در آنها  $cv$  مخفف وارسی اعتبار<sup>۲</sup> است در مرحله آموزش الگوریتم اتفاق می‌افتد و  $test$  بیانگر ارزیابی در مرحله آزمون است.

$$(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{m_{cv}}, y_{cv}^{m_{cv}}) \quad \text{رابطه (۷)}$$

و رابطه مجموعه آزمون و ارزیابی آن به صورت زیر خواهد بود.

$$(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{m_{test}}, y_{test}^{m_{test}}) \quad \text{رابطه (۸)}$$

چنانچه در قسمت اول این بخش با مثال وضعیت موتور هواپیما آشنا شده‌اید که داده‌ها در آنجا دارای برچسب نبود، در اینجا فرض ما بر این است که داده‌های مربوطه برچسب خورده باشند و از کل داده‌های اولیه حدود ۱۰۰۰۰ داده نرمال و حدود ۲۰ داده ناهنجار داشته باشیم در این حالت می‌توان گفت که این نسبت نشان‌دهنده وضعیت قابل قبول برای موتور است.

در مرحله آزمون الگوریتم از ۱۰۰۰۰ نمونه داده موجود که هنجار می‌باشند و ۲۰ داده ناهنجار به شرح زیر در مرحله آموزش و مرحله آزمون استفاده می‌کنیم.

برای تقسیم داده‌های اولیه چنانچه در فصل‌های گذشته گفته شد کل داده‌ها را می‌توان به صورت ۷۰ درصد مرحله آموزش و بقیه را برای مرحله آزمون استفاده کرد در اینجا با توجه به اینکه داده‌ها برچسب‌دار می‌باشند و داده‌های ناهنجار و داده‌های غیرناهنجار مشخص می‌باشند همیشه باید مقداری از داده‌های ناهنجار را نیز برای مرحله آزمون نگهداشت تا در مرحله آزمون بتوان توانایی الگوریتم در تشخیص داده‌های ناهنجار را براساس داده‌های ناهنجار جدید ارزیابی کرد.

## ۹-۵- مثالی برای تشخیص ناهنجاری با کدهای R

در این برنامه هدف تحلیل داده‌ها و یافتن داده‌های ناهنجار یا داده‌های پرت است. یک الگوریتم خوب باید بتواند داده‌های پرت را تشخیص داده و از آنها در محاسبه نتیجه نهایی الگوریتم استفاده نکند. در اینجا می‌خواهیم تعداد بازدید شدن کلمه fifa را در سایت Wikipedia تحلیل کنیم که در یک بازه زمانی مشخص اتفاق می‌افتد تمامی داده‌هایی که درون بازه زمانی مشخص شده قرار گرفته باشند داده‌های نرمال و داده‌هایی که نشان می‌دهد کلمه fifa خارج از آن محدوده زمانی مشاهده شده است یک داده غیرنرمال در نظر گرفته می‌شود. سایت ویکی‌پدیا امکان اتصال زبان R را به منابع خود فراهم نموده است که بر این اساس می‌توان از داده‌های آن استفاده نمود که در برنامه زیر مشاهده می‌نماییم.

از بسته‌های نرمافزاری Rcpp و devtools و AnomalyDetection و wikipediaTrend و ggplot2 برای محاسبه و نمایش داده‌های ناهنجار استفاده شده است که نحوه نصب و بارگذاری آن را در خطوط اول برنامه مشاهده می‌نمایید.

```
> install.packages("devtools")
> devtools::install_github("petermeissner/wikipediaTrend")
> devtools::install_github("twitter/AnomalyDetection")
> install.packages("Rcpp")
> library(devtools)
> library(Rcpp)
```

بسته زیر دارای API‌هایی است که امکان ارتباط برنامه ما با سایت ویکی‌پدیا را به وجود می‌آورد.

```
> library(wikipediaTrend)
```

بسته‌های زیر برای محاسبه داده‌های ناهنجار و نمایش نموداری آنها به کار می‌روند.

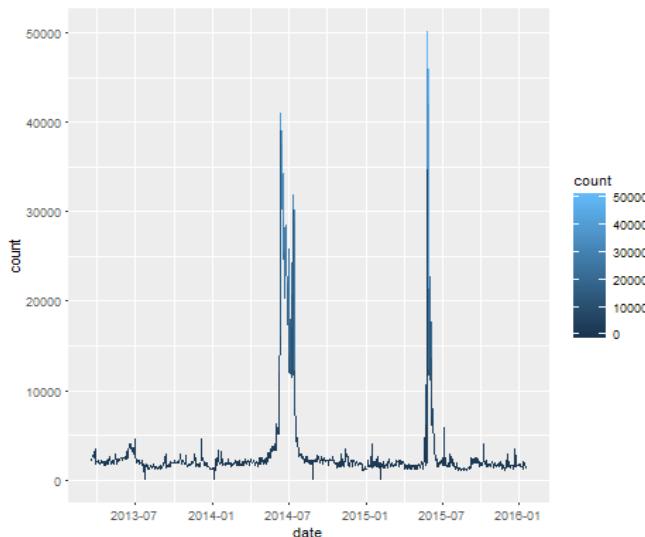
```
> library(AnomalyDetection)
> library(ggplot2)
```

خط برنامه زیر برای دانلود کلمه fifa از سایت ویکی‌پدیا مورد استفاده قرار می‌گیرد و از آنجایی که نیاز داریم تا تعداد کلمه fifa را در یک بازه زمانی مشخص محاسبه نماییم، در تابع wp\_trend() آرگومان اول کلمه‌ای را مشخص می‌کند که می‌خواهیم آن را جستجو و تحلیل نماییم، آرگومان دوم تاریخی است که می‌خواهیم داده‌های موجود را از آن زمان به بعد به دست آوریم و در آرگومان سوم نوع زبان کلمه را مشخص می‌نماییم.

```
> fifa_data = wp_trend("fifa", from="2013-03-18", lang = "en")
```

از طریق تابع زیر می‌توانیم داده‌های به دست آمده را به صورت نموداری نمایش دهیم.

```
> ggplot(fifa_data, aes(x=date, y=count, color=count)) + geom_line()
```



خطوط برنامه زیر داده‌ها را به صورت متغیری از بازه زمانی و صفحات بازدید شده سازماندهی و مشخص می‌نماید.

```
> fifa_data$date = as.POSIXct(fifa_data$date)
```

خط زیر از مجموعه داده تهیه شده تنها داده‌های مربوط به تاریخ و صفحات مشاهده شده را برای استفاده انتخاب می‌نماید.

```
> fifa_data=fifa_data[,c(1,2)]
```

در خروجی متغیر زیر تاریخ و تعداد مشاهده شدن صفحات مربوطه مشخص شده است.

```
> fifa_data
    date      count
648 2014-12-25 03:30:00    998
985 2015-11-28 03:30:00   1104
289 2013-12-31 03:30:00   1191
1035 2016-01-17 03:30:00   1248
700 2015-02-15 03:30:00   1298
293 2014-01-04 03:30:00   1344
...
429 2014-05-20 04:30:00   3237
812 2015-06-07 04:30:00   6105
804 2015-05-30 04:30:00  21300
```

... 1009 rows of data not shown

خط زیر تشخیص ناهنجاری را روی داده‌ها اعمال می‌کند.

```
> data_anomaly = AnomalyDetectionTs(fifa_data, max_anoms=0.01,
direction="pos", plot=TRUE, e_value = T)
```

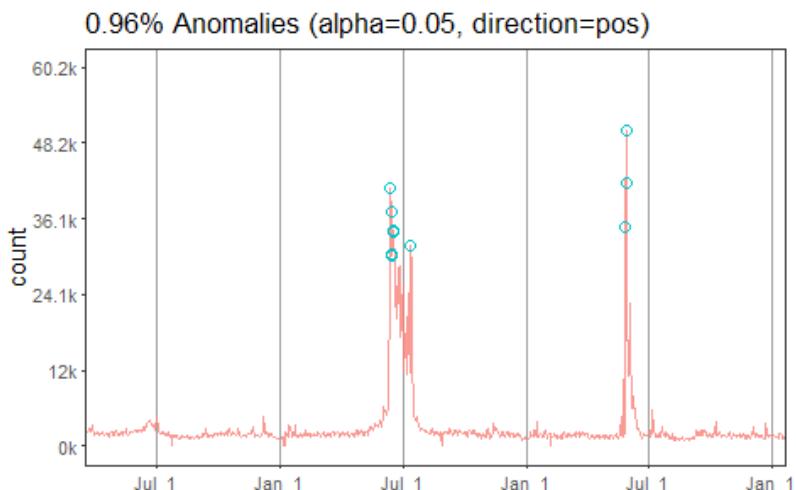
خروجی به صورت زیر خواهد بود که نشان‌دهنده زمان مربوطه، تعداد داده‌های پرت، مقادیر مورد انتظار و نمودار مربوطه است.

```
> data_anomaly
```

\$anoms

	timestamp	anoms	expected_value
1	2014-06-12 04:30:00	40976	25146
2	2014-06-13 04:30:00	37283	30379
3	2014-06-14 04:30:00	30160	29673
4	2014-06-15 04:30:00	30430	29262
5	2014-06-16 04:30:00	34028	28553
6	2014-06-17 04:30:00	34277	27645
7	2014-07-13 04:30:00	31869	17007
8	2015-05-27 04:30:00	34737	34736
9	2015-05-28 04:30:00	50166	38391
10	2015-05-29 04:30:00	41930	41930

\$plot



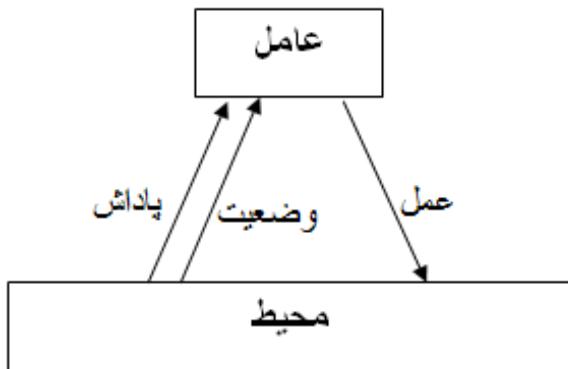
١٠  
فصل

## یادگیری تقویتی

## ۱-۱- یادگیری تقویتی

یادگیری تقویتی پیرامون مباحثی بحث می‌کند که چگونه یک عامل خودکار بتواند به صورت هوشمند و با تکیه بر توانایی‌های درونی خود محیط را حس کرده و رفتار مناسب و بهینه‌ای را برای تأثیرگذاری مناسب بر محیط از خود نشان دهد و بر این اساس بتواند به هدف نهایی خود برسد. این حوزه یکی از زیرشاخه‌های یادگیری ماشین به حساب می‌آید که جامعیت تعیین‌پذیری برای بسیاری از مسائل مربوط به بهینه‌سازی<sup>۱</sup> الگوریتم‌ها و همچنین پیش‌بینی‌ها<sup>۲</sup> را دارد. مانند شرایطی که در آن یک عامل هوشمند مصنوعی در یک شرایط متغیر و متنوع قرار دارد و هدف آن است که بتواند براساس پیش‌بینی‌هایی که انجام می‌دهد رفتاری را برای اتخاذ انتخاب کند که در نهایت بر اساس آن به هدف نهایی خود برسد.

در این روش هر وقت عامل عملی را در محیط خود انجام می‌دهد یک آموزندهای دارد که مدام به انتخاب‌های عامل پاداش و جریمه در نظر می‌گیرد به این صورت که برای یک انتخاب راه حل بهینه و خوب، امتیاز بالا و یک انتخاب راه حل غیرمناسب و یا بد، امتیاز پایین‌تر اختصاص می‌دهد که نشان دهد برای رسیدن به نتیجه مطلوب‌تر نهایی کدام اعمال بهترین انتخاب‌ها بوده‌اند که در ادامه عامل باید تلاش کند تا از تجربیات به دست آمده تشخیص دهد که کدام دنباله‌ای از اعمال دارای بالاترین امتیاز هستند و آن را به عنوان بهترین راه موجود برای رسیدن به جواب مسئله یا هدف اصلی خود انتخاب کند.



شکل (۱-۱۰): عملکرد کلی الگوریتم یادگیری تقویتی.

## ۱-۲- حوزه عملکرد

یادگیری تقویتی در حال حاضر در حوزه‌های متنوعی مانند داده کلای، پیش‌بینی، رباتیک، کلاسه‌بندی تصاویر، استخراج ویژگی، تقریب توابع، مسیریابی، تشخیص پزشکی و حیطه‌های بسیار دیگری مورد استفاده قرار می‌گیرد.

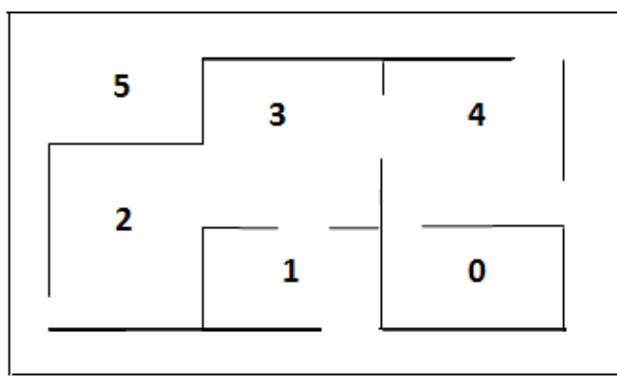
## ۱۰-۳- الگوریتم‌های یادگیری تقویتی

الگوریتم‌های متعددی در یادگیری تقویتی مورد استفاده قرار گرفته که می‌توان به تعدادی از آنها مانند روش یادگیری Q، کنترل بهینه<sup>۱</sup>، یادگیری تفاوت زمانی<sup>۲</sup> اشاره کرد که هر الگوریتم از نقطه نظر متفاوتی در رساندن عامل به نتیجه بهینه تلاش می‌کنند که در این کتاب ما روش یادگیری Q که یکی از معروف‌ترین الگوریتم‌های یادگیری تقویتی می‌باشد را توضیح می‌دهیم. در این روش تلاش بر روی به دست آوردن استراتژی‌های کنترل بهینه بر اساس امتیازات به دست آمده از پاداش‌ها می‌باشد که حتی عامل هیچ اطلاعات از پیش تعیین شده‌ای را در رابطه با تأثیرات انتخاب عمل خود در محیط ندارد.

## ۱۰-۴- روش یادگیری Q

یادگیری Q یکی از الگوریتم‌های رسیدن به جواب بهینه در یادگیری تقویتی است که با حل مسئله‌ای به وسیله آن نحوه عملکرد عامل هوشمند را نشان داده‌ایم و بر این اساس شما می‌توانید درک درستی از کاربرد یادگیری تقویتی را به دست آورید. در این مثال ما عامل هوشمندی را توصیف می‌کنیم که از یادگیری بدون نظارت برای درک و یادگیری یک محیط غیرآشنا استفاده می‌کند. فرض کنید ما پنج اتاق به هم چسبیده داریم که از طریق درب‌ها به هم راه دارند.

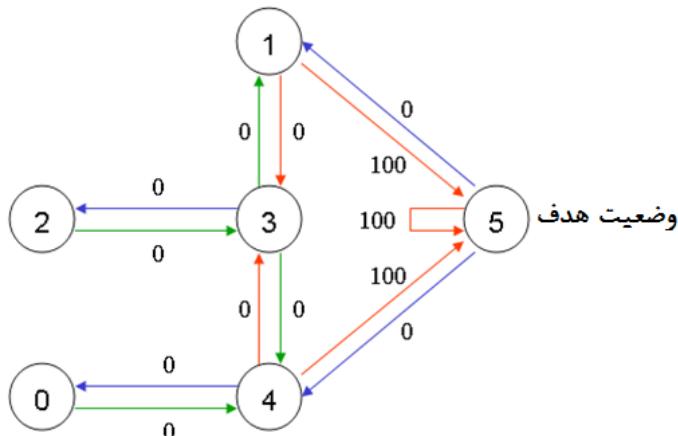
همان‌طور که در شکل شماره (۲-۱۰) مشاهده می‌کنید. به هر اتاق از ۰ تا ۴ عددی را اختصاص می‌دهیم و هر منطقه‌ای که خارج از این اتاق‌ها قرار دارد را با عدد ۵ مشخص می‌کنیم توجه شود که تنها از اتاق ۱ و ۴ به بیرون ساختمان یعنی منطقه ۵ راه دارد.



شکل (۲-۱۰): گرافی به شکل اتاق که در آن چندین مسیر به سمت بیرون وجود دارد.

همچنین می‌توان شکل بالا به صورت گراف ترسیم کرد که در آن هر اتاق یکی از گره‌های گراف می‌باشد که آن را در شکل (۳-۱۰) ملاحظه می‌نمایید.

- 
1. Optimal control
  2. Temporal difference learning



شکل (۳-۱۰): گراف و مقادیر مربوط به وضعیت هر مسیر.

در یادگیری Q دو واژه با عنوان وضعیت<sup>۱</sup> و همچنین عمل<sup>۲</sup> وجود دارد در این مثال ما به اتفاق‌ها و حتی شماره ۵ که بیرون است، یک "وضعیت" و به حرکت عامل از یک اتفاق به اتفاق دیگر یک "عمل" می‌گوییم. همچنین در گراف هم به هر گره یک وضعیت و هر یک از پیکان‌های جهت‌دار نیز با واژه عمل معرفی می‌شوند.

در اینجا ما قصد داریم عامل را در هر کدام از اتفاق‌ها قرار دهیم تا عامل از آن اتفاق مسیر خود را به بیرون پیدا کند به بیانی دیگر عامل بتواند از اتفاق‌های مختلف مسیر خود را به قسمت شماره ۵ پیدا کند. توجه شود که بعضی از اتفاق‌ها دارای یک در و بعضی دارای دو و بعضی دارای سه در هستند. ما به هر در یک مقدار عددی که همان مقدار پاداش می‌باشد را اختصاص می‌دهیم بر این اساس دربی که به صورت مستقیم عامل را به هدف که همان منطقه ۵ می‌باشد هدایت می‌کند را با عدد ۱۰۰ مقداردهی می‌کنیم و دربی که به صورت مستقیم به منطقه ۵ نمی‌رسد را با ۰ مقداردهی می‌کنیم. توجه شود که هر در موجود در اتفاق‌ها، دو جهت دارد مثلاً از اتفاق ۰ به ۴ و از اتفاق ۴ به ۰ که در گراف شکل (۳-۱۰) آن را با دو پیکان رفت و برگشتی نشان داده‌ایم و به همین ترتیب تمامی پیکان‌های جهت‌دار دارای مقداری هستند که به آن مقدار پاداش گویند.

حال فرض کنید عامل ما که یک ربات می‌باشد و آن را در اتفاق شماره ۲ قرار داده‌ایم، هیچ اطلاعاتی از وضعیت اتفاق‌ها ندارد و باید یاد بگیرد که چگونه به بیرون از اتفاق‌ها یعنی منطقه ۵ برود. بر اساس گراف موجود در شکل بالا عامل از وضعیتی که در اصل نقطه شروع آن می‌باشد یعنی وضعیت ۲، تنها می‌تواند به وضعیت ۳ برود به دلیل آنکه تنها به آن وضعیت راه دارد و در ادامه مسیر از وضعیت ۳ هم به وضعیت ۴ و هم به وضعیت ۱ راه دارد و باید تنها یکی را انتخاب کند. اگر عامل از وضعیت ۴ شروع کند و به وضعیت ۰ برود همان‌طور که در گراف می‌توان دید عامل از آنجا تنها می‌تواند به وضعیت ۴ برگردد و حرکت عامل از ۴ به ۰ یک عمل خطأ بوده است، ولی اگر از وضعیت ۴ به ۵ برود به هدف خود رسیده است.

- 
1. State
  2. Action

همان طور که در شکل گراف مشاهده می‌کنید همه مسیرهای مستقیم به اتاق یا گره شماره ۵ دارای مقدار ۱۰۰ و برگشت آنها دارای مقدار ۰ می‌باشد و همچنان گره ۵ دارای یک دور برگشت به خود است که مقدار ۱۰۰ گرفته است.

## ۱-۵-آموزش عامل در روش Q

ابتدا عامل باید در محیط مورد نظر آموزش ببیند و بعد از رسیدن به حالت بهینه که معمولاً زمان بر است مرحله آموزش به پایان می‌رسد در این مرحله معمولاً عامل تمامی حالات را امتحان کرده است.

به عبارتی عامل در مرحله آموزش تلاشش بر آن بوده که از گره آغازین شروع به پیدا کردن مسیر رسیدن به هدف کند و در هر بار تکرار، نتایج تجربیات خود را ثبت کند و طبق توضیحات داده شده برای هر مسیر، عددی را به عنوان پاداش اختصاص می‌دهد. هرچه عدد اختصاصی بزرگ‌تر باشد به معنای آن است که این مسیر بهینه است. حال در ادامه، اطلاعات به دست آمده در درون مغز عامل به عنوان یک راهنمای قرار داده می‌شود که از این به بعد عامل هر اقدام جدیدی برای حرکت از نقطه شروع به منظور رسیدن به نقطه پایان را آگاهانه و براساس مسیرهای آموزش دیده شده انجام می‌دهد که در این حالت عامل با حداقل هزینه و با سرعت بالایی به هدف خود می‌رسد. در عمل، عامل موردنظر باید به وضعیتی با بالاترین پاداش برسد که در اصل همان نقطه هدف یا منطقه خارج از اتاق‌ها می‌باشد.

## ۱-۶-مراحل الگوریتم یادگیری Q

الگوریتم Q در کل براساس مراحل زیر انجام می‌شود که در ادامه هر مرحله توضیح داده شده است.  
۱. تعیین مقدار مناسب برای نرخ یادگیری یا گاما

۲. ایجاد یک ماتریس با نام R که مقادیر پاداش‌ها و دیاگرام وضعیت را براساس گراف موجود در بر داشته باشد.  
۳. انجام مراحل زیر برای هر پیمایش مسیر از کل مسیرها {

۱-۳. انتخاب یک وضعیت به صورت تصادفی.

۲-۳. اجرای حلقه تا زمانی که عامل به وضعیت هدف نرسیده باشد.

}

۲-۳.۱. از میان تمامی عمل‌های ممکن یکی را برای وضعیت جاری انتخاب کن.

۲-۳.۲. براساس انتخابی که در مرحله ۱-۲-۳ انجام شد به مرحله بعد برو.

۲-۳.۳. براساس تمام حالات عمل ممکن، حالتی را که در آن Q در بیشترین مقدار است را انتخاب کن.

۲-۳.۴. مقدار جدید Q را محاسبه کن.

۲-۳.۵. وضعیت بعدی را به عنوان وضعیت فعلی قرار بده.

{

{

## ۱-۶-۱- نرخ یادگیری در روش یادگیری Q

در مرحله آموزش یکی از مهم‌ترین پارامترها نرخ یادگیری یا alpha می‌باشد این نرخ تعیین می‌کند که تا چه میزان اطلاعات بدست آمده جدید بر اطلاعات قدیمی ترجیح داده شود. مقدار صفر باعث می‌شود عامل چیزی یاد نگیرد و مقدار یک باعث می‌شود عامل فقط اطلاعات جدید را ملاک قرار دهد.

## ۱-۶-۲- عامل تخفیف<sup>۱</sup> در روش یادگیری Q

یکی از پارامترهای بسیار مهم دیگر این الگوریتم عامل تخفیف یا gamma است که برای آن یک عدد بین ۰ و ۱ تعیین می‌شود. عامل تخفیف، اهمیت پاداش‌های آینده را تعیین می‌کند که در آن مقدار صفر باعث می‌شود عامل ماهیت فرستطلبه‌نگرته و فقط پاداش‌های فعلی را مدنظر قرار دهد. در حالی که مقدار یک عامل را ترغیب می‌کند برای یک دوره زمانی طولانی برای پاداش تلاش کند.

## ۱-۶-۳- ماتریس R یا پاداش در یادگیری Q

مقادیر پاداش‌ها و دیاگرام وضعیت را به صورت یک ماتریس می‌نویسیم که در مرحله مقداردهی اولیه الگوریتم براساس شرایط موجود به وجود می‌آید.

## ۱-۶-۴- یک دورآموزشی از عمل عامل در آموزش Q

به حرکت عامل از یک نقطه اولیه و پیمودن یک مسیر از مسیرهای موجود و رسیدن به نقطه پایانی یا هدف را یک دورآموزشی گویند که این عمل می‌تواند مدام تکرار شود و عامل حالات مختلف را تجربه کند. بر این اساس عامل از گره‌های موجود در گراف، یکی را به عنوان نقطه شروع انتخاب می‌کند.

## ۱-۷- یک مثال مفهومی برای الگوریتم Q

برای درک بهتر الگوریتم یک دورآموزشی را به صورت دستی پیاده‌سازی می‌کنیم تا با نحوه عملکرد الگوریتم آشنا شویم. در این مسئله عامل از گراف یک شروع می‌کند و می‌خواهد به گراف پنجم برسد. مقدار اولیه را برای نرخ یادگیری ۰,۸ قرار می‌دهیم و همچنین نقطه شروع وضعیت یا گره یا اتاق ۱ می‌باشد.

در ابتدا ماتریس Q را ایجاد و آن را با مقادیر صفر مقداردهی اولیه می‌کنیم.

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

ماتریس  $R$  نیز به شکل زیر می‌باشد که همان وضعیت‌ها و عمل‌ها در آن براساس شناخت اولیه شرایط محیط مقداردهی شده است.

$$R = \begin{array}{c} \text{عمل} \\ \text{وضعیت} \end{array} \begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[ \begin{matrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{matrix} \right] \end{matrix}$$

در مقادیر داخل ماتریس عدد ۱- نشان‌دهنده این است که هیچ ارتباطی بین گره‌های مربوطه وجود ندارد و در اصلاح مقدار null به خود گرفته است. برای مثال از وضعیت ۰ نمی‌توان به وضعیت ۱ رفت. ردیف‌های ماتریس نشان‌دهنده وضعیت فعلی عامل و ستون ماتریس نشان‌دهنده اعمال ممکن برای رسیدن به مرحله بعدی است.

به ردیف دوم (وضعیت ۱) در ماتریس  $R$  نگاهی بیاندازید. در آنجا دو عمل ممکن برای وضعیت جاری که همان گراف یک است وجود دارد، می‌توان به وضعیت ۳ رفت و یا می‌توان به وضعیت ۵ رفت که ما به صورت تصادفی وضعیت ۵ را برای انجام عمل انتخاب کردی‌ایم. حال تصور کنید چه اتفاقی رخ می‌داد اگر عامل ما در وضعیت ۵ قرار داشت، همان‌طور که در ماتریس  $R$  مشاهده می‌شود سه عمل ممکن وجود دارد که بر آن اساس می‌توان به وضعیت ۱ یا وضعیت ۴ یا به وضعیت ۵ رفت. به رابطه (۱) برای محاسبه  $Q$  توجه نمایید.

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})] \quad (1)$$

$$Q(1, 5) = R(1, 5) + 8 * \text{Max}[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0 = 100$$

از آنجایی که ماتریس  $Q$  هنوز صفر است پس  $Q(5, 1), Q(5, 4), Q(5, 5)$  همه صفر می‌باشند.

ولی براساس مقدار اولیه  $(5, 1)$   $R$  جواب محاسبه  $(1)$   $Q$  برابر  $100$  می‌باشد.  
حال وضعیت بعد یعنی ۵، تبدیل به وضعیت جاری گردیده است و به دلیل آنکه وضعیت هدف همان وضعیت ۵ می‌باشد پس ما یک دور آموزشی را انجام داده و یک مسیر را پیموده و به هدف رسیدیم حال باید ماتریس  $Q$  را که مقداردهی اولیه صفر شده بود و در مغز عامل ما قرار دارد به مقدار جدید به روزرسانی کنیم.

$$Q = \begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

برای دور آموزشی بعدی به صورت تصادفی یکی از وضعیت‌ها انتخاب گردید که در این میان وضعیت انتخاب شده ۳ می‌باشد و اگر به ردیف چهارم از ماتریس  $R$  توجه کنید در آن سه عمل ممکن ۱، ۲ یا ۴ برای انتخاب وجود دارد که ما با انتخاب تصادفی وضعیت ۱ را انتخاب کرده‌ایم.

حال فرض می‌کنیم که در وضعیت ۱ قرار داریم و اگر به ردیف دوم از ماتریس  $R$  یا همان وضعیت ۱ نگاهی بیاندازیم برای این حالت دو عمل ممکن وجود دارد یا باید به وضعیت ۳ و یا باید به وضعیت ۵ برویم و در انتهای  $Q$  را محاسبه می‌کنیم به روابط زیر برای محاسبه  $Q$  توجه نمایید.

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$$

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Max}[Q(1, 2), Q(1, 5)] = 0 + 0.8 * \text{Max}(0, 100) = 80$$

چنانچه می‌بینید ماتریس  $Q$  بهروز شده است که در آن  $Q(1, 5) = 100$  و  $Q(1, 3) = 0$  و نتیجه محاسبه  $Q(3, 1) = 80$  می‌باشد و به دلیل اینکه مقدار قبلی پاداش یا امتیاز موجود ۰ بود پس ماتریس جدید  $Q$  به صورت زیر خواهد بود.

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

حال با توجه به مسیرهای موجود که در شکل (۳-۱۰) قابل مشاهده است، وضعیت بعدی که ۱ بود وضعیت جاری ما می‌باشد و به دلیل آنکه وضعیت ۱ وضعیت هدف نیست ما حلقه داخلی الگوریتم را مجددًا تکرار می‌کنیم. برای شروع حلقه از وضعیت ۱، دو عمل وجود دارد رفتن به وضعیت ۳ یا رفتن به وضعیت ۵ که براساس انتخاب تصادفی رفتن به وضعیت شماره ۵ انتخاب شد. حال مجددًا فرض بر آن است که در وضعیت ۵ قرار داریم و سه عمل ممکن برای انتخاب وجود دارد که عبارت است از رفتن به وضعیت ۱، ۴ یا ۵ و در ادامه براساس بالاترین مقدار سه عمل موجود ما مقدار  $Q$  را براساس رابطه (۱) به صورت زیر محاسبه می‌کنیم.

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Max}[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$$

ورودی‌های بهروزرسانی شده از ماتریس  $Q$ ،  $Q(5, 1)$ ،  $Q(5, 4)$ ،  $Q(5, 5)$  همه مساوی ۰ می‌باشند و بر اساس مقدار اولیه ماتریس  $R$  نتیجه محاسبه  $Q(1, 5)$  مساوی ۱۰۰ شده است پس در این حالت نتیجه به دست آمده ماتریس  $Q$  را تغییر نمی‌دهد. و به دلیل اینکه عامل در وضعیت هدف قرار دارد پس این دور آموزشی نیز به پایان می‌رسد.

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

و در نهایت پس از انجام مکرر این دورهای آموزشی ماتریس  $Q$  به حالت بهینه خود همگرا می‌شود و مرحله آموزش کلی به پایان می‌رسد و از آن به بعد عامل می‌تواند در هر وضعیتی که باشد با مراجعه به ماتریس  $Q$  مسیر بهینه را انتخاب کرده و به وضعیت هدف انتقال یابد.

## ۱-۱- برنامه یادگیری $Q$ با زبان R

در برنامه زیر هدف پیاده‌سازی یک الگوریتم یادگیری تقویتی با روش یادگیری  $Q$  است که در آن نحوه بهروزرسانی ماتریس پاداش را می‌توانید مشاهده نمایید.

این برنامه از دو بخش اصلی تشکیل شده است که در بخش اول یکتابع ایجاد شده که کارش ساخت یک الگوریتم یادگیری  $Q$  و فرایند بهروزرسانی ماتریس پاداش است و در مرحله دوم برنامه پارامترهایی را به عنوان آرگومان به تابع ایجاد شده در بخش اول برنامه می‌دهیم تا الگوریتم درون تابع با استفاده از مقادیر پارامترها شروع به کار کند و در خروجی خود نتیجه الگوریتم را که یک ماتریس پاداش بهروز شده است نمایش دهد.

### بخش اول برنامه و بدنه تابع QlearnigFunction()

باید به این نکته توجه نمایید که اسامی پارامترهای مورد نیاز برای یادگیری هم در زمان ایجاد تابع و هم در زمان اجرای آن تابع باید در آرگومان تابع QlearnigFunction() نوشته شوند که هر کدام از این آرگومان‌ها در زیر توضیح داده شده است.

آرگومان یا پارامتر  $R$  همان ماتریس پاداش را شامل می‌شود،  $N$  مشخص کننده تعداد تکرار هر عمل جستجوی بهینه است،  $\alpha$  نرخ یادگیری و  $\gamma$  به عنوان عامل تخفیف مورد استفاده قرار گرفته است. در نهایت آرگومان  $tgt.state$  مشخص کننده وضعیت هدف است.

خط اول برنامه تابع را نام‌گذاری می‌کند.

```
QlearnigFunction <- function(R, N, alpha, gamma, tgt.state) {
```

خط زیر  $Q$  را در اندازه  $R$  به صورت یک ماتریس صفر، مقدار دهی اولیه می‌کند.

```
Q <- matrix(rep(0,length(R)), nrow=nrow(R))
```

حلقه زیر برای تکرار عمل بادگیری برای رسیدن به بهترین وضعیت است.

```
for (i in 1:N) {1
```

خط زیر برای هر تکرار جهت پیدا کردن بهترین مسیر، یک وضعیت اولیه را به صورت تصادفی انتخاب می‌کند.

```
cs <- sample(1:nrow(R), 1)
```

حلقه زیر تا زمانی که به وضعیت هدف رسیده تکرار خواهد شد.

```
while (1) {
```

براساس خطوط برنامه زیر عامل از وضعیت فعلی به وضعیت ممکن بعدی حرکت می‌کند، اگر تنها یک حالت باشد آن را انتخاب می‌کند و اگر چندین حالت باشد از بین آنها یکی را به صورت تصادفی انتخاب می‌کند.

```
next.states <- which(R[cs,] > -1)
```

```
if (length(next.states)==1)
```

```
ns <- next.states
```

```
else
```

```
ns <- sample(next.states,1)
```

خط برنامه زیر نیز وضعیت به روز شده عامل را ایجاد می‌کنند.

```
Q[cs,ns]<-Q[cs,ns]+alpha*(R[cs,ns]+gamma*max(Q[ns,which(R[ns,]>-1)])-Q[cs,ns])
```

شرط زیر در صورت برقراری که همان رسیدن عامل به وضعیت هدف است، حلقه while را به اتمام می‌رساند و در غیر این صورت برنامه ادامه دارد.

```
If (ns == tgt.state) break
```

```
cs <- ns
```

```
}
```

```
}
```

خط زیر نیز نتیجه نهایی تابع را برمی‌گرداند.

```
return(100*Q/max(Q))
```

```
}
```

## بخش دوم برنامه و اجرای الگوریتم بر اساس مقدار دهی اولیه پارامترها.

در خطوط برنامه زیر به پارامترها مقادیر اولیه را به صورت دستی می‌دهیم و سپس تابع `QlearningFunction()` را با پارامترهای مقداردهی شده اجرا می‌کنیم.

```
> N <- 1000
> alpha <- 1
> gamma <- 0.8
> tgt.state <- 6
> R <- matrix(c(-1,-1,-1,-1,0,-1,-1,-1,0,-1,0,-1,-1,0,-1,-1,0,0,-1,0,-1,0,-1,0,-1,0,-1,100,-1,-1,100,100),nrow=6)

> print(R)
 [,1] [,2] [,3] [,4] [,5] [,6]
[1,] -1   -1   -1   -1   0   -1
[2,] -1   -1   -1   0   -1   100
[3,] -1   -1   -1   0   -1   -1
[4,] -1   0   0   -1   0   -1
[5,]  0   -1   -1   0   -1   100
[6,] -1   0   -1   -1   0   100
>
>
> Q <- QlearningFunction(R,iter,alpha,gamma,tgt.state)
```

خروجی نهایی برنامه که همان ماتریس `Q` می‌باشد نیز با استفاده از تابع `print()` در خروجی چاپ می‌شود.

```
> print(Q)
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,]    0    0.00000    0.0       0     80    0.00000
[2,]    0    0.00000    0.0      64       0 100.00000
[3,]    0    0.00000    0.0      64       0    0.00000
[4,]    0  80.00000   51.2       0     80    0.00000
[5,]   64    0.00000    0.0      64       0 100.00000
[6,]    0  77.28683    0.0       0     80   96.60853
>
```



۱۱

فصل

## سیستم‌های توصیه‌گر

## ۱-۱۱- سیستم‌های توصیه‌گر<sup>۱</sup>

سیستم‌های توصیه‌گر ابزارهای نرم‌افزاری تهیه شده بر پایه الگوریتم‌های یادگیری ماشین می‌باشند که برای استفاده کاربران تهیه شده‌اند. وظیفه این سیستم‌ها ارائه پیشنهاداتی به کاربران برای انتخاب است که متناسب با نیازهای آنها و در راستای تصمیم‌گیری آنها باشد مانند پیشنهاد کالای خاصی منطبق بر نیازهای کاربر برای خرید، پیشنهاد موسیقی خاصی براساس سلیقه فرد یا اینکه چه نوع خبر برخطی<sup>۲</sup> براساس علايق افراد برای آنها ارسال شود.

## ۱-۱۲- تعریف آیتم<sup>۳</sup> در سیستم‌های توصیه‌گر

یک واژه کلی برای اشاره به چیزی است که سیستم توصیه‌گر به کاربران پیشنهاد می‌کند. یک سیستم توصیه‌گر عموماً بروی آیتم خاصی مانند موزیک یا خبر تمرکز دارد و از طریق یک رابط کاربری با کاربران تعامل می‌کند و در هسته و درون خود از الگوریتم‌های مختلف یادگیری ماشین استفاده می‌کند که در مجموع هدف آن ارائه بهترین پیشنهاد ممکن به کاربران در راستای یک هدف از پیش تعیین شده و خاص است.

## ۱-۱۳- تعریف کاربر<sup>۴</sup> در سیستم‌های توصیه‌گر

به اشخاص و نفرات استفاده کننده سیستم‌ها که توصیه‌گر برای آنها عمل توصیه را انجام می‌دهد اطلاق می‌شوند. به دلیل تنوع بسیار زیاد کالاهای خدمات مخصوصاً آنهاست که از طریق وب سایت‌ها ارائه می‌شوند برای افرادی که در تصمیم‌گیری خود برای انتخاب درست دچار تردید هستند سیستم توصیه‌گر می‌تواند به عنوان یک ابزار تصمیم‌یار نقش ایفا کند.

برای مثال یک فروشگاه کتاب اینترنتی را در نظر بگیرید که به کاربر پیشنهاد می‌دهد که چه کتابی را مطالعه کند. فروشگاه‌های معروف اینترنتی امروزه از این روش استفاده می‌کنند و براساس پروفایل افراد پیشنهادهایی را برای آنها سفارشی‌سازی می‌کنند که براساس آن افراد متفاوت با علايق و تمایلات متفاوت کتاب‌های مختص به خود را در سبد پیشنهادات خود ببینند و همچنین یک سری از پیشنهادهای عمومی که سفارشی‌سازی نشده هم برای همه ارائه می‌کنند مثلاً سایت‌های خبری مدام ۱۰ خبر پربازدید روز را برای تمامی بازدیدکننده‌های سایت توصیه می‌کنند، ولی توجه به نکته ضروری است که با توجه به کاربرد مفید آیتم‌های سفارشی‌سازی نشده یا آیتم‌های عمومی، این موضوع خیلی مورد توجه حوزه سیستم‌های توصیه‌گر نیست و سیستم‌های توصیه‌گر به دنبال پیداکردن بهترین آیتم مورد نیاز کاربران و توصیه به آنها هستند.

اگر به دنیای واقعی نگاهی بیاندازیم نه همه ولی بیشتر افراد در برخی موارد و موضوعات براساس پیشنهادهایی که می‌شوند انتخاب می‌کنند مثلاً وقتی افراد صحنه‌هایی از تعدادی از فیلم‌هایی که تبلیغ می‌شوند را می‌بینند ممکن است فرد یکی از آن فیلم‌ها را برای دیدن انتخاب کند و یا یک کارفرما براساس توصیه‌نامه همکاران دیگر کارمند جدید خود را انتخاب کند. در راستای شبیه‌سازی این وضعیت توصیف شده، الگوریتم‌هایی برای تولید

- 
1. Recommender systems
  2. online
  3. Item
  4. User

سیستم‌های توصیه‌کننده ایجاد شد که در این حالت آیتم‌هایی که توصیه می‌شوند براساس علایق مشترک افراد است مثلاً اگر یکی از افراد از گروه افراد مشابه از آیتمی استقبال کرده باشد سیستم آن آیتم را برای سایر افراد آن گروه توصیه خواهد کرد و حتی ممکن است یکی از کاربران فعال، یکی از کاربران قبلی و غیرفعال را مورد تأیید قرار دهد پس بر این اساس تمامی آیتم‌های مورد تأیید کاربر قلی توسط سیستم توصیه‌گر به کاربر فعال نیز پیشنهاد می‌گردد که در کل این تنها نمونه‌ای از عملکرد این سیستم است.

## ۱۱-۴- عملکردها و اهداف سیستم‌های توصیه‌گر

در اصل دلایل بسیاری وجود دارد که تهیه‌کنندگان سرویس‌ها همواره تمایل دارند تا از سیستم‌های توصیه‌گر استفاده کنند و براساس استراتژی‌های مختلف آنها این سیستم‌ها با کارایی متفاوتی تولید می‌شوند. انواع مختلف سیستم‌های توصیه‌گر از نظر هدف عبارتند از:

### ۱- بالا بردن میزان فروش آیتم‌ها

بدون شک این یکی از مهم‌ترین نوع در تجارت می‌باشد که تلاش می‌کند تا با پیشنهاد آیتم‌های مشابه میزان فروش را بالا ببرد.

### ۲- فروش بیشتر آیتم‌هایی با نوع متفاوت و گونه‌های مختلف

در برخی شرایط پیدا کردن بعضی از آیتم‌ها بدون دریافت پیشنهادات در عمل ممکن نیست چون کاربر اطلاعی از وجود آن آیتم یا کالا ندارد و با آگاهی از آن ممکن است علاقمند به استفاده از آن شود و همچنین تهیه کننده سیستم نیز آیتم‌هایی با تنوع گوناگون را به فروش برساند مانند پیشنهاد یک فیلم غیرمحبوب به یک کاربر خاص که علاقمند به دیدن آن فیلم باشد.

### ۳- بالا بردن رضایتمندی مشتری

طراحی و پیاده‌سازی یک الگوریتم مطلوب توصیه‌گر هم از نظر توصیه‌های مناسب و مفید و هم از نظر طراحی رابط کاربر پسند باعث بالا رفتن رضایتمندی کاربر شده و باعث بالا رفتن میزان استفاده فرد از وبسایت یا برنامه کاربردی می‌باشد.

### ۴- بالا بردن وفاداری کاربر

هر چه مراجعه و میزان وفاداری استفاده کاربر از سایت یا سیستمی که توصیه‌گر است بالاتر رود عملًا کاربر در بخش‌های مختلف سایت بیشتر فعالیت کرده است و الگوهای رفتاری و نیازهای او و علاقه‌مندی او بیشتر تشخیص داده می‌شوند و بر این اساس سیستم توصیه‌گر پیشنهادهای بهتری به او می‌دهد و همین به صورت فزاینده باعث بالاتر رفتن خوشنودی و وفاداری کاربر می‌شود.

### ۵- تشخیص بهتر نیازهای مشتری

هر چه این سیستم توسط جمع‌آوری اطلاعات صریح و یا براساس پیش‌بینی علاقه‌مندی‌های کاربر بتواند اطلاعات بیشتری به دست آورد هم می‌تواند در تشخیص نیازهای او بهتر عمل کند و هم می‌تواند از همین اطلاعات به صورت‌های دیگر مانند نحوه مدیریت آیتم‌های موجود در انبار و غیره بهره ببرد.

موارد اشاره شده در بالا تعدادی از اهداف سیستم‌های توصیه‌گر هستند که باعث می‌شوند تا ارائه‌دهنده‌گان سرویس‌ها تمایل زیادی به استفاده از آن در تجارت خود نشان دهند ولی این توصیه‌گرها باید برای هر دو طرف یعنی همانند سرویس‌دهنده‌ها برای کاربران نیز مفید باشند و نیازهای آنها را نیز به خوبی پوشش دهند تا بر این اساس بتوان تعادل مناسبی بین سرویس‌دهنده و سرویس‌گیرنده برقار کرد.

## ۱۱-۵-داده و منبع دانش

در اصل سیستم توصیه‌گر یک سیستم پردازش دانش می‌باشد که برای ساختن توصیه‌های خود به صورت فعال داده‌های مختلف و متنوعی را جمع‌آوری می‌کند که این داده‌ها در مورد آیتم‌های موجود و همچنین در مورد کاربران می‌باشد. به هر جهت صرف نظر از هدف اصلی سیستم، سیستم توصیه‌گر براساس منابع داده‌ای خود از تکنیک‌های متفاوتی استفاده می‌کند که به صورت کلی می‌توان آنها را به دو دسته تقسیم کرد:

۱. الگوریتم‌هایی که براساس دانش کم عمل می‌کنند مانند اطلاعاتی که از ارزیابی و امتیاز دهی کاربران به آیتم‌ها به دست می‌آید.
۲. الگوریتم‌هایی که واپسگی بیشتری به دانش در دست برای تصمیم‌گیری دارند مانند اطلاعاتی در رابطه با وضعیت اجتماعی و تمامی فعالیت‌های کاربر.

در هر کدام از شرایط گفته شده توصیه‌گر برای انجام کلاسه‌بندی‌های خود بر مبنای داده‌های به دست آمده با سه عامل سر و کار دارد که عبارت است از:

۱. آیتم، ۲. کاربر، ۳. تراکنش.

## ۱۱-۵-۱-تراکنش<sup>۱</sup>

هر تعامل یا نقل و انتقال داده بین کاربر و سیستم توصیه‌گر را یک تراکنش گویند که در عمل از چگونگی این تراکنش‌ها داده‌هایی تولید شده و ثبت می‌شوند. به عنوان مثال از اینکه کاربر کدام آیتم را در کدام بخش و چند بار بازدید کرده و یا اینکه کاربر به کدام آیتم چه امتیازی داده و چگونه ارزیابی کرده است می‌تواند به عنوان داده‌های تراکنشی دخیره گردد.

## ۱۱-۵-۲-امتیاز ارزیابی

این فرایند که نشان‌دهنده سطح رضایت کاربر از آیتم مورد نظر است یکی از موارد پراستفاده از مزیت داده‌های تراکنشی در سیستم‌های توصیه‌گر می‌باشد. تعدادی از روش‌های امتیازدهی در سیستم توصیه‌گر به صورت زیر می‌باشد.

۱. امتیازدهی و یا نظرسنجی توسط پنج ستاره که از یک تا پنج شمرده می‌شوند.
۲. نظرسنجی موافق، بینظر و مخالف.
۳. مدل باینری که نشان‌دهنده خوب یا بد است.

## ۶-۱۱- روش‌های توصیه‌گری

برای ایجاد توابع درونی برای توصیه‌گر باید روشی را پیاده‌سازی کرد که بتواند آیتم‌های مفید را برای کاربر شناسایی کند و به عبارتی سیستم توصیه‌گر باید تشخیص دهد که آیتم مورد نظر ارزش توصیه شدن به کاربر مورد نظر را دارد یا خیر. برای این منظور الگوریتم موجود باید بتواند؛ پیش‌بینی این را که چه آیتمی می‌تواند برای کاربر مفید باشد؛ را انجام دهد. یا حداقل بتواند تعدادی از آیتم‌ها را مقایسه کرده و تفاوت‌ها و شباهت‌های آنها را بهفهمد و برمبنای این موارد آیتم موردنظر را توصیه کند.

## ۶-۱۲- انواع رویکردهای اصلی پیاده‌سازی سیستم توصیه‌گر

رویکردهای مختلفی برای پیاده‌سازی الگوریتم توصیه‌گر وجود دارد که ما به شش مورد از آن در ادامه اشاره خواهیم کرد.

### ۶-۱۲-۱- بر مبنای محتوا<sup>۱</sup>

در این روش سیستم یاد می‌گیرد که آیتمی را توصیه کند که شبیه به همان آیتمی است که کاربر قبلًا به آن در ارزیابی امتیاز خوب را اختصاص داده است و همچنین شباهت آیتم‌ها با هم براساس مقایسه از روی ویژگی‌های موجود در آیتم‌ها انجام می‌شوند. به عنوان مثال اگر یک کاربر یک فیلم با سبک کمدی را در ارزیابی به آن امتیاز خیلی خوب داده است سیستم توصیه‌گر به عنوان توصیه به آن کاربر یک فیلم کمدی دیگر را نشان می‌دهد.

### ۶-۱۲-۲- فیلتر مشترک<sup>۲</sup>

حالت ساده این سیستم به این صورت پیاده‌سازی می‌شود که با توجه به در نظر گرفتن علاویق مشترک کاربر قبلی با کاربر جاری این توصیه انجام می‌شوند. به این ترتیب که اگر کاربر شماره ۱ با کاربر شماره ۲ علاویق مشترک داشته باشد و اگر کاربر شماره ۱ به آیتم‌هایی امتیاز بالا داده باشد و توجه بیشتر نشان داده باشد پس بر این اساس به کاربر شماره ۲ که در اصل کاربر جاری می‌باشد همان آیتم‌هایی پیشنهاد می‌شوند که کاربر شماره ۱ به آنها توجه نشان داده است.

این روش با عنوان همبستگی مردم به مردم<sup>۳</sup> نیز معروف است و یکی از متدائل ترین روش‌ها در سال‌های گذشته در پیاده‌سازی سیستم‌های توصیه‌گر است که برای پیاده‌سازی آن از روش‌هایی مانند نزدیک‌ترین همسایه<sup>۴</sup> و غیره می‌توان استفاده کرد.

### ۶-۱۲-۳- جمعیتی<sup>۵</sup>

این سیستم بر مبنای اطلاعات جمعیتی پایه‌ریزی شده و به عبارتی براساس مشخصات پروفایل گروه سنی خاصی و یا مشخصات جمعیتی خاص مانند زبان و یا رسومات زندگی آنها توصیه خود را انجام می‌دهد.

- 1. Content-based
- 2. Collaborative-filtering
- 3. People to people correlation
- 4. Nearest neighbors
- 5. Demographic

## ۱۱-۷-۴- بر مبنای دانش<sup>۱</sup>

سیستم‌های برمبنای دانش آیتم‌هایی را توصیه می‌کند که براساس دامنه خاصی از دانش که از طریق تحلیل و مقایسه ویژگی‌های آیتم با ویژگی نیازمندی کاربر به دست آمده توصیه خود را انجام می‌دهد و ملاک اصلی آن این است که آیا این آیتم نیاز مشتری را می‌تواند پاسخ دهد یا خیر.

## ۱۱-۷-۵- مبتنی بر جامعه<sup>۲</sup>

در این روش توصیه‌ها برمبنای پیشنهادات و معرفی‌های دوستان کاربر انجام می‌شوند این تکنیک برمبنای این مثال معروف است که می‌گوید "به من بگو دوستت کیست تا به تو بگوییم تو چگونه فردی هستی". در اصل شواهد نشان داده که مردم وقتی دوستانشان آیتمی را پیشنهاد داده است برای انتخاب آن ایتم استقبال بیشتری از خود نشان می‌دهند تا اینکه همان آیتم را فرد ناشناسی به آنها توصیه کند.

## ۱۱-۷-۶- توصیه‌گر ترکیبی<sup>۳</sup>

این سیستم به صورت ترکیبی از روش‌های بالا عمل می‌کند و از نقاط قوت هر روش برای پوشاندن ضعف دیگری استفاده می‌کند مثلاً اگر روش یک را با روش دوم برای بالا بردن کارایی سیستم تلفیق کرده باشد تلاش می‌کند تا ضعف‌های روش دوم را با قوت روش اول پوشاند.

## ۱۱-۷- کاربردهای سیستم توصیه‌گر

هرچند که روش‌های سیستم توصیه‌گر به صورت علمی و در تئوری هنوز در مرحله تحقیق و بررسی می‌باشد، ولی این سیستم‌ها در حال حاضر به صورت کاربردی در حوزه‌های مختلفی مانند تجارت الکترونیکی، نشر محتوای اطلاعاتی، تفریح و سرگرمی و همچنین در بخش سرویس و خدمات مورد استفاده قرار می‌گیرند.

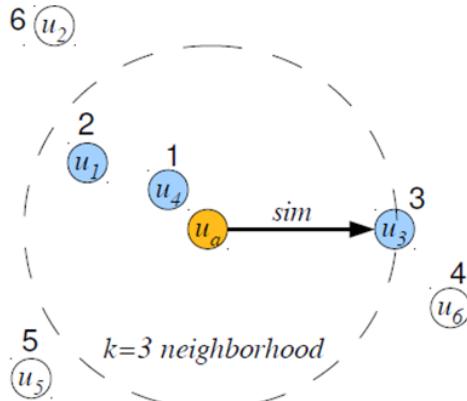
## ۱۱-۸- روابط ریاضی و پیاده‌سازی سیستم توصیه‌گر

امروزه روش‌های مختلف و پیچیده‌ای برای پیاده‌سازی سیستم توصیه‌گر وجود دارند که در اینجا ما برای سادگی و درک راحت‌تر عملکرد این سیستم، به شما یک روش پایه‌ای را معرفی می‌کنیم. در این مسئله هدف پیدا کردن آیتم‌های مورد پسند چند کاربر مشترک است. همان‌طور که در بخش‌های قبلی اشاره کردہ‌ایم از طریق روش‌های ارزیابی علایق کاربران مانند سیستم امتیازدهی به آیتم‌ها، میزان علایق افراد را ارزیابی کنیم و سپس آیتم‌هایی که کاربری به آن علاقه نشان داده را به کاربر دیگر با علایق مشترک در همان دسته پیشنهاد می‌دهیم.

در این روش ابتدا براساس الگوریتم نزدیک‌ترین همسایگی<sup>۴</sup> کاربرانی با خصوصیات مشترک، مشخص می‌شوند و در یک دسته قرار می‌گیرند. برای این کار یکی از کاربران را می‌توان به عنوان کاربر مرکزی در نظر گرفت و سایر

- 
1. Knowledge base
  2. Community base
  3. Hybrid recommender system
  4. Neighborhood

کاربران را که فاصله آنها از یک حد آستانه‌ای از کاربر مرکزی فراتر نرفته است را در دسته موردنظر قرار داد. در شکل (۱۱) کاربر مرکزی را با  $u_a$  نشان داده که در مرکز دسته قرار دارد و  $u_3$  دورترین کاربر است که در دسته مشترک جای گرفته و کاربران دورتر از آن در دسته مربوطه شرکت نداده‌اند.



شکل (۱۱): نحوه آرایش کاربران بر اساس همسایگی

برای به دست آوردن فاصله‌ای که بتوانیم نزدیکی یا دوری علایق کاربران از هم را تشخوص دهیم از دو روش معروف پیرسون<sup>۱</sup> و یا شباهت کسینوسی<sup>۲</sup> می‌توان استفاده کرد که در این قسمت از روش پیرسون استفاده می‌کنیم.

## ۱۰-۱۱- محاسبه شباهت با رابطه پیرسون در فیلتر مشترک

فاصله شباهت<sup>۳</sup> در روش فیلتر مشترک باید محاسبه شود تا مشخص شود آیا درجه شباهت دو کاربر یا آیتم‌های مورد علاقه دو کاربر چیست. رابطه پیرسون یکی از محبوب‌ترین رابطه‌ها برای محاسبه فاصله شباهت است که به صورت زیر می‌باشد.

$$p_{similarity}(a, b) = \frac{\sum_j(p_{aj} - V_a)(p_{bj} - V_b)}{\sqrt{\sum_j(p_{aj} - V_a)^2} \sqrt{\sum_j(p_{bj} - V_b)^2}} \quad \text{رابطه (۱)}$$

حروف a و b دو کاربری هستند که می‌خواهیم شباهت آنها به دست آید،  $p_{aj}$  به علایق جاری کاربر a به آیتم  $j$  اشاره دارد و  $p_{bj}$  به علایق جاری کاربر b به آیتم  $j$  اشاره دارد.  $V_a$  میانگین علایق جاری کاربر a و  $V_b$  میانگین علایق جاری کاربر b است.

ضریب همبستگی پیرسون به تفاوت‌های میانگین علایق کاربران توجه دارد که بر این اساس میانگین علایق را از علایق جاری کم می‌کند و با تقسیم بر انحراف‌معیار، اختلاف مقادیر امتیازدهی‌های کاربران را به دست می‌آورد.

1. Pearson correlation coefficient

2. Cosine similarity

3. Similarity distance

و در ادامه می‌توانیم با به دست آوردن اختلاف علایق کاربران مختلف، کاربرانی که اختلاف کمتری با هم دارند را در یک دسته قرار دهیم.

## ۱۱-۱۱- پیاده‌سازی فیلتر مشترک با کدهای R

در اینجا از مجموعه داده Jester5k استفاده کردیم که اطلاعات آیتم‌های زیادی را و همچنین امتیازاتی که به آن آیتم‌ها داده شده را در خود جای داده و بر این اساس هدف ما این است که محبوب‌ترین پنج آیتم را استخراج کنیم این محبوبیت براساس امتیازات کاربران بوده است.

ابتدا بسته نرم‌افزاری recommenderlab و مجموعه داده Jester5k را بارگذاری می‌کنیم این بسته از هر دو روش Cosine و Pearson برای ارزیابی می‌تواند استفاده کند.

```
>library("recommenderlab")
>data(Jester5k)
```

از طریق دستور زیر هزار ردیف از رکوردهای کاربران را برای استفاده انتخاب کرده و نوع روش توصیه را به صورت محبوبیت انتخاب می‌کنیم

```
> r <- Recommender(Jester5k[1:1000], method="POPULAR")
```

در ادامه از لیست انتخاب‌های کاربر ۱۰۰۱ امین پنج انتخاب برتر را انتخاب می‌کنیم.

```
> recom<-predict(r, Jester5k[1001], n=5)
> as(recom, "matrix")
```

و خروجی اجرای کد به صورت زیر خواهد بود.

	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18	j19	j20	j21	j22	j23	j24	j25	j26
u20089	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
	j27	j28	j29	j30	j31	j32	j33	j34	j35	j36	j37	j38	j39	j40	j41	j42	j43	j44	j45	j46	j47	j48				
u20089	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	-0.7235266	NA				
	j49	j50	j51	j52	j53	j54	j55	j56	j57	j58	j59	j60	j61	j62	j63	j64	j65	j66	j67	j68	j69	j70	j71			
u20089	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
	j72	j73	j74	j75		j76	j77	j78	j79	j80	j81	j82	j83	j84	j85	j86	j87	j88		j89	j90					
u20089	0.8171622	NA	NA	NA	-0.01459394	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
	j91	j92		j93	j94	j95	j96	j97	j98	j99	j100															
u20089	NA	NA	-0.01052075	NA	NA	NA	NA	NA	NA	NA	NA															

اما اگر توجه کنید این خروجی از ۱۰۰۱ تا ۱۰۰۱ است و فقط در j72, j76, j89, j93 و j93 مقدار برگردانده است.

وضعیت امتیازدهی پنج آیتم بالا را با دستور زیر بررسی می‌کنیم.

```
> rating <- predict(r, Jester5k[1001], type="ratings")
> as(rating, "matrix")[,c("j89", "j72", "j47", "j93", "j76")]
```

خروجی نهایی برنامه به صورت زیر خواهد بود که نشان‌دهنده ۵ لیست برتر از نظر محبوبیت آیتم‌ها است.

j89	j72	j47	j93	j76
2.6476613	2.1273894	0.5867006	1.2997065	1.2956333

۱۲

فصل

## داده‌های جریانی

## ۱-۱۲- داده‌های جریانی

امروزه مدل کردن داده‌های با حجم بالا و متغیر در زمان با توجه به محدودیت‌های زمانی و محاسباتی نیازمند الگوریتم‌های یادگیری هستند که با برخورد با داده و با رعایت محدودیت حافظه بصورت بلاذرنگ عمل کنند. اگر الگوریتم‌هایی که با داده‌های پویا و نامحدود کار می‌کنند را الگوریتم‌های یادگیری داده‌های جریانی<sup>۱</sup> می‌نامند. اگر منبع تولید داده، داده‌ای متنی تولید کند؛ به دلیل وقوع بیشتر رانش مفهوم<sup>۲</sup> متن و پردازش زبان‌های طبیعی<sup>۳</sup> فرایند پیچیده‌تر می‌شود. رانش مفهوم به تغییر الگوی توزیع داده‌ها در طول زمان گفته می‌شود. الگوریتم‌های داده کاوی و یادگیری ماشین سنتی تمرکز خود را بر روی داده‌های ایستا، هم‌جنس، پایدار و مدل‌های ایستا متمرکز کرده‌اند. روش‌های سنتی یادگیری ماشین معمولاً بر روی مخزن‌های داده ساکن و ایستا متمرکز است ولی امروزه روش‌های مبتنی بر داده‌های جریانی، فرایند ذخیره و پردازش داده را تغییر داده‌اند. مسئله مهم امروزه این است که چگونه می‌توان داده نامحدودی که سریع تولید می‌شود و در زمان متغیر است را با وجود محدودیت در زمان و منابع محاسباتی مدل کرد. این مجموعه داده‌ها بسیار بزرگ می‌باشند و باید روی حافظه‌ای جانبی ذخیره شوند پس دسترسی به آنها پرهزینه است. هدف کاوش داده‌های جریانی، ساختن یک فرایند یادگیری است که به صورت خطی با افزایش تعداد نمونه‌ها رشد کند. شاید مدلی که از روی داده‌های قدیمی ساخته شده روی داده‌های جدید مفید نباشد چون آموزش داده‌های جدید ایده‌ای ناکارآمد است. همچنین داده‌های جریانی با شاخه‌هایی نظری حجم نامتناهی و تغییرات پویا شناخته می‌شوند و محدودیت‌های الگوریتم‌های داده‌های جریانی به چهار دسته عمده زیر تقسیم می‌شوند.

## ۲-۱۲- محدودیت‌های داده‌های جریانی

۱. یک بار مشاهده داده: الگوریتم‌ها در داده‌های جریانی فقط یک بار می‌توانند داده‌ای را مشاهده کنند چون خواندن یا نوشتمن روی حافظه جانبی پرهزینه است.
۲. پاسخ بلاذرنگ: میزان زمان پردازش داده‌ها و رسیدن به تصمیم باید کم باشد.
۳. محدودیت حافظه: حجم داده‌ها بسیار بزرگ و نامحدود است تنها تعداد کم یا خلاصه‌ای از داده‌های جریانی را می‌توان ذخیره کرد و به اصل داده دسترسی نداریم در نهایت روش‌های تخمینی هم کاربرد بالایی دارند.
۴. تشخیص رانش مفهوم: رانش مفهوم زمانی است که الگوها در طی زمان تغییر می‌کنند و پردازش داده‌هایی مانند داده‌های متنی یه دلیل غیرساخت یافته بودن، خطاهای سطح بالا و قالب‌های متنوع و پیچیده‌تری را دارند و بر این اساس در کار با آنها به راحتی با رانش مفهوم رو به رو می‌شویم که تشخیص و تحلیل مناسب رانش مفهوم یکی از موضوعات اجتناب‌ناپذیر است. برای داده‌های جریانی به تعاریف و مفاهیم زیر توجه فرمایید.

- 
1. Streaming data's
  2. Concept drift
  3. Natural language processing

## ۱۲-۳-۱- تعاریف داده‌های جریانی

### ۱۲-۳-۱-۱- تعریف جریان داده

یک جریان داده به شکل دنباله‌ای از نمونه‌های داده تعریف می‌شود و آن را با نماد DS نشان می‌دهند.

### ۱۲-۳-۱-۲- تعریف مفهوم

یک مفهوم یا یک منبع داده، توسط احتمال پیشین برای رده‌های تابع توزیع احتمال رده‌ی آنها تعریف می‌شود. روش‌های سنتی داده‌کاوی با یک مفهوم کار می‌کنند یعنی تابع توزیع احتمال مجموعه آموزش و مجموعه آزمون یکسان است در حالی که داده‌های جریانی پویا هستند و تعداد زیادی مفهوم دارند.

### ۱۲-۳-۱-۳- تعریف رانش مفهوم

اگر فرض کنیم یک منبع داده که آن را با DS نشان می‌دهیم داشته باشیم که شامل k منبع جریانی مستقل S و با وزن w وجود داشته باشد اگر در دو زمان  $t_1$ ,  $t_2$  رابطه (۱) برقرار باشد می‌گوییم رانش مفهوم رخ داده است

$$\sum^{DS}(t_1) \neq \sum^{DS}(t_2) \quad \text{رابطه (۱)}$$

هر نوع تغییری در منبع جریان، یک رانش مفهوم به حساب می‌آید بنابراین دو نوع رانش مفهوم داریم.

### ۱۲-۳-۱-۴- انواع رانش مفهوم

۱. رانش مفهوم مجازی: هنگامی است که رانش مفهوم تأثیر مستقیمی بر حدود تصمیم‌گیری نداشته باشد ولی بروی تابع چگالی احتمال اثر بگذارد.
۲. رانش مفهوم حقیقی: هنگامی است که رانش مفهوم هم تأثیر مستقیمی بر حدود تصمیم‌گیری دارد و هم بروی تابع چگالی احتمال اثر می‌گذارد.

### ۱۲-۳-۱-۵- تعریف پنجره زمانی

تنها می‌توانیم بخشی از داده‌های جریانی را پردازش کنیم این بخش مورد توجه از داده، توسط یک پنجره زمانی از نمونه‌های داده تعریف می‌شود که انواع مختلفی دارد.

### ۱۲-۳-۱-۶- انواع پنجره زمانی

۱. پنجره نقطه عطفی: استفاده از این پنجره یعنی تمامی تراکنش‌ها به اندازه یکسان مهم هستند و داده‌های گذشته و حال هیچ تفاوتی ندارند.
۲. پنجره کشویی: ما تنها به W تراکنش اخیر توجه می‌کنیم و بقیه داده‌های جریانی نادیده گرفته می‌شوند. اگر W خیلی بزرگ باشد رانش مفهوم رخ داده و اگر W کوچک باشد پنجره شامل داده‌های ناکارآمد است.
۳. پنجره محوشونده: هر نمونه از داده‌های جریانی را با یک وزن که با زمان رسیدن آن نمونه ارتباط دارد شناسایی می‌کنند. این پنجره تأثیر داده‌های قدیمی منقضی شده را کاهش می‌دهد.
۴. پنجره یکبر: در سطوح مختلف ریزدانگی که با توجه به تأخیر داده شکل گرفته‌اند اعمال می‌شود و تقریباً تمام مجموعه داده را ذخیره کرده و یک تعادل بین فضای مورد نیاز برای ذخیره‌سازی و دقت آن برقرار می‌کند.

## ۱۲-۴- رویکردهای محاسباتی در داده‌های جریانی

۱. یادگیری افزایشی: در این روش یادگیری مدل به صورت افزایشی تغییر می‌کند تا با تغییرات در داده‌هایی که در حال آمدن هستند سازگار شود که برای این حالت نیز دو نوع بهروزرسانی وجود دارد. (الف) بهروزرسانی با هر نمونه داده. (ب) بهروزرسانی با پنجره.
۲. یادگیری دوگامی: کاوش بر داده‌ها بر دو بخش تقسیم می‌شود. (الف) چکیده‌های از داده‌ها در لحظه ایجاد می‌شود. (ب) پردازش موردنظر روی چکیده‌ها انجام می‌شود که در این روش حالت پردازش با سرعت بیشتری انجام می‌شود.

## ۱۲-۵- اعتبارسنجی در داده‌های جریانی

در محیط جریان‌های داده‌ای، از آن جایی که داده می‌تواند نامحدود باشد، اعتبارسنجی بر ارزیابی مدل در صحنه‌های مختلف متمرکز است. یک روش شناخته شده رسم منحنی یادگیری با ذخیره‌سازی کارایی مدل در طول زمان است. این منحنی نشان خواهد داد که چقدر مدل پس از مشاهده داده‌های بیشتر، بهتر شده است و مدل چگونه با رانش مفهوم سازگار می‌شود. یک الگوریتم برتری خود را به سایر روش‌ها زمانی نشان می‌دهد که منحنی یادگیریش بیشتر موقع از سایر الگوریتم‌ها بالاتر باشد.

## ۱۲-۵-۱- روش‌های اعتبارسنجی در داده‌های جریانی

روش جداسازی<sup>۱</sup> و روش بی‌درپی<sup>۲</sup> دو روش پرکاربرد برای اعتبارسنجی جریان‌های داده‌ای هستند. در روش جداسازی، نمونه‌های داده به قطعه‌های مختلفی تقسیم می‌شوند از هر قطعه داده ابتدا برای آزمون و سپس برای بهروزرسانی مدل استفاده می‌شود. روش جداسازی زمانی که رانش مفهوم رخ داده، ترجیح داده می‌شود چون این روش اجازه می‌دهد که مدل با آخرین تغییرات داده سازگار شود. روش پی‌درپی یا آزمون-پس-آموزش، یک روش دیگر برای ارزیابی جریان‌های داده‌ای است. هر نمونه داده ابتدا و پیش از آنکه برای آموزش به صورت افزایشی استفاده شود، برای آزمون استفاده می‌شود. این روش می‌تواند یک حالت خاص برای جداسازی در نظر گرفته شود اگر اندازه قطعه برابر با یک باشد. مزیت این روش این است که نیازی به تعریف از پیش اندازه قطعه نیست ولی متأسفانه کارایی این الگوریتم در زمان محدود می‌باشد، زیرا اشتباهات اولیه مدل به سرعت در طول زمان کاهش پیدا می‌کند.

## ۱۲-۶- روش‌های یادگیری در داده‌های جریانی

فرایند یافتن یک مدل عمومی مناسب برای داده‌های گذشته، به طوری که بتوان آن را روی داده‌های جدید اعمال کرد، دسته‌بندی می‌نامند. در ادبیات داده‌های جریانی تعداد زیادی الگوریتم دسته‌بندی مانند درخت تصمیم، دسته‌بندی بیزین، ماشین‌های بردار پشتیبان، نزدیک‌ترین همسایگی و روش‌هایی که مجموعی از دسته‌بندها را می‌سازند، وجود دارد. در ادامه مطالب تعدادی از این دسته‌بندها را معرفی می‌کنیم.

## ۱-۶-۱۲- درخت هافدینگ

درخت هافدینگ<sup>۱</sup> یک دسته‌بندی برمنای درخت تصمیم برای داده‌های جریانی است. روش‌های سنتی درخت تصمیم، برای انتخاب خصیصه تقسیم نیاز به چندین بار پویش دارند که این موضوع در محیط داده‌های جریانی، عملأً نشدنی است.

سایر روش‌های دسته‌بندی داده‌های جریانی نیز معمولاً کاستی‌هایی مانند موارد زیر را دارند.

۱. حساسیت زیاد به ترتیب نمونه‌ها
۲. زمانی که تعداد نمونه کافی برای ساخت درخت پویش شود، نتیجه درخت در روش هافدینگ تقریباً مشابه با درخت‌های ساخته شده با روش‌های مرسوم یادگیری دسته‌ای است.
۳. عدم قطعیت در زمان یادگیری در درخت هافدینگ برای هر نمونه ثابت است و این بدان معنی است که درخت هافدینگ برای کاوش در داده‌های جریانی مناسب است.

به صورت کلی می‌توان گفت که الگوریتم درخت هافدینگ، یک الگوریتم با دقت بالاست که می‌تواند بسیار خوب با مجموعه داده‌های بزرگ کار کند، ولی این الگوریتم نمی‌تواند با مسئله رانش مفهوم در داده‌های جریانی تعامل کند. البته الگوریتم هافدینگ ارتقاء یافته‌ای نیز وجود دارد که برای شرایط رانش مفهوم سازگار شده است.

## ۲-۶-۱۲- دسته‌بند بیزین در داده‌های جریانی

سیدل<sup>۲</sup> و همکارانش یک روش جالب بر پایه نمایه‌سازی<sup>۳</sup> دسته‌بند ارائه داده‌اند که درخت بیز نامیده می‌شود. درخت بیز از یک درخت مخلوط-گوسین سلسه مراتبی<sup>۴</sup> جهت نمایان کردن تمامی مجموعه داده استفاده می‌کند. هر گره از درخت شامل آماره‌های از نمونه‌های داده مثل مستطیل محدود کننده کمینه<sup>۵</sup>، تعداد نمونه‌های داده، جمع خطی و مجموع مربعات تمام داده است. برای حل مسئله دسته‌بندی چند کلاسه یک درخت بیز برای هر کلاس ساخته می‌شود. برای هر داده‌ای آزمون مانند<sup>۶</sup>، الگوریتم سعی می‌کند که نزدیک‌ترین مجموعه گره‌ها را که مرز نامیده می‌شوند، پیدا کند. درخت بیز یک دسته‌بند است و پس از مدت کمی که شروع به کار کرد می‌تواند نتیجه و تصمیم داشته باشد و سپس دقت عمل کردن را با انتخاب مزهای دقیق‌تر افزایش دهد.

## ۳-۶-۱۲- شبکه عصبی در داده‌های جریانی

روش شبکه‌های عصبی داده دانه تغییرپذیر<sup>۷</sup> یک روش دسته‌بندی است که دارای دو مرحله است. در مرحله اول از نرون‌های T-S برای ساختن ریزدانه‌های اطلاعات برای داده‌هایی که در حال آمدن هستند استفاده می‌کند. در گام دوم شبکه عصبی روی این ریزدانه‌های اطلاعاتی به جای داده اصلی ساخته می‌شود. یک ریزدانه مرتبط با برچسب یک کلاس به شکل یک تابع عضویت سه گوش<sup>۸</sup> تعریف می‌شود که بعداً جهت تطبیق با داده‌های جدید

- 
1. Hofding tree
  2. Seidl
  3. Index based
  4. Hierarchical Gaussian mixture
  5. Minimum rectangle bounding
  6. Evolving granular neural network
  7. Triangular membership function

تغییر می‌کند. وزن‌ها با یک ثابت زوال کاهش می‌یابند، این پردازش کمک می‌کند که درجه اهمیت ریزدانه‌های منقضی شده کاهش پیدا کند. زمانی که داده‌های آزمون می‌آیند، یک نرون بیشینه‌گیری بهترین پیش‌بینی ریزدانگی<sup>۱</sup> را انتخاب می‌کند و برچسب آن را به نمونه، جهت پیش‌بینی کلاسش، اختصاص می‌دهد. وزن‌ها با یک ثابت زوال<sup>۱</sup> کاهش پیدا می‌کند تا درجه اهمیت ریزدانه‌های منقضی کاهش یابد. با ورود داده‌های آزمون یک نرون بهترین پیش‌بینی ریزدانگی را انتخاب و برچسب کلاسش را به آن اختصاص می‌دهد. در مسائل دسته‌بندی در محیط‌هایی که دائمًا تغییر می‌کند استفاده می‌شود، ولی نیاز به زمان آموزش طولانی دارد و این محدودیت است.

#### ۴-۶-۴- دسته‌بند نزدیک ترین همسایه

یک روش خوش‌بندی به نام کلواستریم<sup>۲</sup> وجود دارد که برای ریز-خوشه استوار است. یک ریز-خوشه برای تعدادی داده، شامل آماره‌هایی نظیر مجموع مربعات نقاط، مجموع نقاط، مجموع مربعات زمان رخداد نقاط، مجموع خطی زمان رخداد و تعداد نمونه‌ها است. این روش بجای ذخیره کل داده‌های خوشه‌ها، در گام آنلاین که ریز-خوشه‌های آنها حجم کمتر اشغال می‌کنند و پردازش آنها آسان‌تر است را ساخته و ذخیره می‌کند و در گام آنلاین بجای خوش‌بندی کل داده‌ها، این ریز-خوشه‌ها را خوش‌بندی می‌کند.

#### ۴-۶-۵- درخت سازگار تصمیم مبتنی بر رویکرد یکی در برابر بقیه

یک درخت تصمیم به عنوان ابزاری برای به تصویر کشیدن و تحلیل تصمیم، به صورتی که مقادیر مورد انتظار از روابط‌ها به صورت متناوب محاسبه می‌شود، استفاده می‌گردد. در روش یکی در برابر همه مجموع  $k$  دسته‌بندی باقی را یاد می‌گیرد هر دسته‌بند آموزش داده شده است تا تشخیص دهد یک نمونه به یک کلاس خاص متعلق است یا به باقی کلاس‌ها. برای دسته‌بندی یک داده جدید همه دسته‌بندها اجرا می‌شوند و دسته‌بندی که مطمئن‌ترین نتیجه را داد به عنوان برچسب نمونه جدید انتخاب می‌شود.

#### ۴-۶-۶- انتخاب پویای ویژگی در داده‌های جریانی

در داده‌های جریانی با ابعاد ویژگی‌های بالا که داده‌های متنه هم جزء آنها است، تمام ویژگی‌ها در فرایند تحلیل با اهمیت می‌باشند. سه نوع مختلف ویژگی وجود دارد: ۱- ویژگی‌های غیرمرتب ۲- ویژگی‌های مرتب اما تکراری ۳- ویژگی‌های مرتب و غیرتکراری.

وظیفه اصلی موضوع انتخاب ویژگی، استخراج مجموعه ویژگی‌های مرتب و غیرتکراری از ویژگی‌هایی است که فرایند یادگیری را با معنی‌تر و سریع‌تر کند.

#### ۴-۶-۷- مدل فیلتر در انتخاب ویژگی‌های داده‌های جریانی

این مدل به عنوان یک معیار مستقل برای ارزیابی مجموعه ویژگی‌ها به کار برده می‌شود، بنابراین این روش فقط به مشخصه‌های عمومی داده انتکا می‌کند.

- 
1. Decay
  2. Clustream
  3. Micro-cluster

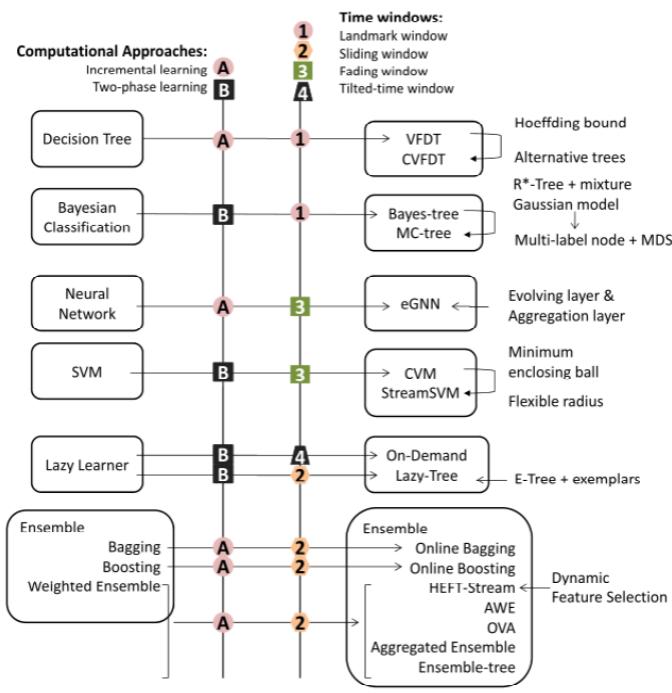
## ۳-۶-۶-۱۲- مدل پوششی انتخاب ویژگی‌های داده‌های جریانی

این مدل به همراه الگوریتم‌های یادگیری اجرا می‌شود، از کارایی الگوریتم‌های یادگیری برای ارزیابی ویژگی‌ها استفاده می‌کند و در مدل‌های ترکیبی از مزیت دو مدل گفته شده استفاده می‌کند.

اهمیت ویژگی‌ها در داده‌های جریانی تغییر می‌کند و محدود بودن به یک بازه زمان، مشخص است. ویژگی‌هایی که پیش از این با ارزش فرض می‌شد ممکن است نامریوط شوند و بر عکس ممکن است ویژگی‌ای استفاده نشده در آینده با اهمیت شوند. بنابراین استفاده از روش‌های پویای انتخاب ویژگی برای رصد کردن تغییرات ویژگی‌ها ضروری است.

## ۳-۶-۷- کاوش در جریان‌های متنی

روش‌های داده‌های جریانی با ابعاد بالا می‌توانند برای داده‌های متنی استفاده شوند، چرا که معمولاً این داده‌ها غیرساخت یافته، شامل خطاهای سطح بالا و در فرمت‌های گوناگون هستند. به علاوه در داده‌های متنی بیشترین تغییر موضوع در زمان رخ می‌دهد. در شکل (۱-۱۲) ارتباط بین روش‌های سنتی در سمت چپ و دسته‌بندی در راست داده‌های جریانی نشان داده می‌شود. دسته‌بندی‌های سنتی در سمت چپ و دسته‌بندی‌های جریانی در سمت راست است. می‌توان دید که دسته‌بندی‌های جریانی از دسته‌بندی‌های سنتی برگرفته شده‌اند با این تفاوت که از رویکردهای محاسباتی مختلف و پنجره‌های زمانی متفاوتی استفاده می‌کنند.



Traditional Classification

Data Stream Classification

شکل (۱-۱۲): روش‌های سنتی کلاسه‌بندی با روش‌های یادگیری مبتنی بر داده‌های جریانی.

به عنوان نمونه، درخت تصمیم خیلی سریع<sup>۱</sup> (VFDT) یک گسترش از درخت‌های تصمیم برای داده‌های جریانی است زمانی که به حد کافی داده موجود باشد. درخت تصمیم خیلی سریع از حد هادفینگ برای ساختن گره‌های درخت استفاده می‌کند. در واقع این دنباله‌ای از رویکرد یادگیری افزایشی و پنجره نقطه عطفی است. همچنین درخت تصمیم خیلی سریع سازگار شونده<sup>۲</sup> (CVFDT) یک نسخه بهبود یافته از درخت تصمیم خیلی سریع است که می‌تواند به وسیله ساختن درخت‌های جایگزین با رانش مفهوم سازگار شود.

به طور خلاصه در جدول (۱-۱۲) ظرفیت‌های دسته‌بندی‌های مختلف روی داده‌های جریانی به شکل مقایسه‌ای آورده شده است. همان‌طور که مشهود است بسیاری از دسته‌بندی‌های مطرح شده، عموماً نمی‌توانند در آن واحد در تمام محدودیت‌های جریانی بگنجند. همان‌طور که دور از انتظار نیست، تمام روش‌های دسته‌بندی داده‌های جریانی، با محدودیت حافظه مشکل چندانی ندارند و هنگامی که یک دسته‌بند از رویکرد محاسباتی از پنجه زمانی استفاده می‌کند بعضی مزایا و معایب و محدودیت‌ها خودشان را نشان می‌دهند.

جدول (۱-۱۲): الگوریتم‌های مختلف و نحوه عملکرد آنها.

محدودیت	مزیت	روش
بیش‌برازش هنگامی که عمق درخت زیاد باشد.	فهم آسان. مقاوم در برابر خطأ. هزینه محاسباتی کم حتی با داده آموزش زیاد. توانایی حذف ویژگی‌های تکراری.	درخت تصمیم
پیش‌فرض استقلال که در دنیای واقعی معمولاً نقض می‌شود.	ساده، کارا، بهینه و مقاوم در برابر خطأ.	بیزین
عدم توانایی تفسیر مدل یادگرفته شده. پیچیدگی بالا.	به راحتی عمومی می‌شود. می‌تواند مسائل پویا و غیرخطی را حل کند.	شبکه عصبی
به انتخاب کرنل وابسته است. پیچیدگی بالاست. برای مسائل چند کلاسه طراحی سخت است.	ارائه یک راه حل یگانه. توانایی حل مسائل غیرخطی با گرفتن - کرنل‌های متفاوت. وقتی داده‌ها به یک سمت متمایل هستند - هم خوب کار می‌کند.	بردار پشتیبان
عملکرد بسیار بد برای داده‌های با ابعاد بالا.		نژدیک‌ترین همسایگی
بر پایه اکتشافات و نداشتن تئوری دقیق.	صحت بالا. پیاده‌سازی ساده.	مجموع

1. Very fast decision tree algorithm

2. Concept adapting very fast decision tree

۱۲۳  
فصل

## ارزیابی کارایی در یادگیری ماشین

### ۱۳-۱- ارزیابی کارایی در یادگیری ماشین

در تمامی الگوریتم‌های یادگیری ارزیابی نتایج نهایی یکی از بخش‌های جدایی‌ناپذیر است زیرا بدون ارزیابی نحوه عملکرد الگوریتم، نمی‌توان از مناسب یا نامناسب بودن الگوریتم موردنظر آگاهی حاصل نمود. بر این اساس در ارزیابی کارایی الگوریتم‌ها از معیارهای ارزیابی مختلفی استفاده می‌کنند که در این بخش مهم‌ترین آنها را معرفی نموده‌ایم.

### ۱۳-۲- ماتریس درهم‌ریختگی<sup>۱</sup> در ارزیابی کارایی الگوریتم

این ماتریس چگونگی عملکرد الگوریتم دسته‌بندی را با توجه به مجموعه داده ورودی و نحوه تفکیک آنها در دسته‌های مناسب و نامناسب را مورد تحلیل قرار داده و نمایش می‌دهد.

جدول (۱-۱۳): ماتریس درهم‌ریختگی.

		تخمین‌ها یا پیش‌بینی‌ها	
		درست T	نادرست F
نمونه‌های واقعی	ثبت P	TP	FN
	منفی N	FP	TN

ماتریس درهم‌ریختگی چنانچه در جدول (۱-۱۳) مشاهده می‌نمایید دارای چهار عنصر می‌باشد که توضیحات آنها به صورت زیر است:

**TP**: نمونه‌های مثبتی که به درستی مثبت تشخیص داده شده.

**TN**: نمونه‌های منفی که به درستی منفی تشخیص داده شده.

**FN**: نمونه‌های مثبتی که به صورت نادرست منفی تشخیص داده شده.

**FP**: نمونه‌های منفی که به صورت نادرست مثبت تشخیص داده شده.

برای تعیین کارایی یک الگوریتم دقت<sup>۲</sup> یکی از مهم‌ترین معیارها است که این معیار دقت وضعیت یک دسته‌بندی کننده را مشخص می‌کند و مشخص می‌کند که دسته‌بند طراحی شده چند درصد از کل مجموعه رکوردهای آزمایشی را بدرستی دسته‌بندی کرده است. در زیر به تعدادی از رابطه‌های مهم ارزیابی توجه نمایید.

- 
1. Confusion matrix
  2. True positive
  3. False negative
  4. True positive
  5. True negative
  6. Accuracy rate

### ۱۳-۳- ارزیابی نرخ مثبت درست<sup>۱</sup> در استخراج داده

درستی اطلاعات به دست آمده را با فراخوانی<sup>۲</sup> یا TPR محاسبه می‌کنند در اصل به حاصل تقسیم تعداد مستندات بازیابی شده واقعاً با ربط، بر تعداد کل مستندات مرتبط موجود، گفته می‌شود.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{رابطه (۱)}$$

### ۱۳-۴- ارزیابی نرخ مثبت نادرست<sup>۳</sup> در استخراج داده

به آن دسته از داده‌های منفی که به اشتباه در دسته مثبت‌ها کلاسه‌بندی شده‌اند گفته می‌شود که با رابطه زیر به دست می‌آید.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad \text{رابطه (۲)}$$

### ۱۳-۵- ارزیابی نرخ منفی درست<sup>۴</sup> در استخراج داده

به آن دسته از داده‌های منفی که به درستی در دسته منفی‌ها کلاسه‌بندی شده‌اند گفته می‌شود.

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad \text{رابطه (۳)}$$

### ۱۳-۶- ارزیابی نرخ منفی اشتباه<sup>۵</sup> در استخراج داده

نرخ داده‌های مثبتی که به اشتباه در دسته منفی‌ها کلاسه‌بندی شده‌اند را با رابطه زیر مشخص می‌نماییم.

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \quad \text{رابطه (۴)}$$

### ۱۳-۷- روش ارزیابی دقت<sup>۶</sup> الگوریتم

به حاصل تقسیم تعداد مستندات بازیابی شده واقعاً باربط بر تعداد کل مستندات بازیابی شده گفته می‌شود.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{رابطه (۵)}$$

### ۱۳-۸- روش ارزیابی درستی<sup>۷</sup> الگوریتم و حساسیت<sup>۸</sup>

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \text{sensitivity} \quad \text{رابطه (۶)}$$

- 
1. True positive rate
  2. Recall
  3. False positive rate
  4. True negative rate
  5. False negative rate
  6. Precision
  7. Recall
  8. Sensitivity

### ۱۳-۹- روش ارزیابی اختصاصی بودن<sup>۱</sup>

$$\text{specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}} \quad \text{رابطه ۷}$$

### ۱۳-۱۰- روش ارزیابی صحت<sup>۲</sup> الگوریتم

همان نسبت تعداد کل پیش‌بینی‌های درست می‌باشد که با رابطه زیر مشخص می‌شود.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \quad \text{رابطه ۸}$$

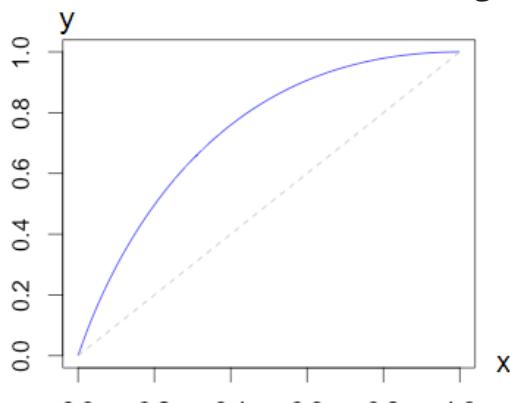
### ۱۳-۱۱- روش ارزیابی F-Measure

ارزیابی F در حیطه‌های مانند استخراج اطلاعات، دسته‌بندی اسناد و ارزیابی کارایی الگوریتم‌های یادگیری ماشین مورد استفاده قرار می‌گیرد. در موافقی که بخواهیم میانگین بین درستی و دقت را محاسبه کنیم از این معیار ارزیابی (F1-measure) استفاده می‌کنیم. در واقع در این روش یک نوع میانگین بین پارامتر precision و پارامتر recall به دست می‌آید که رابطه آن به صورت زیر است:

$$f_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad \text{رابطه ۹}$$

### ۱۳-۱۲- منحنی ROC

یکی از روش‌های پرکاربرد برای ارزیابی کارایی الگوریتم کلاسه‌بندی کننده، منحنی مشخصه عملکرد سیستم یا منحنی عملیاتی دریافت<sup>۳</sup> می‌باشد. این گراف به شکل یک نمودار در محور X خود وضعیت FPR و در محور Y خود وضعیت TPR را نمایش می‌دهد.



شکل (۱-۱۳): نمودار منحنی ROC.

1. Specificity
2. Accuracy
3. Receiver operator characteristic

اگر در نمودار ROC که به صورت شکل (۱-۱۳) است نقطه (X,Y) به صورت (0,1) باشد بهترین حالت کلاسه‌بندی را نشان می‌دهد. اگر نمودار ROC چنین وضعیتی را نشان دهد به معنی این است که درصد مناسبی از نمونه‌های مثبت و نمونه‌های منفی به درستی کلاسه‌بندی شده‌اند.

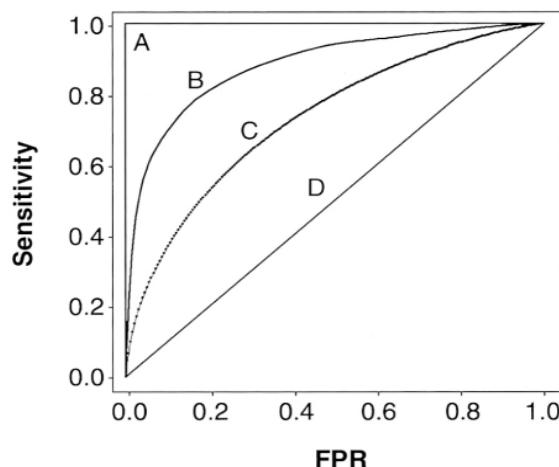
نقطه (0,0) در نمودار نشان‌دهنده این است که تمامی نمونه‌های منفی تخمین‌زده شده‌اند و نقطه (1,1) نشان‌دهنده تخمین تمامی نمونه‌های مثبت است.

نقطه (1,0) نشان‌دهنده نمونه‌های نادرست کلاسه‌بندی شده برای تمامی دسته‌ها است و در برخی از مثال‌های الگوریتم دسته‌بندی دارای پارامتری برای زیاد کردن مقدار TP در مقابل افزایش FP و یا حالت برعکس آن می‌باشد.

با تغییر این پارامتر مانند حد آستانه در شکل (۱-۱۳) مقادیر مختلفی برای خروجی کلاسه‌بند بذست می‌آید و از روی این خروجی‌ها (TPR, FPR) با تغییر این پارامتر محاسبه شده و سپس منحنی به صورت افزایشی رسم می‌شود.

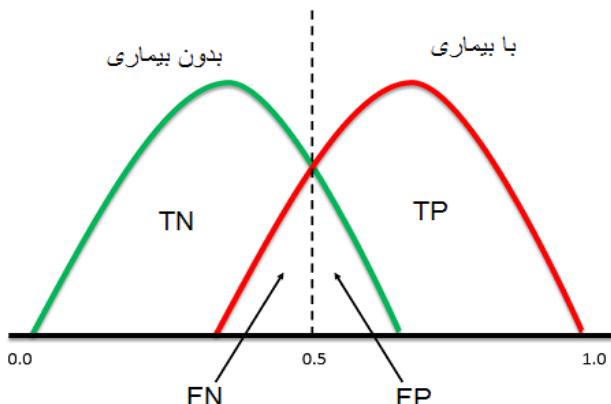
### ۱۳-۱۳- سطح زیر منحنی<sup>۱</sup> (AUC)

در حالت عمومی سطح زیر منحنی ROC نشان‌دهنده معیاری از ارزیابی تلقی می‌شود و بیشتر بودن آن نشان‌دهنده بهتر عمل کردن است. به عنوان مثال در شکل (۲-۱۳) منحنی D مانند این است که خروجی اتفاقی تولید می‌کند و برای پارامترهای مختلف مانند (سطوح مختلف آستانه) نرخ مثبت درست و نرخ مثبت نادرست مساوی ایجاد کرده است عملکرد پایینی دارد. عملکرد الگوریتمی که منحنی A را ایجاد کرده بالاتر از B و C است و عملکرد بالاتر از C و D است.



شکل (۲-۱۳): منحنی زیر نمودار (AUC).

1. Area under curve



شکل (۳-۱۳): نمودار مشخص کننده چهار حالت مختلف ROC.

چنانکه مشاهده می‌نمایید یک خط به صورت عمودی نتایج مربوط به افراد بیمار را از نتایج مربوط به افراد سالم جدا کرده است که این خط به خط حد آستانه معروف است در اصل وقتی که گفته شد با تغییر این حد آستانه می‌توان خروجی‌هایی بدست آورد که حساسیت سیستم را نسبت به افزایش دقت زیاد یا کم کرد؛ منظور تغییر جایگاه همین خط حد آستانه مقداری به سمت راست یا مقداری به سمت چپ در نمودار است.

در این مثال که هدف تشخیص بیماری است حد آستانه باید به نحوی تعیین شود که احتمال خطای تعیین افراد بیمار به سالم به حداقل ممکن و حتی المقادور به صفر برسد و بنابراین FN حداقل موردنظر است. گرچه این مسئله ممکن است باعث افزایش FP شود. افزایش FP مشکل ایجاد هزینه اضافه می‌کند زیرا افراد سالم بیشتری ممکن است بیمار تشخیص داده شده یعنی آزمایشات بعدی و بیشتری روی این افراد باید انجام گیرد تا درنهایت مشخص شود این افراد سالم بوده‌اند. بنابراین برای کاربردهایی مانند مثال فوق تلاش این است که حد آستانه به گونه‌ای تعیین شود که حساسیت بالا باشد. یعنی نزدیک به حداقل مقدار خود که همان ۱ می‌باشد.

### ۱۴-۱۳- مثالی با R برای ترسیم منحنی ROC

در این مثال می‌خواهیم با استخراج داده‌های مربوط به ویژگی خاصی از مجموعه داده iris با استفاده از یک مدل خطی تعمیم‌یافته برای تخمین دقت و درستی داده‌های استخراج شده یک نمودار ROC ترسیم کنیم.

ابتدا بسته نرم‌افزاری ROCR با استفاده از دستور خط اول و دوم نصب و بارگذاری می‌شود.

```
> install.packages("ROCR")
> library(ROCR)
> data("iris")
```

مجموعه داده iris با دستور زیر بارگذاری شده و با استفاده از دستور خط بعدی یکی از گونه‌های مشخص شده که در مجموعه دادهiris وجود دارد انتخاب می‌شود.

```
> iris$isv <- as.numeric(iris$Species == "versicolor")
```

خط زیر یک مدل برای مشخص کردن استخراج داده براساس ویژگی‌های خاص می‌باشد که در اینجا ویژگی‌های خاصی از گونه مشخص شده گلiris درون مجموعه داده این مدل را می‌سازد.

```
> mod <- glm(isv~Sepal.Length+Sepal.Width, data=iris, family="binomial")
```

خط زیر یک پیش‌بینی براساس مدل ایجاد شده و داده‌هایiris انجام می‌دهد.

```
> pred1 <- prediction(predict(mod), iris$isv)
```

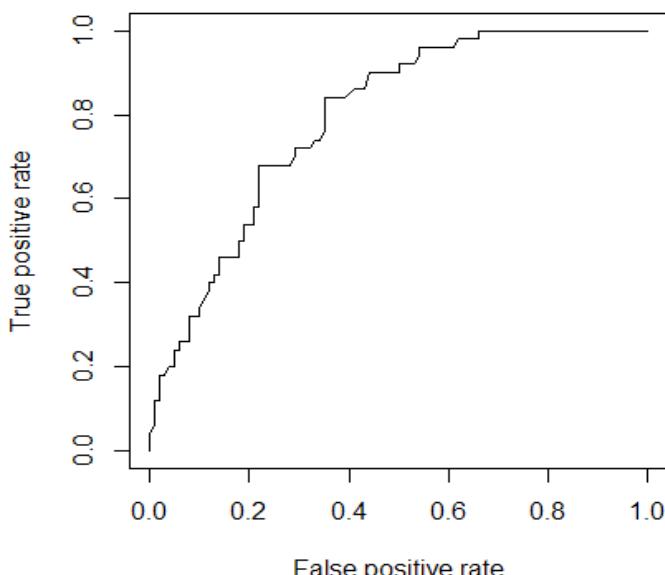
خط زیر کارایی نتیجه به دست آمده در مرحله پیش‌بینی را که همان نرخ داده‌های استخراج شده به صورت false positive و true positive است ارزیابی می‌کند.

```
> perf1 <- performance(pred1,"tpr","fpr")
```

خط زیر نمودار ROC را براساس tpr و fpr مشخص می‌کند.

```
> plot(perf1)
```

خروجی نهایی نمودار ROC به صورت زیر است:



### ۱۵-۱۳- ترسیم نمودار ROC با استفاده از پایتون

هدف این برنامه پیاده‌سازی یک نمودار ROC حاصل از یک الگوریتم دسته‌بندی کننده است که بتواند نتیجه عملکرد الگوریتم را ارزیابی نماید. نمودار ROC در اصل همان نرخ TP را بر روی محور Y و نرخ FP را بر روی محور X ترسیم و مشخص می‌نماید.

شش خط اول برنامه زیر بسته‌های نرم افزاری و کتابخانه‌هایی را که اجرای یک الگوریتم نزدیکترین k همسایه و ROC لازم است را تعریف می‌نماید. مجموعه داده آموزشی در این برنامه را به صورت مصنوعی خودمان تولید می‌نماییم.

```
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
```

خط زیر یک مجموعه داده را تولید می‌نماید که دارای ۲ کلاس می‌باشد.

```
X, y = make_classification(n_samples=1000, n_classes=2, weights=[1, 1], random_state=1)
```

خط برنامه زیر مجموعه داده را به دو بخش داده‌های آموزشی و داده‌های آزمایشی تقسیم می‌نماید.

```
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)
```

خط زیر مدل نزدیکترین k همسایه را ایجاد می‌نماید.

```
model = KNeighborsClassifier(n_neighbors=3)
model.fit(trainX, trainy)
```

خط زیر با استفاده از مجموعه آزمون درستی مدل ایجاد شده را تخمین می‌زند.

```
probs = model.predict_proba(testX)
```

خط زیر فقط نتایج مثبت تخمین شده را مشخص می‌نماید.

```
probs = probs[1, :]
```

خط زیر ناحیه زیر منحنی AUC را محاسبه می‌نماید و در خروجی چاپ می‌نماید.

```
auc = roc_auc_score(testy, probs)
print('AUC: %.3f' % auc)
```

خط زیر ROC را محاسبه می‌نماید.

```
fpr, tpr, thresholds = roc_curve(testy, probs)
```

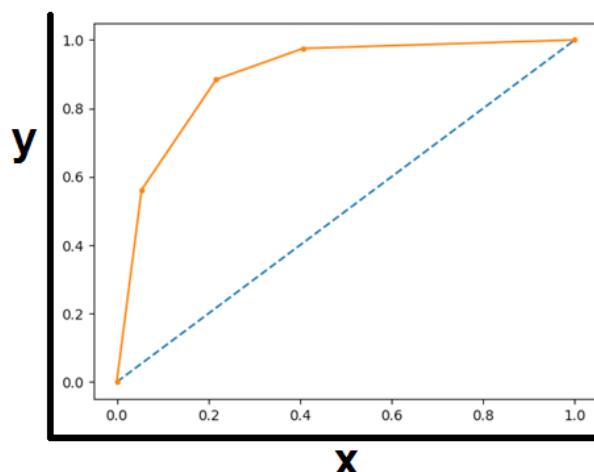
خطوط زیر نمودار ROC را در خروجی چاپ می‌نماید.

```
pyplot.plot([0, 1], [0, 1], linestyle='--')
```

```
pyplot.plot(fpr, tpr, marker='.'
```

```
pyplot.show()
```

نمودار زیر خروجی نهایی برنامه می‌باشد.





۱۲۹  
فصل

## مدل‌های گرافی احتمالاتی

## ۱-۱۴- مدل‌های گرافی احتمالاتی

مدل‌های گرافی احتمالاتی<sup>۱</sup> یک چارچوب مناسب برای ترکیب عدم قطعیت (احتمالات) و ساختار منطقی (قیدهای استقلال) است که به صورت فشرده پدیده‌های پیچیده دنیای واقعی را مدل می‌کند. بسیاری از مدل‌های آماری متداول مانند مدل مخفی مارکوف و یا فیلتر کالمون را می‌توان به صورت مدل‌های گرافی توصیف کرد. مدل‌های گرافی به دلیل انعطاف و قدرت بازنمایی بالا و همچنین بخاطر قابلیت یادگیری مؤثر و استنتاج در شبکه‌های بزرگ، مورد علاقه پژوهش‌گران است.

یک مدل گرافی احتمالاتی، مدلی احتمالی است که حاوی یک گراف بوده و ساختار وابستگی شرطی بین متغیرهای تصادفی را توصیف می‌کند. از این مدل‌ها در تئوری احتمال، آمار و یادگیری ماشین استفاده می‌شود. این مدل‌ها در نظریه احتمال، آمار و مخصوصاً در یادگیری ماشین کاربرد دارند. در حالت کلی، مدل گرافی احتمالاتی از یک بازنمایی گرافی به عنوان پایه‌ای برای کد کردن یک توزیع احتمال کامل بروی یک فضای چندبعدی و همچنین یک گراف که بازنمایی گرافی توزیع‌ها در دو شاخه استفاده می‌شوند، یکی شامل شبکه‌های بیزی و دیگری می‌کند. معمولاً بازنمایی گرافی توزیع‌ها در ارائه الگوریتم‌هایی به منظور کشف و تحلیل توزیع‌های پیچیده و استخراج اطلاعات می‌شود. توانایی این مدل‌ها در ارائه الگوریتم‌هایی به منظور کشف و تحلیل توزیع‌های پیچیده و استخراج اطلاعات غیرساخت‌یافته موجب شده است این مدل‌ها به طور بهینه‌ای ساخته و استفاده شوند. کاربردهای مدل‌های گرافی احتمالاتی شامل استخراج اطلاعات، بازشناسی گفتار، توصیف ساختار پروتئین‌ها، بینایی ماشین و غیره می‌شود.

این مدل‌ها با استفاده از تعدادی متغیر بازنمایی می‌گردند. این متغیرها را می‌توان در دسته‌های متغیرهای مشاهده شده، متغیرهای خروجی و تعدادی متغیر کمکی تقسیم‌بندی کرد. متغیرهای مشاهده شده داده‌های استخراج شده از اطلاعات مانند تصویر، ویدیو صوت و غیره را توصیف می‌کنند و متغیرهای خروجی، مربوط به اطلاعات سطح بالا می‌باشد. همچنین در یک مدل علاوه‌بر تعریف متغیرها باید مشخص شود که این متغیرها چه ارتباطی باهم دارند. با مدل‌های گرافی احتمالاتی می‌توان ارتباط بین چندین متغیر را با استفاده از یک زبان دقیق و خوش‌تعریف‌آ، توصیف کرد. می‌توان از این زبان برای ربط دادن مشاهدات و متغیرهای ناشناخته مسئله به هم استفاده کرد. طبیعتاً در برخی از مسائلی که با آنها روبرو می‌شویم، مشخص کردن قطعی جواب درست با استفاده از مشاهدات ممکن نیست. در این موارد می‌توان از مدل‌های گرافی احتمالاتی استفاده کرد. در این مدل‌ها یک توزیع احتمال توام یا شرطی طوری کد می‌شود که با داشتن تعدادی مشاهده، فقط به یک جواب تقریبی نمی‌رسیم بلکه به یک توزیع احتمالی روی همه راه حل‌های ممکن دست پیدا می‌کنیم. علاوه‌بر این می‌توان از فرض‌های اضافی در قالب توزیع احتمالی پیشین<sup>۲</sup> هم استفاده کرد.

- 
1. Probabilistic graphical models
  2. Well-defined
  3. Prior probability distribution

انواع مختلفی از مدل‌های گرافی احتمالی وجود دارد، اما ویژگی مشترک همگی این است که با استفاده از یک گراف، یک خانواده از توزیع‌های احتمالی را مشخص می‌کنند. انواع مختلف این مدل‌ها در ساختار گراف و فرض‌های استقلال شرطی که در گراف در نظر گرفته شده، با یکدیگر تفاوت دارند. با داشتن یک مدل گرافی احتمالاتی می‌توان آن را به صورت یک فیلتر برای توزیع‌های احتمالی در نظر گرفت که فقط توزیع‌هایی از آن عبور می‌کنند که همه شرط‌های استقلالی که در گراف در نظر گرفته شده را دارا باشد. در نتیجه یک مدل گرافی احتمالی فقط یک توزیع را مشخص نمی‌کند بلکه یک خانواده از توزیع‌های احتمالی را مشخص می‌کند.

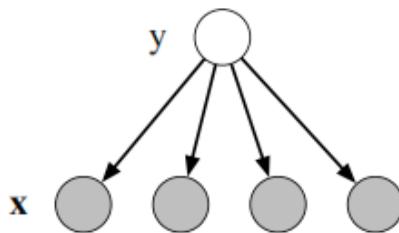
## ۲-۱۴- بازنمایی مدل‌های گرافی

دو دسته عمده مدل‌های گرافی عبارتند از شبکه بیزی<sup>۱</sup> و شبکه مارکوف<sup>۲</sup>. شبکه‌های بیزی با گراف‌های جهت‌دار بازنمایی می‌شوند و بنابراین مدل‌های گرافی جهت‌دار نامیده می‌شوند. بازنمایی شبکه‌های مارکوف نیز با استفاده از گراف‌های بدون جهت بوده و مدل‌های گرافی بدون جهت نامیده می‌شوند. همچنانی ممکن است مدل‌های تربیبی که شامل هر دو بازنمایی جهت‌دار و بدون جهت می‌باشند را داشته باشیم که به اندازه دو دسته قبلی متداول نیستند. مقالات زیادی در دهه‌های گذشته ارائه شده است که استفاده از مدل‌های گرافی برای تجزیه و تحلیل داده‌های متوالی را توصیه کرده است. بیشتر روش‌های موجود این رده احتمال و نظریه گراف را برای پیدا کردن ساختار در داده‌های ترتیبی ترکیب نموده‌اند. این روش را می‌توان بطور گسترده به دو زیر مجموعه مدل‌های گرافی جهت‌دار و مدل‌های گرافی بدون جهت تقسیم کرد. از جمله مدل‌های معروف دسته اول شامل مدل مارکوف پنهان<sup>۳</sup>، شبکه‌های بیزی می‌باشند. میدان‌های تصادفی مارکف<sup>۴</sup>، میدان‌های تصادفی شرطی<sup>۵</sup> نیز از جمله مدل‌های متعلق به دسته دوم می‌باشند. ساده‌ترین مورد از مدل گرافی جهت‌دار مدل مارکوف پنهان است که در آن مشاهدات در حالت جاری به حالت‌های قبلی بستگی دارند. مشاهدات می‌توانند به عنوان نمادهای گستته یا توزیع پیوسته نمایش داده شوند. در تشخیص رویدادها و فعالیت‌های پیچیده، مدل گرافی جهت‌دار توسط گراف بدون دور نشان داده می‌شود که می‌تواند روابط فضایی حالت بین زیر رویدادها یا رویدادهای سطح پایین را مدل کند.

## ۳-۱۴- شبکه‌های بیزی

شبکه‌های بیزی با گراف‌های جهت‌دار بدون حلقه بازنمایی می‌شوند. رأس‌های گراف نمایان گر متغیرهای تصادفی و یال‌ها نیز به صورت شهودی نشان‌دهنده تأثیر مستقیم یک متغیر روی متغیر دیگر است. این گراف را می‌توان به صورت یک ساختار داده برای بازنمایی توزیع احتمال توأم در نظر گرفت. به عنوان مثال در شکل (۱-۱۴) شبکه مربوط به دسته‌بند Naive Bayes از جمله مدل‌های گرافی جهت‌دار قابل مشاهده می‌باشد.

- 1. Bayesian network
- 2. Markov network
- 3. Hidden Markov models
- 4. Markov random field
- 5. Conditional random field



شکل (۱۴-۱): مدلی از یک گراف جهت دار.

هر متغیر تصادفی در شبکه برای خودش یک توزیع احتمال شرطی<sup>۱</sup> دارد که آن را به اختصار CPD می‌نامیم. برای هر متغیر  $X$  در شبکه، با داشتن والدین آن متغیر در گراف، CPD برابر است با احتمال شرطی متغیر  $X$  به شرط داشتن والدین  $X$ . CPD را می‌توان به روش‌های مختلفی بازنمایی کرد. یکی از این روش‌ها استفاده از جدول است که به ازای هر مقدار ممکن برای والدین متغیر  $X$  دارای یک ردیف می‌باشد که در آن احتمال مقادیر مختلف  $X$  داده شده است.

همان‌طور که در مقدمه اشاره شد یکی از ساده‌ترین انواع مدل‌های گرافیکی احتمالاتی، مدل مخفی مارکوف می‌باشد. یک نقطه ضعف از مدل مخفی مارکوف عدم توانایی آن در مدل کردن رابطه علی است. این مشکل توسط نوع متفاوت از مدل گرافی شبکه‌های جهت دار به نام شبکه بیزی<sup>۲</sup> که به اختصار آن را با BN نمایش می‌دهیم، حل می‌گردد. شبکه‌های بیزی قادر به مدل‌سازی مؤثر علیت با استفاده از استقلال شرطی بین حالتاً می‌باشند. این روش با اعمال فاکتور باعث تسهیل معنایی و محاسباتی در فضای حالت می‌شود. شبکه‌های بیزی به طور ضمنی نمی‌توانند اطلاعات زمانی بین گره‌ها یا حالات مختلف در مدل ماشین حالت محدود را مدل کنند. شبکه‌های بیزی پویا<sup>۳</sup> با استفاده از اصول فاکتور گیری که در شبکه‌های بیزی وجود دارد، روابط زمانی را به صورت بهتری مدل می‌کنند. شبکه‌های بیزی و مدل‌های مخفی مارکوف و انواع آنها از لحاظ فلسفی در گروه مدل‌های مولد قرار می‌گیرند. به دلیل ماهیت مولد مدل، توزیع از روی مشاهدات با توجه به حالت یاد گرفته می‌شود. با این حال، در زمان استنتاج و یا دسته‌بندی، تنها مشاهدات در دسترسی می‌باشند. از این رو از نظر شهودی، شرط گذاشتن برروی مشاهدات بهتر از شرط گذاشتن برروی حالت‌ها می‌باشد. این موضوع انگیزه‌ای برای محققان شد تا مدل‌سازی حوادث پیچیده با استفاده از مدل‌های گرافی بدون جهت را مورد مطالعه قرار دهند.

#### ۴-۱۴ - شبکه‌های مارکوف

دومین دسته از مدل‌های گرافیکی احتمالاتی شبکه‌های مارکوف یا میدان تصادفی مارکوف نامیده می‌شود. اساس این مدل‌ها بدون جهت بودنشان است. این مدل‌ها برای مدل‌سازی انواع مختلفی از پدیده‌ها قابل استفاده می‌باشند که در آنها نمی‌توان جهتی را بین متغیرهای تصادفی مختلف متصور شد. علاوه‌بر این، مدل‌های بدون جهت هم

- 
1. Conditional probability distribution
  2. Bayesian network
  3. Dynamic Bayesian network

از لحاظ ساختار استقلال شرطی و هم از لحاظ استنتاج نسبت به مدل‌های جهت‌دار ساده‌تر می‌باشند. یک نوع از بازنمایی برای پیاده‌سازی این نوع از دیدگاه استفاده از گراف بدون جهت است. همانند شبکه‌های بیزی رأس‌های گراف نشان‌دهنده متغیرها هستند و یال‌های گراف نمایان گر ارتباط بین متغیرها می‌باشند.

موضوع دیگری که در این ارتباط مطرح است چگونگی بیان این مدل‌ها در قالب پارامترهای مختلف است. ساختار گراف ویژگی‌های کیفی توزیع احتمالی را توصیف می‌کند. برای بازنمایی این توزیع احتمالی، باید ساختار گراف را به تعدادی پارامتر ارتباط دهیم، مشابه تعریف CPD برای مدل‌های جهت‌دار که بالاتر به آن اشاره شد، با این تفاوت که در شبکه‌های مارکوف فاکتورها متناظر با احتمال یا احتمال شرطی نیستند. برای تبدیل این فاکتورها به احتمال، نیاز به نرمال کردن آنها با استفاده ازتابع پتانسیل است.

- اگر  $H$  را یک شبکه مارکوف در نظر بگیریم، توزیع  $P_H$  این شبکه مارکوف را توصیف می‌کند به شرطی که:
- اگر مجموعه‌ای از زیرمجموعه‌های  $D_m, D_1, \dots$  وجود داشته باشد به طوری که هر  $D_i$  یک زیرگراف کامل از  $H$  باشد.
  - اگر فاکتورهای  $\pi_1[D_1], \dots, \pi_m[D_m]$  را داشته باشیم به طوری که:

$$p_H(x_1, \dots, x_n) = \frac{1}{z} p'(x_1, \dots, x_n) \quad \text{رابطه (1)}$$

که در آن:

$$p'(x_1, \dots, x_n) = \pi_1[D_1] \times \pi_2[D_2] \times \dots \times \pi_m[D_m] \quad \text{رابطه (2)}$$

نرمال نشده است و رابطه (۳):

$$Z = \sum_{x_1, \dots, x_n} p'(x_1, \dots, x_n) \quad \text{رابطه (3)}$$

یک مقدار ثابت برای نرمال کردن فاکتورها می‌باشد. این عبارت تابع پتانسیل نامیده می‌شود. یک توزیع احتمال  $P$  که روی شبکه  $H$  تعریف می‌شود را توزیع گیبس<sup>1</sup> روی  $H$  می‌نامیم. (این نام‌گذاری در فیزیک آماری ریشه دارد.) توجه داشته باشید که در شبکه‌های مارکوف تنها قید روی پارامترهای موجود در فاکتورها غیرمنفی بودن آن‌ها می‌باشد. با توجه به اینکه هر زیرگراف کامل یک زیرمجموعه از یک گروه با ویژگی‌های شبیه به هم<sup>2</sup> است، می‌توان پارامتریزه کردن مدل را با تعریف فاکتور تنها برای یک گروه‌هایی با ویژگی‌های شبیه به هم ساده کرد. به عبارت دیگر، می‌توان فاکتوری برای زیر گروه‌ها با ویژگی‌های شبیه به هم تعریف نکرد. این فاکتورها پتانسیل یک گروه با ویژگی‌های شبیه به هم، نامیده می‌شوند.

#### ۱۴-۱-۱- استقلال متغیرها در شبکه‌های مارکوف

در شبکه‌های مارکوف، ساختار گراف را می‌توان تعدادی فرض استقلال بین متغیرها در نظر گرفت. به صورت شهودی، در این شبکه‌ها تأثیر احتمالات از طریق یال‌های منتقل می‌شود. دو دسته فرض استقلال بین متغیرها

1. Gibbs

2. clique

می‌توان در نظر گرفت، ویژگی‌های مارکوف محلی و ویژگی‌های مارکوف سراسری. ویژگی‌های محلی مارکوف مربوط به یک متغیر و متغیرهای همسایه آن می‌شود، به این صورت که می‌توان با شرط داشتن متغیرهای همسایه یک متغیر از تأثیر سایر متغیرها روی آن متغیر جلوگیری کرد. به عبارت دیگر طبق فرض مارکوف یک متغیر تصادفی در گراف از بقیه متغیرها مستقل خواهد بود در صورتی که مقدار همسایه‌های آن را داشته باشیم.

#### ۱۴-۵- عمل استنتاج<sup>۱</sup> در مدل‌های گرافی

در مدل گرافی برخلاف شبکه‌های دیگر که وزن یک تأثیر محلی دارد، در الگوریتم استنباط یک تأثیر سراسری روی گره‌ها دارد که برای حصول این نتیجه از یک توزیع احتمال پیوسته برای شبکه استفاده شده است. استنتاج یک عمل احتمالی می‌باشد که با استفاده از توزیع‌های احتمال مقادیر احتمال کناری و شرطی را به دست می‌آورد. هدف اصلی از استنباط، تخمین مقادیر گره‌های مخفی می‌باشد که براساس مقادیر داده شده، گره‌های ظاهری محاسبه می‌شوند اگر ما برگ یک مدل تولید‌کننده را مشاهده کردیم و بخواهیم مقادیر گره‌های پنهان را بدست آوریم به این عمل استدلال پایین به بالا می‌گویند و اگر از ریشه شروع کنیم به این عمل استدلال بالا به پایین گویند.

هر دو نوع مدل‌های جهت‌دار و بدون جهت یک توزیع احتمالی روی یک مجموعه متغیر تعريف می‌کند. پاسخ برخی پرس‌وجوه‌های ممکن را می‌توان با استفاده از توزیع احتمال توأم پاسخ داد. یکی از متداول‌ترین انواع پرس‌وجوه، احتمال شرطی است که به صورت  $P(Y | E = e)$  نوشته می‌شود. این پرس‌وجوه شامل دو قسم است: مشاهده که با  $E$  مشخص شده و متغیر تصادفی که مورد پرسش قرار گرفته که در عبارت فوق با  $Y$  مشخص شده است. در اینجا عبارت خواسته شده، حاصل تقسیم احتمال توأم  $Y$  و  $e$  بر مجموع احتمالات  $Y$  به ازای  $E=e$  خواهد بود.

از انواع دیگر پرس‌جوهای متداول، پیدا کردن محتمل‌ترین حالت اختصاص مقادیر به یک زیرمجموعه از متغیرهای است. در این مورد نیز مانند مثال مطرح شده در بالا برای مشاهدات داریم:  $E=e$ . اما در این حالت می‌خواهیم محتمل‌ترین حالت اختصاص مقادیر به زیرمجموعه‌ای از متغیرها را محاسبه کنیم. این مسئله شامل دو نوع است که نوع اول یک حالت خاص از نوع دوم است. ساده‌ترین نوع این مسئله پرس‌جوهای از نوع محاسبه بیشترین توجیه<sup>۲</sup> MPE است. در یک پرس‌جو از این نوع سعی می‌کنیم که محتمل‌ترین احتمال تخصیص مقادیر به همه متغیرها (به جز مشاهدات) را پیدا کنیم.

به عبارت دقیق‌تر اگر داشته باشیم:  $E = X = W$ ، وظیفه ما پیدا کردن محتمل‌ترین تخصیص مقادیر به متغیرهای موجود در مجموعه  $W$  است. قابل ذکر است که ممکن است بیش از یک حالت بهینه برای تخصیص به متغیرها وجود داشته باشد. در نوع دوم از این‌گونه پرس‌جوهای، یک زیرمجموعه از مجموعه متغیرهای  $Y$  داریم که پرس‌جوی مورد نظر ما را شکل می‌دهند که به آن بیشترین احتمال پسین<sup>۳</sup> (MAP) گفته می‌شود.

1. Deduction
2. query
3. Most probable explanation
4. Maximum a posterior probability

## ۱۴-۶- مدل‌های مولد<sup>۱</sup> و تمایزی<sup>۲</sup> در مدل‌های گرافی

یک تفاوت مهم بین مدل‌های NB و رگرسیون لاجستیک (LR) این است که NB مدلی مولد بوده یعنی مبتنی بر یک توزیع احتمال توأم است. در حالی که LR یک مدل تمایزی است، به این معنی که براساس یک توزیع احتمال شرطی است. حال به تفاوت بین مدل‌های تمایزی و مولد و مزیت‌های مدل‌های مولد در بسیاری از موارد پرداخته خواهد شد. برای ملموس‌تر بودن مثال‌ها از مدل‌های LR و NB خواهند بود ولی بحث کلی مقایسه بین مدل‌های تمایزی و مولد است.

تفاوت اصلی بین این دو نوع مدل این است که توزیع شرطی حاوی مدل ( $x$ ) p نیست، به دلیل اینکه این اطلاعات برای دسته‌بندی نیاز نیست. مشکل اصلی در مدل کردن ( $x$ ) p این است که معمولاً شامل تعداد زیادی ویژگی به هم وابسته می‌باشد که برای مدل کردن دشوار است. برای استفاده از ویژگی‌های مستقل از هم در یک مدل مولد، ما دو راه داریم: بهمود مدل برای بازنمایی وابستگی‌های بین ورودی‌ها، و یا در نظر گرفتن فرض‌هایی برای ساده‌سازی مانند فرض موجود در مدل Naive Bayes. انجام دیدگاه اول یعنی بهمود مدل معمولاً دشوار است. دیدگاه دوم، یعنی استفاده از فرض استقلال بین ورودی‌ها، ممکن است که عملکرد را دچار مشکل کند. برای مثال با اینکه دسته‌بند NB در دسته‌بندی متون عملکرد خوبی دارد، اما در کل دقت آن پایین‌تر از رگرسیون لاجستیک است. مزیت اصلی مدل‌های تمایزی این است که برای حالتی که ویژگی‌های دارای همپوشانی داریم بهتر عمل می‌کنند.

## ۱۴-۷- ساختار مدل گرافی

یکی دیگر از مسائلی که در هنگام استفاده از مدل‌های گرافی باید مشخص شود انتخاب ساختار مناسب مدل گرافی است. به طور کلی در ساخت مدل‌های گرافی دو استراتژی وجود دارد.

۱. آموزش خودکار ساختار مدل گرافی (یادگیری ساختار) براساس تعدادی قید.
۲. ساخت دستی ساختار مدل براساس دانش قبلی که انسان از مسئله دارد.

هرچند یادگیری خودکار ساختار می‌تواند تطبیق بیشتری با مسئله داشته باشد و نتایج بهتری را دربر داشته باشد ولی یادگیری چنین ساختاری حتی برای مدل‌های گرافی ساده مانند شبکه‌های بیزی بسیار دشوار است. در اینجا تمرکز ما بر روی ساختار مدل به صورت دستی می‌باشد. به منظور ساخت مدل Chain Graph به صورت دستی روابط بین متغیرها را به صورت پیوندهای یک طرفه و دو طرفه براساس معنی بین آنها و طبیعت این رابطه در نظر می‌گیریم. برای مثال در صورتی که رابطه به صورت علی و یا یک طرفه باشد از پیوندهای جهت‌دار و در صورتی که رابطه متقابل یا دو طرفه باشد از پیوندهای بدون جهت استفاده می‌کنیم. در حالاتی که بتوان از هر دو نوع ارتباط استفاده کرد مدلی را انتخاب می‌کنیم که به ساده شدن مدل کلی گراف منجر شود.

---

1. generative  
2. discriminative

## ۱۴-۱- به دست آوردن پارامترهای مدل‌های گرافی

هنگامی که از روی مدل ساخته شده توزیع احتمال توأم (JPD)<sup>۱</sup> را به دست می‌آوریم تعدادی پارامتر مجھول در JPD باقی می‌مانند که می‌بایست به دست آیند. به این فرایند، پارامترسازی می‌گویند. به طور کلی در صورتی که پیوندها بدون جهت باشند، پارامترها با استفاده از توابع پتانسیل به دست آمده ولی در صورتی که پیوندها جهت‌دار باشند، پارامترها با استفاده از احتمالات شرطی محلی به دست می‌آیند. ولی در اینجا که به صورت توأم از هر دو فرم پیوندهای جهت‌دار و بدون جهت استفاده می‌شود، حالت‌های پیچیده‌تری به وجود می‌آید.

## ۱۴-۹- توزیع احتمال توأم مدل گرافی ترکیبی

در صورتی که گراف ما تنها از پیوندهای جهت‌دار تشکیل شده باشد، JPD را می‌توان با استفاده از factorization بر روی گراف به دست آورد، ولی اگر گراف ما از تعدادی یال جهت‌دار و بدون جهت تشکیل شده باشد:

۱. تمامی متغیرهای یکه یا پیوندهای بدون جهت به یکدیگر متصل هستند و به عنوان یک متغیر در نظر گرفته می‌شوند. در نتیجه گراف حاصل جهت‌دار خواهد بود و همچنین JPD مربوط به مدل گرافی به دست آمده را با استفاده از Factorization به دست می‌آوریم.
۲. هر کدام از متغیرهای یکه از ادغام<sup>۲</sup> چند متغیر به دست آمده‌اند و بین آنها پیوندهای بدون جهت وجود دارد که بر این اساس ما با استفاده از توابع پتانسیل، احتمالات هر بخش را به دست می‌آوریم. همچنین برای هر بخش از آن متغیرهای یکه که به شکل جهت‌دار به یکی از متغیرهای آن قسمت متصل هستند را در قسمت شرطی تابع احتمال آن قسمت می‌آوریم.
۳. توابع احتمال شرطی به دست آمده در قسمت ۲ را در ۱ جایگزین می‌کنیم.

JPD به دست آمده در بالا فرمی به شکل زیر دارد:

$$p(x) = \frac{1}{z} \prod_{(x_i, x_j)} \Psi_l(x_i, x_j; \theta_{\Psi_l}) \cdot \prod_m f_m(x_{n_m}; \{\theta_{\Psi_l}\}) \cdot \prod_s p(x_s | pa(x_s)) \quad (رابطه ۵)$$

که در آن  $\Psi_l(x_i, x_j; \theta_{\Psi_l})$  تابع پتانسیل است که تعامل بین متغیرها را مدل می‌کند. تابع عامل  $f$  نتیجه اجرای نرمال‌سازی محلی در گراف‌های جهت‌دار است و  $P$  پارامتر احتمال شرطی محلی است. در اینجا  $pa$  به معنی پدر آن گره  $x$  در گراف می‌باشد. همان‌طور که در بالا مشخص می‌باشد دو پارامتر مجھول در این فرمول  $\theta_{\Psi_l}$  و  $p(x_s | pa(x_s))$  می‌باشند.

## ۱۴-۱۰- یادگیری پارامترها در مدل‌های گرافی

به طور کلی یادگیری پارامترها در مدل‌های گرافی بدون جهت مشکل تر از یادگیری پارامترها در مدل‌های گرافی جهت‌دار است. به همین جهت در یادگیری پارامترهای مدل‌های گرافی بدون جهت مسئله را با تغییر یا تقریب تابع هدف ساده می‌کنند و از به دست آوردن مقدار دقیق تابع هدف یا مشتق<sup>۳</sup> آن اجتناب می‌کنند. تلاش‌های زیادی

1. Joint Probability Distribution

3. Derivative

2. Polling

به منظور یادگیری پارامترها در مدل‌های گرافی جهت‌دار و بدون جهت انجام شده است، ولی کارهای زیادی در رابطه با یادگیری مدل‌های گرافی ترکیبی صورت نگرفته است. یکی از راههای عمومی به منظور یادگیری پارامترها در این مسأله استفاده از روش برآورد حداقل احتمال (MLE)<sup>۱</sup> می‌باشد. فرض کنید تعداد K عدد داده آموزش داریم که در آن  $X$  بیان‌گر مقدار همه متغیرهای تصادفی در k امین نمونه می‌باشد. هدف MLE بیشینه کردن لگاریتم احتمال پارامترها است.

$$L(\theta) = \log \prod_{k=1}^K P(x^k | \theta) = \sum_{k=1}^K \log P(x^k | \theta) \quad (6)$$

که در اینجا  $\theta$  تمامی پارامترهای موجود در مدل را شامل می‌شود. در یادگیری پارامترهای تابع پتانسیل که تابع لگاریتم احتمال ( $L$ ) است می‌بایست نسبت تابع پتانسیل بیشینه شود.

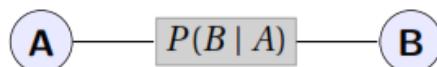
## ۱۱-۱۴- استنتاج در مدل‌های گرافی

گراف عامل<sup>۲</sup> یکی از راههایی است که می‌توان با استفاده از آن در مدل‌های گرافی استنباط انجام داد ولی در ابتدا گراف را در هر فرمی که می‌باشد باید به شکل گراف عامل تبدیل کرد و سپس با روش‌های موجود استنباط را انجام داد. تبدیل مدل‌های گرافی جهت‌دار و بدون جهت به گراف عامل به سادگی امکان‌پذیر است. گراف عامل در مدل‌های گرافی، بین گره‌های موجود که هر کدام نشان‌دهنده یک متغیر می‌باشند یک گره اضافه می‌کند که معادل تابع محلی می‌باشد، پس از آن بین گره‌های اصلی و گره‌های اضافه شده پیوندهای بدون جهت اضافه می‌شود تا گراف عامل به دست آید. در نتیجه تابع کلی گراف یا استفاده از توابع محلی به دست می‌آید. در مدل‌های گرافی جهت‌دار مانند BN توابع گره‌های عامل احتمالات شرطی بین متغیرها می‌باشند و گره‌های عادی گراف عامل متغیرهای توزیع می‌باشند. برای مثال BN شکل (۲-۱۴) را در نظر بگیرید.



شکل (۲-۱۴): گراف جهت‌دار.

گراف عامل آن به شکل زیر خواهد بود:



شکل (۳-۱۴): گراف عامل.

مقدار  $P(B | A)$  در گره عامل قرار می‌گیرد. در مدل‌های گرافی بدون جهت، نیز به همین صورت عمل می‌شود با این تفاوت که در گره‌های عامل مقدار تابع پتانسیل بین دو گره قرار می‌گیرد.

- 
1. Maximum likelihood estimation
  2. Factor Graph

برای مثال گراف زیر را در نظر بگیرید.



شکل (۴-۱۴): گراف اولیه.

گراف عامل آن به شکل زیر است:



شکل (۵-۱۴): گراف عامل.

گره عامل در این گراف معادل باتابع پتانسیل در تابع توزیع تؤام میباشد. بنابر آنچه بیان شد میتوان گراف عامل را در مدل های گرافی ترکیبی به دست آورد. در مدل های گرافی ترکیبی توزیع احتمال تؤام حاصل ضرب دو عامل تابع پتانسیل و احتمالات شرطی میباشد. با داشتن این احتمال توزیع تؤام برای هر زیر گراف ما میتوانیم با اعمال قوانینی که در بالا به آن اشاره شد مدل گرافی تؤام را به نمایش گراف عامل تبدیل کنیم. در این گراف عامل گره های متغیرها همان متغیرهای تصادفی میباشند که در مدل گرافی مرکب حضور داشته اند. بعلاوه گره های عامل به دلیل نمایش تابع پتانسیل احتمال تؤام به این گراف اضافه میشوند. با روشنی که بیان شد میتوان گراف عامل را به دست آورد و پس از آن با روش های مختلفی، استنتاج را روی آن انجام داد. روش های جمع ضرب ها<sup>۱</sup> و بیشینه ضرب ها<sup>۲</sup> دو روش معروف به منظور استنتاج میباشند. روش جمع ضرب ها براساس سازو کار تبادل پیام عمل میکند و دو نوع پیام متصور وجود دارد: یکی پیام هایی که از متغیرها به گره های عامل ارسال میشود و دیگری پیام از گره عامل به متغیر، و در صورتی که پیام ها به درستی مقداردهی اولیه شوند پس از تعدادی به روزرسانی، گراف عامل همگرا خواهد شد. احتمال های حاشیه ای یا استفاده از این پیام ها قابل محاسبه میباشند. در صورتی که گراف به شکل درخت باشد، احتمال های حاشیه ای به صورت دقیق استنتاج خواهد شد. احتمال های حاشیه ای به صورت زیر قابل محاسبه است.

$$P(x_i) = \sum_{x \setminus x_i} p(x) \quad \text{رابطه (7)}$$

که در رابطه بالا  $x$  نشان دهنده تمامی متغیرها میباشد همان طور که مشخص است جمع بر روی تمامی متغیرها به غیر از متغیری که میخواهیم توزیع آن را به دست آوریم انجام میشود. با داشتن توزیع حاشیه ای هر متغیر میتوان مقدار بهینه حالت آن را با استفاده از روش MAP به دست آورد. روش دیگر استفاده از الگوریتم بیشینه ضرب ها است که در آن مقدار متغیرها به نحوی تغیین میشود که ضرب متغیرها یا به عبارت دیگر توزیع احتمال تؤام مقدار بیشینه را داشته باشد.

$$x_i^* = \operatorname{argmax} P(x_i) \quad \text{رابطه (8)}$$

- 
1. Sum-product
  2. Max-product

## ۱۴-۱۳- برنامه بیزین با کدهای R

در این برنامه می‌خواهیم براساس تئوری Naive Bayesian مجموعه داده گل iris دسته‌بندی کنیم. در اینجا نیز از مجموعه داده iris مانند تعدادی از برنامه‌های فصل‌های گذشته به دلیل استاندارد بودن و همچنین آشنایی نسبی شما با آن استفاده شده است با استفاده از دسته‌بند بیزی از روی ویژگی‌های مختلف این مجموعه داده عمل دسته‌بندی را انجام دهیم.

از بسته نرم‌افزاری e1071 برای پیاده‌سازی برنامه استفاده نموده‌ایم که پیش از پیاده‌سازی برنامه باید نصب و بارگذاری شود.

```
>install.packages("e1071")
>library("e1071")
```

برای بارگذاری مجموعه داده iris از دستور زیر استفاده می‌شود.

```
> data(iris)
```

خلاصه‌ای از مجموعه داده را می‌توان با دستور زیر مشاهده نمود.

```
> summary(iris)
```

با تایپ نام زیر نیز می‌توانید محتویات مجموعه داده را چاپ کنید.

```
> iris
```

حال با توجه به اینکه مراحل آماده‌سازی مجموعه داده و بارگذاری بسته‌های نرم‌افزاری را انجام داده‌ایم می‌توانیم مدل دسته‌بندی‌کننده را که یک دسته‌بندی‌کننده بیزی است، با استفاده از خطوط زیر ایجاد کنیم.

```
> classifier<-naiveBayes(iris[,1:4], iris[,5])
```

با خط برنامه زیر نیز عمل تخمین را انجام داده و نتیجه را در یک ماتریس در هم ریختگی چاپ می‌کنیم.

```
> table(predict(classifier, iris[,-5]), iris[,5])
```

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	47	3
virginica	0	3	47

```
> classifier$apriori
```

```
iris[, 5]
```

```
setosa versicolor virginica
```

```
50      50      50
```

با دستور زیر نیز سه توزیع گوسی برای هر متغیر پیش‌بینی کننده، یکی برای هر مقدار از گونه‌های متغیر کلاس ایجاد می‌شود.

```
> classifier$tables$Petal.Length
```

	Petal.Length	
	[,1]	[,2]
iris[, 5]		
setosa	1.462	0.1736640
versicolor	4.260	0.4699110
virginica	5.552	0.5518947

## ۱۴-۱۳- برنامه بیزین با زبان پایتون

در این برنامه با استفاده از مدل احتمالی بیزین یک دسته‌بندی کننده با ناظر پیاده‌سازی شده است که مجموعه داده Iris را به سه کلاس دسته‌بندی می‌نماید.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
```

در سه خط زیر مجموعه داده iris به برنامه افزوده می‌شود که تنها دو ویژگی اول موجود در مجموعه داده در اینجا مدنظر قرار دارد.

```
iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target
```

خط زیر با استفاده از clf.fit() دسته بند را برای دریافت داده و دسته‌بندی آنها ایجاد می‌نماید تا مدل آموزش دیده را ایجاد نماید.

```
h = .02
clf = GaussianNB()
clf.fit(X, Y)
```

در زیر تخمین نهایی دسته داده‌های آزمایشی با استفاده از `clf.predict()` انجام می‌شود و نمودار خط تصمیم ترسیم می‌شود. برای این منظور برای هر نقطه داده موجود در نمودار یک رنگ مشخص کننده دسته مورد نظر اختصاص داده می‌شود.

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

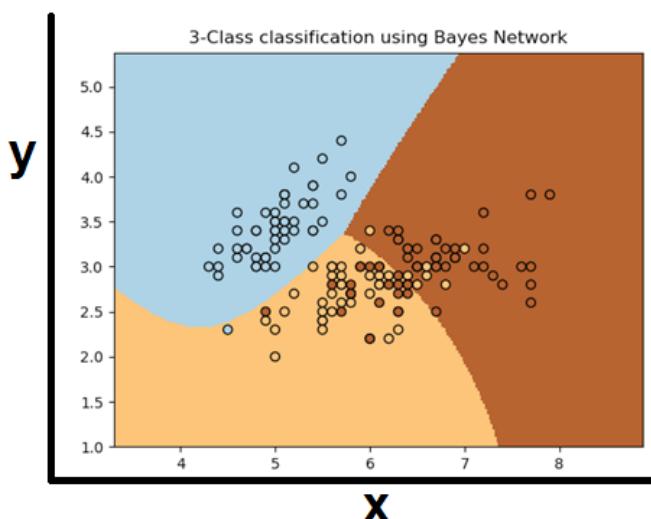
خطوط زیر نقاط داده‌ای را به صورت رنگی در می‌آورند.

```
Z = Z.reshape(xx.shape)
plt.pcolormesh(xx ,yy ,Z ,cmap=plt.cm.Paired)
```

خطوط زیر نمودار نهایی را در خروجی چاپ می‌نمایند که خط اول کار ترسیم بلوک‌های رنگی در نمودار را به عهده دارد.

```
plt.scatter(X ,[0 ,:]X ,[1 ,:]c=Y ,cmap=plt.cm.Paired ,edgecolors='k'
plt.title('3-Class classification using Bayes Network')
plt.axis('tight')
plt.show()
```

نمودار زیر خروجی نهایی برنامه می‌باشد که داده‌ها را به سه دسته مختلف تقسیم نموده است.





۱۰  
فصل

## مقدمه‌ای بر یادگیری عمیق

## ۱-۱۵- مقدمه‌ای بر یادگیری عمیق

یادگیری عمیق به عنوان یک مدل توسعه یافته از روش‌های سنتی یادگیری ماشین مانند شبکه عصبی پرسپترون است که تلاش می‌کند مفاهیم انتزاعی سطح بالا را با استفاده از یادگیری در سطوح و لایه‌های مختلف مدل کند که برای آن ساختارهای مختلفی مطرح شده است. از جمله پرکاربردترین آنها می‌توان به شبکه‌های کانولوشنال<sup>۱</sup>، شبکه‌های بازگشتی<sup>۲</sup> و شبکه‌های باور عمیق<sup>۳</sup> اشاره کرد که به دلیل ارائه نتایج بسیار مناسب در استخراج ویژگی و دسته‌بندی، مراکز علمی تحقیقاتی و شرکت‌های معروف و بزرگ دنیا آنها را در کانون توجه خود قرار داده‌اند و از آنها در حوزه‌های بینایی ماشین، پردازش زبان طبیعی و غیره، جهت ارائه راه حل برای مسائل پیچیده و یا ایجاد بسیاری از نرم‌افزارهای کاربردی و هوشمند استفاده می‌کنند.

## ۲-۱۵- دسته‌بندی انواع روش‌های یادگیری عمیق

۱. مدل تمایزی.<sup>۴</sup> ۲. مدل مولد.<sup>۵</sup> ۳. مدل ترکیبی.<sup>۶</sup>

### ۱-۲-۱- مدل تمایزی یا پیش‌بینی کننده

مدل‌های تمایزی، مدل‌های شرطی نیز نامیده می‌شوند. در مدل‌های تمایزی الگوریتم مورد استفاده سعی می‌کند تا یک مدل آماری را برای تخمین تابع یاد بگیرد که به تابع احتمال پسین معروف است و شکل تابع آن به صورت  $p(y|x)$  است که در آن  $x$  نمونه ورودی و  $y$  نمونه خروجی می‌باشد. به عنوان مثال اگر  $x$  یک تصویر ورودی باشد  $y$  نیز یک شیء بخصوص درون تصویر، مانند یک گربه موجود در درون یک تصویر است. به عبارتی تابع احتمالی  $p(y|x)$  به ما می‌گوید که به چه اندازه مدل باور دارد که با توجه به در نظر گرفتن تمامی حالات مشابه دیگر براساس ویژگی‌های به دست آمده از یک اطلاعات مانند یک تصویر، این شیء درون تصویر یک گربه است. الگوریتم‌هایی که تلاش می‌کنند تا این فرایند احتمالی را به صورت مستقیم مدل کنند به آنها مدل‌های تمایزی یا مدل‌های پیش‌بینی کننده گویند.

### ۲-۲-۱- مدل‌های مولد

مدل‌های مولد در عمل با مدل‌های تمایزی متفاوت هستند به این صورت که یک مدل مولد تلاش می‌کند تا توابع مرتبط را یاد بگیرد که به تابع احتمال مشترک<sup>۷</sup> معروف است. در این مدل اگر تابع ما به صورت  $(x, y)$  باشد این تابع نشان‌دهنده آن است که چه اندازه مدل ایجاد شده باور دارد که  $x$  یک تصویر است و در عین حال  $y$  یک گربه است که درون همان تصویر  $x$  قرار دارد که این دو احتمال  $x$  و  $y$  مرتبط به هم هستند و رابطه آن به صورت

- 
1. Convolutional neural network
  2. Recurrent network
  3. Deep belief network
  4. Discriminative model
  5. Generative model
  6. Hybrid model
  7. Joint probability function

$p(y, x) = p(x) p(y|x)$  می‌باشد که در آن  $p(x)$  نشان‌دهنده درجه مثبت بودن احتمال این است که  $x$  یک تصویر باشد. معمولاً  $p(x)$  در این حوزه به تابع چگالی<sup>۱</sup> معروف است.

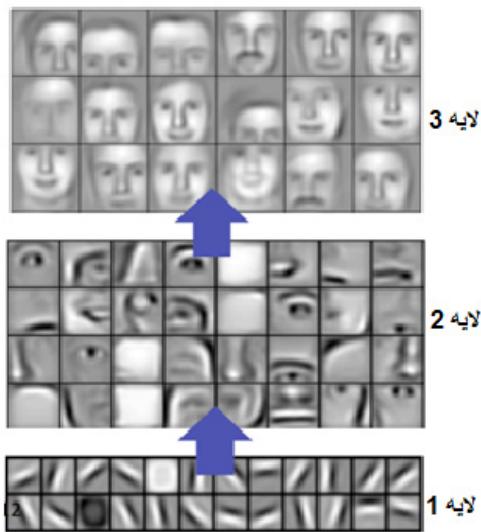
دلیل اینکه این مدل را به مولد نام‌گذاری کرده‌اند این است که این مدل می‌تواند به صورت همزمان درجه احتمال وقوع ورودی و خروجی را بررسی کند و یکی از کاربردهای آن در بینایی ماشین این است که با نمونه‌برداری از تصاویری مانند تصاویر حیوانات، یک تصویر جدید مصنوعی شبیه به آن حیوانات در خروجی تولید کند.

### ۱۵-۳-۲- مدل ترکیبی در روش‌های یادگیری عمیق

با توجه به توضیحاتی که درباره مدل تمایز ارائه شد امروزه این مدل‌های تمایزی به خاطر کسب موفقیت‌های قابل توجهی در برخی از مسائل نسبت به مدل‌های مولد، این مدل‌های تمایزی به صورت گسترده‌تر مورد استفاده محققین قرار می‌گیرند. مدل‌های تمایزی به راحتی می‌توانند براساس ورودی‌های دریافتی، پیش‌بینی‌هایی را به عنوان نتیجه در خروجی ارائه کنند اما به خودی خود نمی‌توانند مانند یک مدل مولد یک نمونه جدید را خلق<sup>۲</sup> کنند. با توجه به اینکه هر کدام از این مدل‌های مولد و تمایزی براساس کاربردهای متفاوت دارای مزای و ممنوعیت متفاوتی هستند بر این اساس می‌توان با ترکیب کارایی این دو مدل، یک مدل ترکیبی کارامدی برای حل مسائل گوناگون ایجاد کرد. معمولاً نتایج ارائه شده توسط مدل‌های تمایزی دارای قابلیت اطمینان بالاتری هستند و یا به دلایل مختلف ممکن است بخواهیم یک مسئله مولد را به یک مسئله تمایزی تبدیل کنیم که نتیجه به دست آمده قابل قبول‌تر باشد. برای این منظور می‌توانیم با یک خود رمزکننده، یک مدل مولد را به یک مدل تمایزی تبدیل کرده و با استفاده از توانایی بالای الگوریتم‌های یادگیری عمیق مانند شبکه عصبی کانولوشن در حل مسائل تمایزی به یک نتیجه بسیار بهتری برسیم. به این نکته باید توجه کرد که به دلیل عدم وجود یک روش استاندارد که بتواند تمامی مسائل را حل کند روش‌های ترکیبی متفاوتی برای این مدل‌ها می‌توان اتخاذ کرد.

### ۱۵-۳-۳- استخراج ویژگی خودکار

نیاز اصلی هر الگوریتم یادگیری، ویژگی‌هایی است که از ورودی‌ها استخراج می‌شود. ممکن است این ویژگی‌ها از پیش به صورت دستی تهیه شده و به الگوریتم به صورت ورودی داده شود که این روش در الگوریتم‌های با نظارت به کار می‌رود. در مقابل روش‌های بدون نظارت هستند که خود اقدام به استخراج ویژگی‌ها از ورودی خواهند نمود. استخراج دستی ویژگی‌ها علاوه‌بر اینکه زمان بر است معمولاً با مشکلاتی مانند استخراج ویژگی نامناسب و برخی مشکلات دیگر را در پی دارد. یادگیری عمیق برای ما یک راه استخراج خودکار ویژگی‌ها پدید می‌آورد که بتوانیم مفاهیم با سطح انتزاع بالا را با استفاده از یادگیری چندلایه، از پایین به بالا بسازیم. در شکل زیر می‌توانید این فرایند را در سه لایه مشاهده نمایید.



شکل ۱۵-۱: بازنمایی ویژگی‌های استخراج شده در لایه‌های مختلف شبکه‌های عمیق.

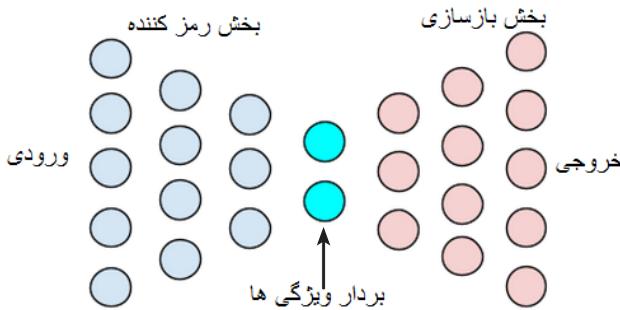
## ۴-۱۵- خود رمزکننده<sup>۱</sup>

الگوریتم خود رمزکننده کاربردهای گسترده و بسیار جالب توجهی مانند استخراج ویژگی‌ها به صورت خودکار، فشرده‌سازی داده، مصورسازی داده‌ها و سایر کاربردهای اساسی را دارد اما در سال‌های ۲۰۰۶ و ۲۰۰۷ محققان به این امر پی برندند که این الگوریتم می‌تواند کاربرد بسیار مهم‌تری مانند حل مشکلات مربوط به آموزش شبکه‌های بسیار عمیق دیگر را در یک مرحله پیش‌پردازشی داشته باشد.

### ۴-۱۵-۱- عملکرد کلی یک خود رمزکننده

به صورت کلی یک خود رمزکننده از دو بخش اصلی تشکیل می‌شود، اولی یک رمزکننده<sup>۲</sup> و دومی یک رمزگشای<sup>۳</sup> یا بازسازی کننده است. بخش رمزکننده مدل تلاش می‌کند تا یک نگاشت<sup>۴</sup> مناسب از فضای ورودی<sup>۵</sup> را به فضایی که به فضای مخفی<sup>۶</sup> معروف است یاد بگیرد و نتایج را در یک بردار به نام بردار ویژگی‌های فشرده<sup>۷</sup> ذخیره کند و در بخش دوم خود رمزکننده که همان بخش رمزگشای الگوریتم است تلاش بر یادگیری یک نگاشت مناسب از بردار ویژگی فشرده شده به یک فضای بزرگ‌تر در خروجی می‌نماید. شکل زیر یک نمای کلی از خود رمزکننده را نشان می‌دهد که قسمت چپ تصویر مربوط به رمزکننده و قسمت میانی بردار ویژگی فشرده شده و قسمت سمت راست مربوط به بازگشاینده یا بخش بازسازی است.

- 
- 1. Auto encoder
  - 2. Encoder
  - 3. Decoder
  - 4. Map
  - 5. Input space
  - 6. Hidden space
  - 7. Compressed feature vector



شکل (۱۵-۲): یک نمای کلی از یک خودرمزنده.

با توجه به توضیحات ارائه شده از خودرمزنده علاوه‌بر یک روش استخراج ویژگی می‌توان به عنوان یک الگوریتم فشرده‌سازی نیز استفاده کرد به این صورت که بخش رمز کننده را یک الگوریتم فشرده‌ساز و بخش بازگشایی کننده را یک بازسازی کننده داده‌های فشرده شده می‌توان در نظر گرفت.

### ۱۵-۴-۳- یادگیری ویژگی با خودرمزنده

خودرمزنده‌ها یک بازنمایی جامع از نمونه‌های بدون برچسبی که می‌بینند را یاد می‌گیرند. فرض کنید مجموعه  $X = \{x_1, x_2, \dots, x_n\}$  داده‌های آموزشی باشند که هدف ایجاد یک بازنمایی (یک مجموعه ویژگی) از داده‌های ورودی است این کار با بهینه کردن رابطه زیر انجام می‌شود.

$$\sum_{i=1}^n \|x_i - \delta(w_2(w_1 x_i))\|^2$$

که  $w_1$  و  $w_2$  به ترتیب وزن‌های لایه ورودی به لایه نهان و لایه نهان به لایه خروجی است،  $\delta$  نیز تابع فعالیت استفاده شده در شبکه عصبی است. این فرمول با استفاده از گرادیان نزولی و همانند یک شبکه عصبی بهینه می‌شود و جهت بهینه‌سازی این رابطه ورودی و خروجی شبکه را با هم مساوی قرار می‌دهیم.

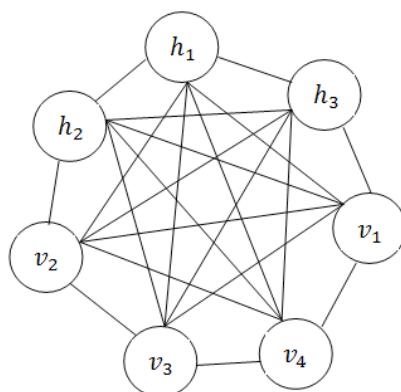
به عبارتی دیگر خودرمزنده تلاش می‌کند تا یک تابع همانی را یاد بگیرد به گونه‌ای که ورودی‌های شبکه عصبی را به خود آن نگاشت کنند. بعد از اینکه این شبکه به صورت کامل آموزش داده شد، خروجی لایه مخفی به عنوان بازنمایی داده ورودی در نظر گرفته می‌شود، به بیان دیگر هنگام استفاده از خودرمزنده‌ها لایه آخری حذف می‌شود و هر داده ورودی به خروجی لایه‌های نهان نگاشت می‌شود.

### ۱۵-۵- ماشین بولتزمن

ماشین بولتزمن<sup>۱</sup> یکی دیگر از الگوریتم‌های بدون نظارت در یادگیری عمیق می‌باشد که توسط جفری هینتون<sup>۲</sup> ارائه شد. ماشین بولتزمن شبکه‌ای از واحدهای نرون مانند است که به صورت متقاضی به هم متصل می‌باشند و

1. Boltzman machine  
2. Geoffrey hinton

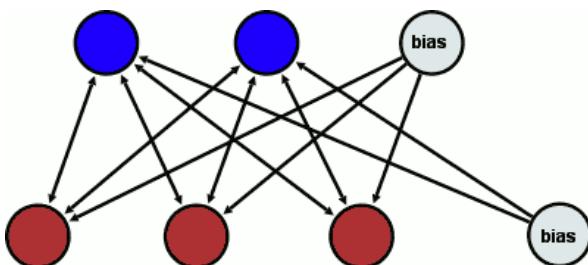
به صورت اتفاقی<sup>۱</sup> درباره فعال یا غیرفعال بودن تصمیم می‌گیرند. این ماشین یک الگوریتم یادگیری ساده‌ای دارد که می‌تواند ویژگی‌های موردنظر را در مجموعه داده‌های مشکل از بردارهای باینری کشف کرده و یاد بگیرد. این الگوریتم در صورتی که برای کشف ویژگی‌ها تعداد لایه‌های زیادی داشته باشد به کندی عمل می‌کند. این الگوریتم هم برای حل مسائل جستجو و هم برای حل مسائل مربوط به یادگیری مورد استفاده قرار گرفته است. در شکل (۳-۱۵) می‌توانید یک مدل از ماشین بولتزمن را مشاهده نمایید.



شکل (۳-۱۵): نحوه آرایش نرون‌ها در الگوریتم ماشین بولتزمن.

### ۱-۵-۱- ماشین بولتزمن محدود

ماشین بولتزمن محدود<sup>۲</sup> یک مدل توسعه‌یافته از ماشین بولتزمن است که شامل یک لایه آشکار و یک یا چند لایه مخفی از نرون‌ها است. نرون‌های هر لایه هیچ ارتباطی بین هم نداشته ولی هر کدام از نرون‌ها به تمامی نرون‌های لایه دیگر اتصال دارند که می‌توانید در شکل (۴-۱۵) مشاهده نمایید. در هر اتصال بین نرونی داده‌ها در دو جهت به صورت مشابه می‌توانند حرکت کنند که مقادیر وزن‌ها برای هر دو جهت یکسان است. این مدل از ماشین بولتزمن را به دلیل عدم وجود ارتباط بین گره‌های یک لایه، بولتزمن محدود می‌گویند.

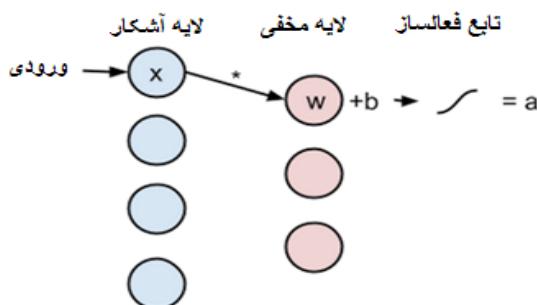


شکل (۴-۱۵): نحوه اتصال نرون‌های ماشین بولتزمن محدود.

1. Stochastic  
2. Restricted boltzman machine

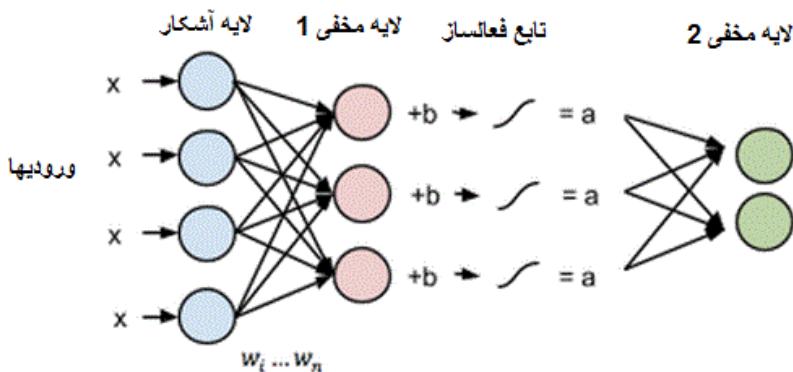
در این ماشین هر گره موجود در لایه آشکار ویژگی سطح پایینی از آیتم موجود که در مجموعه داده قرار دارد را برای عمل یادگیری بر می‌دارد. برای مثال در پردازش یک تصویر اگر تصویر مورد نظر دارای ۸۵۴ پیکسل باشد، ماشین بولتزمن محدود در لایه آشکار خود از ۸۵۴ گره برای اختصاص هر پیکسل به یک گره استفاده می‌کند. اگر یک مقدار موجود برای یک پیکسل را  $x$  در نظر بگیریم این  $x$  در یک وزن ( $w$ ) ضرب شده و مجموع آن با یک مقدار بایاس ( $b$ ) جمع می‌شود و نتیجه این محاسبات به یک تابع فعال کننده ( $f$ ) که یک تابع حد آستانه‌گیری است انتقال می‌یابد و اگر از یک حد آستانه مشخص شده بیشتر بود به عنوان خروجی ( $y$ ) شبکه در نظر گرفته می‌شود. برای درک بیشتر این فرایند به رابطه (۱) و شکل (۱۵-۵) توجه نمایید.

$$f((w * x) + b) = y \quad \text{رابطه (۱)}$$



شکل (۱۵-۵): فرایند محاسبه یک ورودی و تولید خروجی در ماشین بولتزمن.

شرایطی را در نظر بگیرید که بخواهیم تمامی ورودی‌های لایه آشکار را به یک گره از لایه مخفی ارسال نموده و با هم جمع کنیم، برای این کار هر یک از مقادیر ورودی را با یک وزن مخصوص به خود ضرب کرده و نتیجه آنها را همراه با مقدار یک بایاس جمع می‌نماییم و مقادیر به دست آمده نهایی را به یک تابع فعال ساز ارسال می‌کنیم. در شکل (۱۵-۶) این فرایند را می‌توانید مشاهده نمایید.

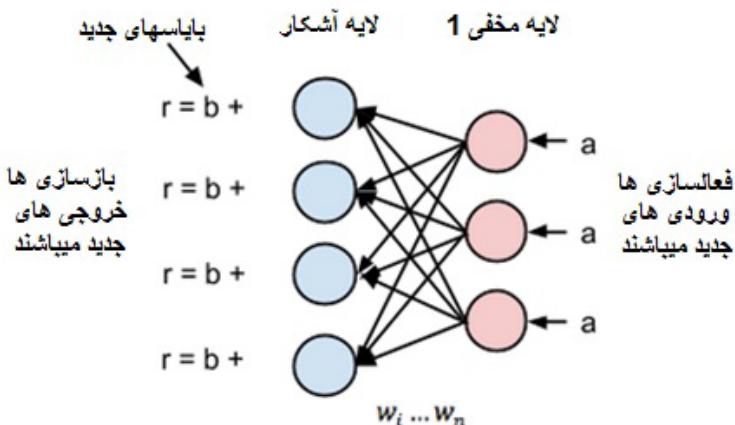


شکل (۱۵-۶): ماشین بولتزمن با دو لایه مخفی و چندین نرون ورودی

به دلیل اینکه ورودی‌های تمامی گره‌های لایه آشکار به تمامی گره‌های لایه مخفی هدایت می‌شوند و این یک مسیر برگشتی نیز می‌باشد بر این اساس ماشین بولتزمن محدود به گراف دو بخشی متقارن<sup>۱</sup> نیز معروف است. توجه به این نکته ضروری است که اگر براساس شکل (۶-۱۵) ماشین بولتزمن محدود در شبکه خود چهار گره در لایه آشکار داشته باشد که آنها به سه گره در لایه مخفی متصل باشند برای هر گره ورودی سه وزن جداگانه وجود دارد که با این شرایط موجود برای این شبکه در مجموع ۱۲ وزن وجود داشته و بردار وزن ما به صورت  $w_1, \dots, w_{12}$  می‌باشد و هر کدام از خروجی‌های لایه مخفی همان‌طور که گفته شد با یک بایاس جمع شده و نتیجه را از طریق یکتابع فعال‌ساز به سمت خروجی هدایت می‌کنند. حال اگر شبکه ما دارای دو لایه مخفی باشد. لایه مخفی اول دارای سه گره و لایه مخفی دوم دارای دو گره خواهد بود و تمامی مقادیر مربوط به گره‌های لایه اول مخفی مقداری را به عنوان نتیجه در خروجی خود تولید می‌کنند که دوباره هر نتیجه به عنوان یک ورودی برای هر دو گره لایه مخفی دوم ارسال خواهد شد.

### ۷-۵-۲- بازسازی<sup>۲</sup> در الگوریتم ماشین بولتزمن

فرض کنید ماشین بولتزمن را برای کار با داده‌های بدون نظرارت در نظر داریم که برای سادگی برای آن تنها یک لایه مخفی در نظر داشته باشیم در این حال این شبکه در مرحله بازسازی، خروجی تابع فعال‌ساز لایه مخفی را به عنوان ورودی به خود گره اختصاص می‌دهد که باید به لایه آشکار انتقال دهد و مقادیر هر گره را با مقادیر وزن یکسان موجود قبلی ضرب کرده و هر مقدار به دست آمده را با بایاس مربوط به گره‌های لایه آشکار جمع می‌کند که بر این اساس خروجی به دست آمده بازسازی تقریبی از ورودی اصلی یا ابتدایی خواهد بود که می‌توانید این فرایند را در شکل (۷-۱۵) مشاهده نمایید.



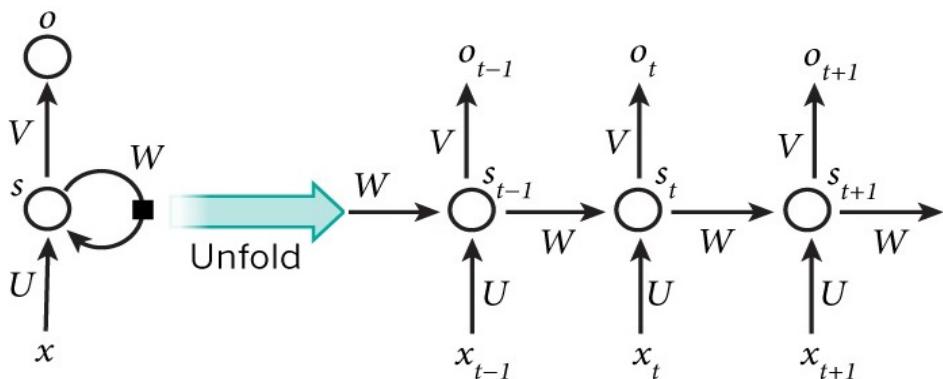
شکل (۷-۱۵): مدل مفهومی بازسازی در ماشین بولتزمن محدود.<sup>۲</sup>

- 
1. Symmetric bipartite graph
  2. Reconstruction
  3. Restricted boltzman

به دلیل آنکه وزن‌های گره‌ها در ماشین بولتزمن محدود به صورت تصادفی مقداردهی اولیه می‌شوند، معمولاً مقادیر تولید شده در بازسازی با مقدار اولیه ورودی تفاوت زیادی دارد که خطای بازسازی، تفاوت بین مقدار تولید شده در مرحله بازسازی با مقدار ورودی اولیه می‌باشد. برای به کمینه رساندن خطای به وجود آمده در این شبکه عمل پس انتشار به صورت مکرر انجام می‌شود تا جایی که خطای به حداقل میزان خود برسد که بعد از آن وزن‌ها ثابت شده و عمل آموزش شبکه به انتهای می‌رسد.

## ۶-۱۵ - شبکه عصبی بازگشتی

شبکه‌های عصبی بازگشتی<sup>۱</sup> در زمینه‌های بسیاری از جمله پردازش زبان طبیعی<sup>۲</sup> عملکرد مناسبی از خود نشان داده است. هدف اصلی ایجاد این شبکه‌ها استفاده از اطلاعاتی است که به صورت متوالی در جریان است و اطلاعات قبلی با بعدی ارتباط دارد مانند داده‌های سری‌های زمانی متوالی. چنانچه در بخش شبکه عصبی ملاحظه نمودید در شبکه‌های عصبی سنتی فرض بر این است که داده‌های ورودی هیچ ارتباطی با هم ندارند و به صورت کاملاً مستقل می‌باشند که در حل بسیاری از مسائل روند مناسبی نمی‌باشد زیرا براساس ماهیت برخی مسائل بدون در نظر گرفتن رابطه داده‌ها نمی‌توان پاسخ مناسبی برای آن به دست آورد. مثلاً اگر در پردازش زبان طبیعی بخواهیم کلمه بعدی را تخمین بزنیم بهتر است که درباره اینکه چه کلماتی در یک جمله قبل از آن کلمه می‌آید اطلاع داشته باشیم. شبکه عصبی بازگشتی را یک الگوریتم بازگشتی می‌گوییم زیرا این الگوریتم برای هر عنصر از یک دنباله عناصر، یک کار مشابهی را انجام می‌دهد که خروجی به دست آمده از پردازش جاری، تحت تأثیر پردازش قبلی بوده است. یکی از شاخصه‌های اصلی این شبکه حافظه موجود آن است که می‌تواند پردازش‌های قدیمی‌تر را نیز به خاطر آورد و در صورت لزوم مورد استفاده قرار دهد. در شکل (۸-۱۵) می‌توانید یک شبکه عصبی بازگشتی را مشاهده نمایید و چنانچه مشخص است شبکه بازگشتی در قسمت چپ تصویر است و حالت باز شده آن در قسمت راست تصویر مشخص است.



شکل (۸-۱۵): شبکه عصبی بازگشتی و نمای بازشده آن.

- 
1. Recurrent neural network
  2. Natural language processing

در شکل (۸-۱۵) مقادیر  $x_t$  مربوط به ورودی برحسب زمان است،  $s_t$  مشخص‌کننده وضعیت مربوط به حافظه شبکه برحسب زمان و  $t$  نیز مشخص‌کننده خروجی شبکه برحسب زمان است.

## ۷-۱-۶- شبکه باور عمیق

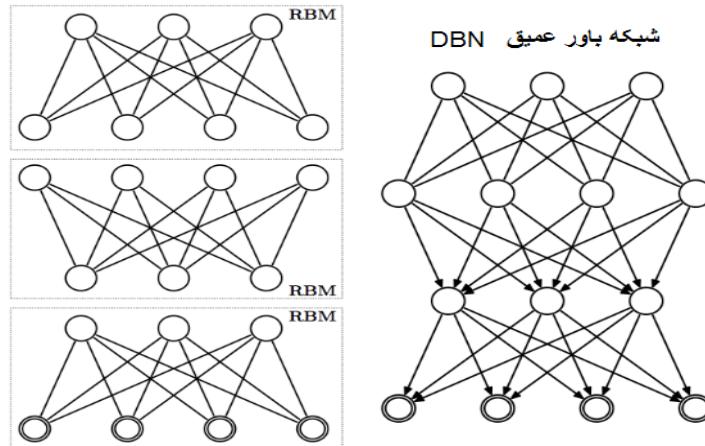
شبکه باور عمیق<sup>۱</sup> یک نوع از شبکه عصبی عمیق مولد است که به صورت گرافی می‌باشد و متشکل از لایه‌های متعدد از متغیرهای پنهان یا واحدهای پنهان می‌باشند که در آن واحدهای پنهان هر لایه دارای هیچ اتصالی به هم نبوده ولی بین لایه‌های مجاور اتصال برقرار است. وقتی شبکه باور عمیق را در حالت بدون ناظر آموزش دهیم قادر خواهد بود ورودی‌هایش را به صورت احتمالی بازسازی کند که پس از آن لایه‌ها مانند یک تشخیص‌دهنده ویژگی از داده‌های ورودی عمل می‌کنند. پس از این مرحله آموزشی، شبکه باور عمیق را می‌توان مجدداً به عنوان یک الگوریتم با ناظر آموزش داد تا بتواند عمل دسته‌بندی را انجام دهد. همچنین این شبکه را می‌توان به عنوان یک ترکیب ساده از شبکه‌های بدون نظارت مانند ماشین بولتزمن یا خودرنز کننده<sup>۲</sup> دانست که هر زیر شبکه در آن به عنوان یک لایه آشکار برای لایه‌های بعدی عمل می‌کند و این موضوع باعث می‌شود تا این الگوریتم با سرعت بالایی عمل یادگیری غیرنظراتی را به صورت لایه‌به‌لایه انجام دهد.

شبکه باور عمیق در کارهای دسته‌بندی نتایج بسیار مناسبی از خود نشان داده است ولی با این حال در برخی از مسائل به دلیل قابلیتی که دارد از آن به عنوان یک روش استخراج ویژگی استفاده می‌شود. به دلیل اهمیت بسیار بالای بازنمایی<sup>۳</sup> در یادگیری ماشین بسیاری از کارهای انجام شده در گسترش الگوریتم‌های یادگیری به سمت یادگیری ویژگی، انتخاب ویژگی و استخراج ویژگی متمایل بوده است. در یادگیری ویژگی سعی می‌شود تا کمک داده ورودی، سیستمی برای استخراج ویژگی ساخته شود تا از خروجی آن برای دسته‌بندی و کاربردهای دیگر استفاده شود. از مزایای شبکه‌های باور عمیق در یادگیری ویژگی آن است که به کمک داده‌های برچسب نخورده می‌تواند ویژگی‌های سطح بالایی از داده‌های آموزشی را استخراج کند و قدرت تمایز بین دسته‌های مختلف در داده‌ها را افزایش دهد.

## ۷-۱-۶-۱- ساختار شبکه باور عمیق

لایه‌های شبکه‌های باور عمیق از ماشین بولتزمن محدود ساخته شده است که هر ماشین بولتزمن محدود، یک مدل احتمالاتی مولد و بدون جهت است که از یک لایه مخفی برای مدل کردن یک توزیع بر روی متغیرهای مشاهده‌بذری خود استفاده می‌کند. به عبارتی با ترکیب چند ماشین بولتزمن محدود، می‌توان شبکه‌های باور عمیق را برای پردازش‌های سلسله مراتبی بوجود آورد که مدل مفهومی آن را در شکل (۹-۱۵) مشاهده می‌نمایید.

- 
1. Deep belief networks
  2. Auto encoders
  3. Representation

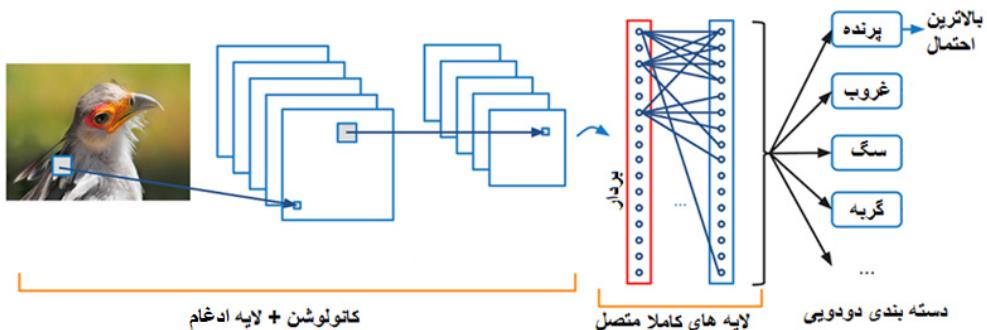


شکل (۹-۱۵): مدل مفهومی چند ماشین بولتزمن محدود با یک شبکه باور عمیق شکل گرفته با آنها.

به دلیل اینکه ساختار اصلی شبکه باور عمیق از ماشین بولتزمن محدود شکل گرفته است پس بر این اساس می‌توان برای بهبود نتایج عملکرد شبکه باور عمیق بر روی ماشین‌های بولتزمن تشکیل‌دهنده آن تمرکز داشت.

## ۱۵-۱- شبکه عصبی کانولوشن

شبکه عصبی مصنوعی کانولوشن یکی از الگوریتم‌های یادگیری عمیق است که نتایج بسیار خوبی در بینایی ماشین و تشخیص اشیاء از خود نشان داده است که به همین دلیل در ادامه مطالب نحوه عملکرد شبکه کانولوشن را در دسته‌بندی تصاویر توضیح داده‌ایم که فرایند کلی دسته‌بندی تصویر و اشیاء را توسط شبکه عصبی کانولوشن در شکل (۱۰-۱۵) مشاهده می‌نمایید.



شکل (۱۰-۱۵): فرایند کلی عملکرد شبکه عصبی کانولوشن برای دسته‌بندی تصویر.

وقتی یک تصویر را به الگوریتم دسته‌بندی‌کننده می‌دهیم هدف ما این است که آن ویژگی‌های درون تصویر توسط الگوریتم مربوطه تحلیل شود و پس از شناخت ماهیت شیء درون آن، تصویر را در دسته متناسب با شیء درون آن قرار دهد. برای ما انسان‌ها توانایی تشخیص اشیاء و دسته‌بندی آنها از بد و تولد به صورت طبیعی آغاز شده و

وقتی به دوران بزرگسالی می‌رسیم بدون هیچ تلاشی که قابل توجه باشد به مهارت بسیار بالایی در انجام این کار دسته‌بندی می‌رسیم. به عبارتی انسان‌ها می‌توانند با راحت‌ترین و سریع‌ترین حالت بدون اینکه حتی دوبار در مورد یک شیء فکر کنند آن را به صورت ذهنی در دسته مربوط به خود قرار داده و تشخیص دهند. حتی وقتی ما در یک چشم‌انداز طبیعی جدید قرار می‌گیریم که تعداد زیادی از اشیاء گوناگون در آن است بدون لحظه‌ای درنگ قادر به تشخیص تمامی الگوها و اشیاء موجود در آن صحنه و قرار دادن برچسب بر روی آنها هستیم.

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08  
49 49 99 40 17 81 16 40 87 17 40 98 43 69 48 04 56 62 00  
81 49 31 73 55 79 14 29 93 73 60 67 53 88 30 03 49 13 34 45  
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 34 91  
22 31 16 71 51 67 43 89 41 92 36 54 22 40 40 28 64 33 13 80  
24 47 32 40 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50  
32 98 81 28 64 23 47 10 26 38 49 47 59 54 70 66 18 38 64 70  
67 26 20 68 02 62 12 20 95 63 94 39 63 08 04 93 64 49 91 21  
24 55 58 05 64 73 99 24 97 17 78 78 94 83 14 88 34 89 43 72  
21 36 29 09 75 00 76 44 20 45 35 14 01 03 33 97 34 31 33 95  
78 17 53 28 22 75 31 67 15 94 03 80 01 62 14 14 09 53 56 92  
14 39 05 42 94 35 31 47 55 58 88 24 01 17 54 24 36 29 85 57  
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58  
88 00 81 65 05 94 47 69 28 73 82 13 86 52 17 77 04 89 55 40  
04 92 08 83 97 35 99 14 07 97 57 32 16 26 26 79 33 27 98 46  
88 34 68 87 57 62 20 72 03 44 33 67 46 55 12 32 63 93 53 49  
04 42 16 73 38 25 39 11 24 94 72 18 00 46 29 32 40 42 76 36  
20 69 36 41 72 30 28 88 34 62 99 69 82 67 59 85 74 04 36 16  
20 73 35 29 78 31 90 03 74 31 49 71 48 86 81 16 23 37 05 34  
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 09 19 47 67
```



(جزی که ما می‌بینیم)

شکل (۱۱-۱۵): نحوه درک یک تصویر توسط انسان و توسط یک کامپیوتر.

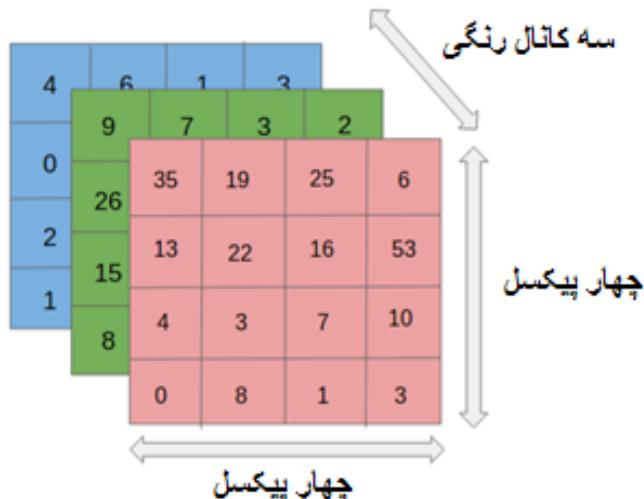
وقتی که تصویر را به کامپیوتری می‌دهیم آن را به صورت آرایه‌ای از مقادیر می‌بیند که براساس وضوح و اندازه تصویر این مقادیر متفاوت است. هر کدام از مقادیر موجود در آرایه مربوط به یک پیکسل از تصویر است که دارای یک مقدار بین ۰ تا ۲۵۵ است که این مقدار براساس شدت روشنایی<sup>۱</sup> آن پیکسل از تصویر مشخص می‌شوند. در حالی که این مقادیر برای ما خیلی قابل درک نیستند اما الگوریتم موجود می‌تواند براساس این مقادیر تصویر مربوطه را تحلیل کرده و در خروجی خود عددی را تولید کند که این عدد نشان‌دهنده احتمال تعلق آن تصویر به یک دسته مشخص شده است. آنچه که ما انتظار داریم تا کامپیوتر انجام دهد این است که بتواند توسط ویژگی‌های خاص یک تصویر تمایزی بین تمامی تصاویر متفاوت قائل شود. به عنوان مثال اگر یک تصویر گربه را به کامپیوتر بدھیم الگوریتم مربوطه باید بتواند با تحلیل ویژگی‌های تصویر تشخیص دهد که تصویر مربوطه متعلق به گربه می‌باشد. شبکه عصبی کانولوشن الهام گرفته از عملکرد بخش خاصی از مغز انسان به نام کورتکس بینایی می‌باشد که وظیفه تحلیل و درک تصاویر ارسالی از چشم‌های انسان را برعهده دارد. کورتکس‌های بینایی منطقه‌های کوچکی از سلول‌ها را دارا می‌باشند که اینها هر کدام به بخش خاصی از میدان دید ما حساس هستند. این ایده براساس آزمایش‌های ارزندهای Hubel و Wiesel در سال ۱۹۶۲ انجام داده‌اند گسترش یافت. آنها شان دادند که بعضی از سلول‌های نرونی خاص در منطقه بینایی قادر به تشخیص لبه‌های تصویر در حالت‌های قرار گرفته مختلف خاص دارند. مثلاً برخی نرون‌ها فقط توانایی تشخیص لبه‌های افقی را دارند و برخی فقط می‌توانند لبه‌های عمودی را تشخیص و بازنمایی کنند که بر این اساس ایده‌ای شکل گرفت که هر جزء از اجزای خاص درون یک الگوریتم برای انجام یک عمل خاص تدارک دیده شود که این اساس عملکرد شبکه کانولوشن می‌باشد.

### ۱-۱-۱۵- ساختار کلی شبکه عصبی کانولوشن

در این شبکه مانند شبکه عصبی پرسپترون چند لایه، داده ورودی از طریق لایه ورودی وارد شبکه شده و پس از پردازش در لایه‌های مخفی به خروجی ارسال می‌شوند. لایه‌ها به هم متصل هستند اما هیچ اتصالی بین نرون‌های موجود در یک لایه مخفی نیست. قسمت‌های اصلی این شبکه شامل لایه کانولوشن<sup>۱</sup>، لایه ادغام<sup>۲</sup>، لایه تماماً متصل<sup>۳</sup> می‌باشد که وظیفه اصلی پردازش داده را به عهده دارند.

### ۱-۱-۱۶- ورودی شبکه عصبی کانولوشن

اگر ورودی یک شبکه عصبی کانولوشن را یک تصویر در نظر بگیریم تصویر ورودی در قالب آرایه‌ای از مقادیر پیکسل به صورت  $32 \times 32 \times 3$  می‌تواند باشد که گویای این است که هر تصویر ورودی می‌تواند حداقل ۱۰۲۴ پیکسل و یا کمتر، و در ۳ طیف رنگی باشد که لایه کانولوشن تلاش خواهد کرد تا یک آرایه از آن را در لایه مخفی ایجاد کند.



شکل (۱۲-۱۵): یک فیلتر  $4 \times 4$  که بر روی یک تصویر با سه کانال رنگی قرار دارد.

### ۱-۱-۱۷- لایه کانولوشن

یک لایه کانولوشن هر یک از پیکسل‌های تصویر ورودی را به یک نرون اختصاص می‌دهد. برای مثال، برای یک تصویر ورودی با ابعاد  $28 \times 28 \times 3$ ، اگر منطقه دریافتی<sup>۱</sup> آن  $5 \times 5$  باشد، هر نرون در لایه کانولوشن به یک منطقه  $5 \times 5 \times 3$  متصل خواهد بود.

1. Convolution layer
2. Pooling layer
3. Fully connected
4. Receptive field

### ۴-۱-۱۵ - تصحیح واحد خطی<sup>۱</sup> (ReLU)

این مرحله مربوط به اعمال یکتابع فعال‌ساز است که به نرون‌های خروجی شبکه کانولوشن اعمال می‌شود به عبارتی در هر مرحله از شبکه عصبی کانولوشن اگر نیاز به تولید خروجی باشد از این تابع استفاده می‌شود که این تابع به صورت  $\max(0, x)$  می‌باشد.

متأسفانه این تابع برای عمل پس انتشار خط‌نمی‌تواند خیلی مناسب عمل کند و برای این رابطه (۲) را به عنوان حالت بهینه شده آن می‌توان استفاده کرد.

$$f(x) = \ln(1 + e^x) \quad \text{رابطه (۲)}$$

با ترکیب تابع رابطه (۲) با تابع سیگموئید یک تابع ترکیبی چنانچه در رابطه (۳) مشاهده می‌نمایید به دست می‌آید که عملکرد بهتری دارد.

$$f'(x) = \frac{d(\ln(1+e^x))}{dx} \quad \text{رابطه (۳)}$$

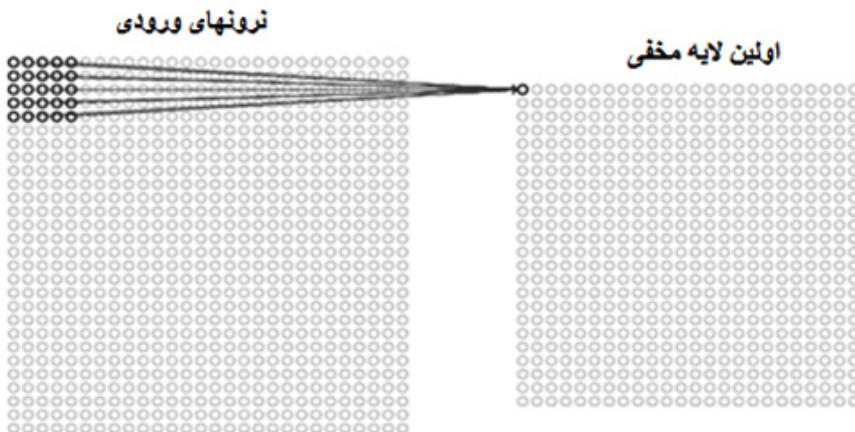
### ۴-۱-۱۶ - لایه ادغام

این لایه درست بعد از لایه کانولوشن قرار می‌گیرد و یکی از ابزارهای اصلی برای کاهش ابعاد مقادیر ورودی از نظر تعداد پیکسل‌ها و ارسال آن برای لایه کانولوشن بعدی است. عملیاتی که در این لایه انجام می‌شود معمولاً به عملیات زیرنمونه‌گیری نیز معروف است که باعث پایین آمدن حجم مجموعه داده‌های ورودی می‌شود و بر این اساس مقداری از اطلاعات مفید را نیز از دست می‌دهد. به هر حال این از دست دادن حجم، فوایدی دارد که بسیار در کارایی الگوریتم تأثیر مثبت دارد. برخی از فواید زیرنمونه‌گیری عبارت است از:

۱. از بیش‌برازش جلوگیری می‌کند.

۲. باعث کاهش حجم محاسباتی برای لایه بعدی می‌شود.

به طور کلی می‌توان گفت که این لایه براساس یک فیلتر با اندازه مشخص که می‌تواند در اندازه‌های مختلف مانند یک فیلتر  $5 \times 5$  باشد، مقادیر مربوط به پیکسل‌های موجود را پوشش می‌دهد و با انجام یک محاسبه ریاضی، تمامی مقادیر درون این فیلتر را به یک عدد واحد تبدیل می‌کند. این فیلتر در هر مرحله یک واحد تغییر مکان داده و مقادیر جدید را به دست می‌آورد که بر این اساس تمامی پیکسل‌های جدید را برای لایه بعدی تولید می‌کند و چنانچه در شکل (۱۳-۱۵) مشاهده می‌نمایید لایه ایجاد شده با عمل زیرنمونه‌گیری، چکیده‌ای از لایه ورودی است و تعداد پیکسل‌های کمتری دارد.



شکل (۱۳-۱۵): تصویر یک فیلتر  $5 \times 5$ .

#### ۶-۸-۶- لایه تماماً متصل

این لایه همان‌طور که از نام آن مشخص است به صورت کامل به خروجی لایه قبلی متصل است. در اصل قبل از خروجی نهایی شبکه عصبی کانولوشن قرار دارد و لایه ادغام را به خروجی متصل می‌کند. این لایه نتیجه پردازش کل شبکه را به صورت بردارهایی از مقادیر در خود نگهداری می‌کند که براساس مقادیر درون این بردارها خروجی مورد انتظار تولید می‌شود.

به صورت کلی شبکه عصبی کانولوشن تصویر ورودی اصلی را لایه‌لایه پردازش می‌کند و در این فرایند پیکسل‌های ورودی را به امتیازهای کلاس نهایی<sup>۱</sup> تبدیل می‌کند. توجه داشته باشید که برخی از لایه‌ها دارای پارامترهایی می‌باشند و برخی از لایه‌ها فاقد آنها. مثلاً اگرچه برای لایه ادغام هیچ پارامتری وجود ندارد ولی لایه کانولوشن تنها بر روی تبدیل داده ورودی به یک داده خروجی مورد نیاز عمل نمی‌کند بلکه دارای پارامترهایی مانند وزن‌ها و بایاس‌ها نیز می‌باشند و با گرادیان نزولی اجرا می‌شود که بتواند الگوریتم را به حالت بهینه برساند.

در این بخش مقدمه‌ای از روش‌های مختلف ایجاد شبکه‌های عمیق را معرفی نمودیم که هرکدام از این روش‌ها بر حسب شرایط مختلف مسئله می‌توانند مورد استفاده قرار گیرند و یا حتی با ترکیب چند روش مختلف می‌توان یک الگوریتم بهینه خاصی برای حل مسائل پیچیده‌تر ایجاد کرد.



۱۹  
فصل

## روش‌هایی جهت افزایش کارایی الگوریتم‌های یادگیری ماشین

## ۱۶- روش‌هایی جهت افزایش کارایی الگوریتم‌های یادگیری ماشین

در این فصل به توجه به اینکه کارایی الگوریتم‌های یادگیری ماشین وابسته به کارایی پیش‌بینی کننده یا کلاس‌بندها است و نیز زمان اجرا نیز نقش مهمی در کارایی دارد این دو موضوع به صورت مختصر ارائه می‌گردد.

۱. چگونه کارایی یک مدل الگوریتم یادگیری ماشین با جستجو کردن مجموعه بهینه‌ای از شرایط یادگیری می‌تواند تنظیم شود.

۲. روش‌های ترکیب مدل‌ها به گروه‌هایی که بتواند مسئله را با کارایی بهتری حل کند.

۳. معروفی روش‌های نوین جهت افزایش کارایی الگوریتم‌های یادگیری ماشین.

یکی از روش‌های بهبود کارایی الگوریتم یادگیری ماشین می‌تواند براساس روش‌های brute-force (بررسی همه حالت‌های مدل یا پارامترهای آن) باشد.

بدلیل دشواری جستجوی همه راه‌های ممکن می‌توان از برنامه‌های خودکار در این راستا بهره گرفت. یکی از روش‌های تنظیم مؤثر پارامترهای مدل است. به عنوان مثال در شبکه‌های عصبی با تنظیم تعداد گره‌ها و تعداد لایه‌های مخفی و یا در ماشین بردار پشتیبان با آزمون توابع هسته مختلف می‌توان بهبودهایی به دست آورد. در برخی از الگوریتم‌های یادگیری ماشین تعداد پارامترهای قابل تنظیم ممکن است خیلی زیاد باشند و یک روش سیستماتیک می‌تواند راحت‌تر منجر به بهبود کارایی شود.

تنظیم خودکار پارامترها نیاز به پاسخ دادن به سؤال‌های زیر را دارد:

۱. چه نوع مدلی از یادگیری ماشین جهت آزمون می‌تواند انتخاب گردد.

۲. چه پارامترهایی از مدل می‌توانند انتخاب شوند و چه میزان تغییرات در این پارامترها برای رسیدن به جواب بهینه ضروری است.

۳. چه معیارهایی برای ارزیابی بهترین مدل انتخابی مهم‌تر هستند.

همان‌طور که در فصل‌های قبلی کتاب توضیح داده شد، مزیت و محدودیت‌های مدل‌های مختلف کلاس‌بندها و یا پیش‌بینی کننده‌ها مورد مطالعه قرار گرفتند. برخی از روش‌ها مانند ماشین بردار پشتیبان و شبکه‌های عصبی در هر دو مورد کلاس‌بندی و پیش‌بینی کاربرد دارند. همچنین بحث‌هایی در مورد تنظیم پارامترها و تأثیر آنها بر کارایی برخی از روش‌ها (مانند روش‌های تنظیم در فصل ۵) بحث شد، و همچنین در مورد ارزیابی کارایی مدل‌ها در فصل ۱۴ بحث شد و معیارهایی همچون دقت، حساسیت، ROC و ماتریس در هم ریختگی مختصرًا توضیح داده شد.

این معیارها نه همه معیارهای موجود ولی مهم‌ترین آنها جهت ارزیابی هستند و در تنظیم پارامترها این معیارها جهت بهبود کارایی می‌توانند مد نظر قرار گیرند. در برخی از کاربردها اندازه مجموعه داده‌ها ممکن است به قدر کافی جهت آزمون مدل و رسیدن به پارامترهای بهینه جهت افزایش کارایی بزرگ نباشد. پیچیدگی محاسباتی مدل از دیگر پارامترهای ارزیابی است بنابراین جهت طراحی یک مدل کارا مصالحه بین این دو باید صورت گیرد، زیرا معمولاً روش‌های با خروجی دقیق‌تر نیاز به قدرت محاسباتی و حجم حافظه بالاتری دارند.

همچنین جهت آزمون بهتر از روش‌های نمونه‌برداری مجدد گوناگون re-sampling و یا bootstrap برای افزایش حجم داده جهت رسیدن به پارامترهای بهینه مدل و در نهایت بهبود کارایی می‌توان بهره جست. روش نمونه‌برداری bootstrap (که می‌توان آن را به عنوان یک راه حل نسبتاً جایگزین برای CV) در k-fold cross validation هم در نظر گرفت) در عمل شامل روش‌های آماری است که از مجموعه داده اصلی با استفاده از روش‌های متنوعی می‌تواند چندین مجموعه داده جدید تولید کند. این روش برای حالت‌هایی که مجموعه داده اصلی اندازه کوچکی دارد می‌تواند چندین مجموعه جدید (به عنوان مثال با تکرار چندین داده به صورت اتفاقی از مجموعه داده اصلی) جهت آزمون بهتر مدل انتخابی تولید نماید.

برای ایجاد یک مجموعه داده بزرگتر با استفاده از این روش‌ها می‌توان چندین مجموعه داده آزمون و آزمون اتفاقی ایجاد کرد. سپس با اعمال این مجموعه‌های مختلف اتفاقی می‌توان خروجی‌ها را محاسبه و متوسط‌گیری کرد و تخمینی از کارایی‌های آینده مدل به دست آورد. در روش bootstrap نمونه‌های موجود در مجموعه داده‌ها می‌توانند چندین بار در پروسه‌های نمونه‌برداری مورد استفاده قرار گیرند. (برخلاف CV که هر داده از مجموعه داده اصلی تنها یک بار استفاده می‌گردد). این به این معناست که از مجموعه اولیه داده با اندازه  $n$ ، روش bootstrap یک یا چندین مجموعه داده جدید تولید می‌کند آنها هم می‌توانند هر یک اندازه  $n$  داشته باشند ولی برخی از داده‌های هر یک از این مجموعه‌ها تکراری از میان داده‌های مجموعه داده اولیه می‌باشند.

## ۱۹- ۲- بهبود کارایی مدل با روش مجمع<sup>۱</sup>

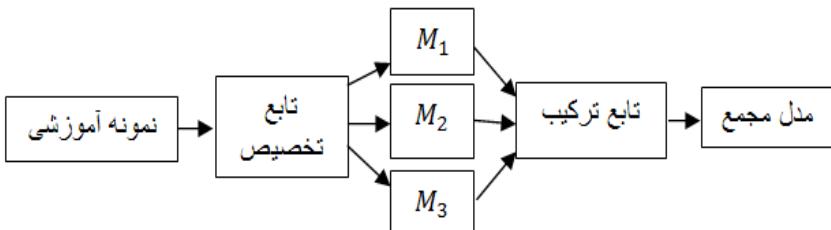
بحث ما در این بخش روی موضوع روش‌های مجمع است که شامل روش‌های متنوعی است که به طور کلی در هر یک از این روش‌ها با استفاده از چندین مدل و ترکیب آنها افزایش کارایی حاصل می‌گردد.

استفاده از کار گروهی در بسیاری از وقایع کارایی را افزایش می‌دهد. یک نمونه را می‌توان مسابقات تلویزیونی نام برد که در آن یک فرد باید پاسخ به سؤالاتی در زمینه‌های گوناگون شامل پزشکی، ورزشی، هنری و غیره را داده تا بتواند جایزه معمولاً بزرگ این مسابقه را نصیب خود کند. معمولاً در این مسابقات شخص می‌تواند از دوستان دیگر خود به صورت محدود در جهت جواب دادن به سؤالات بهره ببرد. طبیعتاً با انتخاب دوستان پزشک، هنرمند و ورزشکار و کمک گرفتن در پاسخ دادن به سؤالات مربوطه از این افراد فرد مذکور می‌تواند موفق‌تر عمل کند.

روش یادگیری مجمع با شبیه‌سازی مثال فوق یک گروه از یادگیرنده‌های متفاوت را به کار گرفته (که در حالت عمومی و ایده‌آل هر یک از این مدل‌ها می‌تواند برجستگی خاصی در یادگیری داشته باشد) و با ترکیب آنها خروجی مطلوب‌تری ایجاد نماید. بنابراین روش‌های مجمع بر این اساس هستند که با ترکیب یادگیرنده‌های ضعیف، یک یادگیرنده قوی تکی می‌توان ایجاد کرد. براساس این اصل ساده، الگوریتم‌های متعددی که در دو مورد زیر می‌توانند متفاوت باشند، ایجاد شده‌اند.

۱. چگونه مدل یادگیرنده ضعیف<sup>۲</sup> برای این هدف انتخاب و پیاده‌سازی شود.

۲. چگونه یادگیرنده‌های انتخاب شده ترکیب شوند تا یک مدل یادگیری تکی و قوی نهایی ایجاد شود.  
معمولًاً روش‌های مجمع از دیاگرام زیر استفاده می‌کنند:



شکل (۱-۱۶): ساختار یک مدل یادگیری مجمع.

در ابتدا داده‌های آزمون برای ساخت تعدادی مدل استفاده می‌شوند. تابع تخصیص<sup>۱</sup> مشخص می‌کند که آیا هر مدل مجموعه کامل آزمون یا بخشی از آن را استفاده کند.

از آنجا که یک مجمع ایده‌آل می‌تواند شامل مجموعه‌های متفاوتی از مدل‌ها باشد ( $M_1$  شبکه‌های عصبی،  $M_2$  ماشین بردار پشتیبان و غیره) تابع تخصیص می‌تواند با تغییر مصنوعی داده‌های ورودی برای آزمون امکان ایجاد یادگیرنده‌های متفاوت، بستگی به نوعشان را فراهم کند. به عنوان مثال این تابع می‌تواند از روش‌های نمونه‌برداری bootstrap برای ساخت مجموعه‌های داده‌های مصنوعی یا زیرمجموعه‌های مختلفی از نمونه‌های آزمون برای هر مدل استفاده کند. به عبارت دیگر، چنانچه مجموع شامل مدل‌های یادگیری متفاوتی باشد، تابع تخصیص ممکن است از داده‌هایی که نسبتاً یکسان می‌باشند جهت آزمون هر یک از مدل‌های یادگیری استفاده کند. بعد از ساخت و آزمون مدل‌ها هر مدل یک عمل مانند پیش‌بینی را انجام می‌دهد که با روشی خروجی مدل‌ها توسط یک تابع ترکیب باید مدیریت شوند. به عنوان مثال تابع ترکیب مدل مجمع ممکن است از رأی اکثریت خروجی (در حالتی که مدل یادگیرنده از نوع کلاس‌بند باشد) و یا از متوسط‌گیری خروجی‌ها (در حالی که یادگیرنده پیش‌بینی کننده باشد) برای تولید خروجی نهایی عمل کند. یا ممکن است از سیاست‌گذاری پیچیده‌تری مانند وزن‌دهی به هر خروجی براساس تقدم کارایی آن استفاده کند. برخی از مدل‌های مجمع حتی از مدل‌های خاصی برای یادگیری تابع ترکیب جهت ایجاد خروجی نهایی استفاده می‌کنند. به عنوان مثال ممکن است حالتی اتفاق بیافتد که خروجی دو مدل یادگیرنده ( $M_1$  و  $M_2$ ) هر دو در وضعیت yes باشند، ولی اگر کلاس واقعی نهایی No باشد، در این حالت تابع ترکیب باید دارای این قابلیت باشد که خروجی‌های  $M_1$  و  $M_2$  را در نظر نگرفته و بجا آن خروجی No را ایجاد کند!

یکی از مزایای استفاده از مجموع‌ها این است که ممکن است زمان کمتری برای طراح جهت رسیدن به نتیجه مطلوب نسبت به طراحی بهترین مدل تکی صرف شود، زیرا طراحی یک مدل تکی و پیچیده دشوارتر از طراحی یک سیستم مجمع که از مدل‌های یادگیرنده ضعیف استفاده می‌کند بوده و کارایی آن نیز غالباً کمتر است. با این

وجود تنها دلیل مناسب بودن راه حل مجمع، دلیل پیشرفت این راه حل نیست، مجمع‌ها همچنین مزایای زیر را نسبت به مدل تکی ارائه می‌دهند.

- تعمیم مناسب‌تر به مسائل آینده: از آنجا که خروجی چندین یادگیرنده جهت خروجی نهایی ترکیب می‌شوند، بایس یکی از پیش‌بینی‌کننده‌ها نمی‌تواند در سیستم نهایی غالب شود. این مسئله شناس بیش‌برازش<sup>1</sup> در مرحله آزمون را کاهش می‌دهد.

### ۱۶-۱- ببود کارایی روی داده‌ای حجمی یا حتی مجموعه داده‌های کوچک

بسیاری از مدل‌ها زمانی که داده‌های با اندازه بزرگ یا داده‌ها با ابعاد یا همان ویژگی‌های زیاد به آنها اعمال می‌شوند با محدودیت‌های پیچیدگی محاسباتی و حافظه مواجه می‌شوند. در این صورت آزمون چندین مدل کوچک نسبت به یک مدل کاملاً بزرگ می‌تواند ارجحیت داشته باشد. به علاوه، موازی کردن یک مجمع غالباً با استفاده از پردازش موازی کار ساده‌تری است. در مقابل مجموعه داده‌های خیلی کوچک با استفاده از روش‌های نمونه‌برداری مجدد مانند bootstrap که غالباً بخشی ذاتی از بسیاری از طرح‌های مجمع‌ها می‌باشند مؤثرترند.

### ۱۶-۲- توانایی سنتز داده از حوزه‌های مجزا

از آنجا که تولید داده‌های عظیم از حوزه‌های متفاوت، روزبه‌روز مهم‌تر و بیشتر می‌شود یک مدل یادگیری تکی به دشواری می‌تواند در این مورد استفاده شود در حالی که مجموعه‌ها که از مشارکت انواع یادگیرنده‌ها بهره می‌برند، در این راستا بسیار مؤثرتر می‌توانند عمل کنند.

### ۱۶-۳- درک بهتر از عملیات یادگیری پیچیده

پدیده‌های دنیای واقعی غالباً بسیار پیچیده‌اند و پیچیدگی‌ها می‌توانند در حد زیاد تعاملی نیز باشند. مدل‌هایی که بتوانند این عملیات را به راهکارهای ساده‌تر تقسیم کنند می‌توانند با دقت بالاتر و هوشمندانه‌تر نسبت به یک مدل سراسری تکی یادگیری را انجام دهند.

### ۱۶-۴- روشن Bagging

یکی از اولین روش‌های مجمع که به صورت گسترده از طرف متخصص مورد استفاده واقع شده است، bagging می‌باشد. در این روش یک تعدادی مجموعه داده آزمون با استفاده از نمونه‌برداری bootstrap از مجموعه داده اصلی و اولیه تولید می‌شوند. سپس هر یک از این مجموعه داده‌ها به هر یک از یادگیرنده‌های موجود در مجمع جهت آموزش اعمال می‌شود. خروجی یادگیرنده‌ها ترکیب شده (رأی‌گیری برای کلاسه‌بندها و یا متوسط‌گیری برای پیش‌بینی‌کننده‌ها) و خروجی نهایی تولید می‌شود. یادگیرنده‌های مورد استفاده در bagging از یک نوع الگوریتم یادگیری استفاده می‌کنند | که هر یک دارای واریانس بالا می‌باشند. به دلیل داشتن واریانس بالا و تغییرات کم در مجموعه داده ورودی می‌تواند منجر به تغییرات زیاد در خروجی هر یک از این یادگیرنده‌های تکی

1. Over fitting

- مورد استفاده در این روش مجمع شود. ولی چند نکته در اینجا قابل ذکر است:
۱. به دلیل استفاده از ترکیب‌کننده مناسب در مجمع واریانس نهایی کاهش می‌یابد و دقت خروجی افزایش می‌یابد.
  ۲. مجموعه داده‌های مورد استفاده از روش bootstrap معمولاً ایجاد شده‌اند، که تغییرات کم در داده‌های آنها وجود دارد، بنابراین تغییرات خروجی ناچیزی در خروجی هر یادگیرنده حاصل می‌شود.
- ولی به هر حال خروجی‌های متفاوت با دقت‌های مختلف توسط هر یادگیرنده تولید می‌شود و با استفاده از ترکیب‌کننده دقت نهایی افزایش می‌یابد. این روش معمولاً از درخت تصمیم برای هر یک از یادگیرنده‌های خود استفاده می‌کند. به عنوان مثال اگر از درخت به عنوان پیش‌بینی‌کننده استفاده گردد علاوه‌بر کاهش واریانس (به دلیل متوسط‌گیری خروجی‌هایی که روی مجموعه داده‌های مختلف ولی با مقادیر نزدیک هم درخت‌ها آزمون دیده‌اند)؛ چنانچه تعداد درخت‌های مورد استفاده هم بزرگ باشد بیش‌برازش نیز رخ نخواهد داد و خطای کلی نیز نسبت به استفاده از یک درخت تکی بزرگ کاهش خواهد داشت.

## ۱۶-۵-۲- روشن Boosting

روش دیگر معروف مجمع، boosting یا تقویت کردن می‌باشد. در این روش کارایی یادگیرنده‌های ضعیف مورد استفاده در مجمع تقویت شده و یادگیرنده‌های قوی‌تر به دست می‌آیند. یادگیرنده یا کلاسه‌بندهای مورد استفاده در این روش دارای ظرفیت پایین می‌باشند مانند کلاسه‌بند خطی که با ترکیب خروجی وزن‌دار این کلاسه‌بندها، بایاس نهایی سیستم مجمع کاهش یافته و ظرفیت افزایش می‌یابد.

این روش به طراح امکان می‌دهد کارایی را تا یک حد آستانه دلخواه به سادگی با افزودن یادگیرنده‌های ضعیف بیشتر افزایش دهد. با استفاده از این امکان بالقوه واضح boosting به عنوان یکی از مهم‌ترین ابداعات در زمینه یادگیری ماشین شمرده می‌شوند.

مشابه با bagging از مجمع مدل‌های آزمون دیده شده روی داده‌های دوباره نمونه‌برداری شده و یک ترکیب‌کننده برای مشخص کردن خروجی نهایی استفاده می‌کند. تفاوت مهم بین این دو در نمونه‌برداری مجدد مجموعه داده‌ها در boosting است. در این روش از bootstrap استفاده نشده و مجموعه داده مورد استفاده هر مدل یک گونه اصلاح شده از مجموعه داده اصلی است، برای آزمون یادگیرنده‌هایی که در سیستم استفاده می‌شوند. در اینجا جهت ترکیب خروجی‌ها روش رأی دادن براساس وزن دادن براساس کارایی هر مدل بجای رأی دادن با وزن مساوی برای هر مدل استفاده می‌شود.

یکی از روش‌های Boosting به نام AdaBoost یا Adaptive Boosting یا Adaboost یا تقویت‌کننده وفقی شهرت دارد که اولین گونه آن در سال ۱۹۹۷ ارائه شد. در این روش یادگیرنده‌های ضعیف به صورت تکراری یک بخش بزرگی از نمونه‌های آزمون از مجموعه داده‌های آزمون که برای کلاسه‌بندی دشوار هستند را به نحوی یاد می‌گیرند که توجه بیشتر (وزن بیشتر) به نمونه‌های غالباً اشتباہ کلاس‌بندی شده می‌دهند.

با شروع آزمون از مجموعه داده‌های وزن داده نشده اولین کلاس‌بند تلاش می‌کند که خروجی را مدل کند. نمونه‌هایی که کلاس‌بند به درستی پیش‌بینی کرده است احتمال ضعیف دارند که در مجموعه داده آزمون هر کلاس‌بند بعدی ظاهر شوند. بالعکس نمونه‌هایی که به دشواری و اشتباهاً کلاس‌بندی شده‌اند، به دفعات بیشتر به کلاس‌بندی اعمال می‌شوند. همان‌گونه که دور بعدی یادگیرنده‌های ضعیف به سیستم افزوده می‌شوند این کلاس‌بندها یا یادگیرنده‌ها، پی‌درپی با داده‌هایی که آزمون آنها دشوار بوده است، آزمون می‌بینند.

این پردازش تا رسیدن به نرخ خطای مطلوب کلی ادامه داشته یا اینکه زمانی که کارایی نهایی بهبود بیشتری نیابد. در این حالت برای هر کلاس‌بند براساس دقت آن روی داده‌های آزمون که به آن اعمال شده است وزن داده می‌شوند. گرچه اصل boosting را تقریباً می‌توان به هر مدلی اعمال کرد ولی این روش بیشتر برای کلاس‌بندها درخت تصمیم متداول است.

## ۶-۳-۱۶- روش Random forest (RF)

روش دیگر RF است که فقط روی مجمع درخت‌های تصمیم استفاده می‌شوند و مانند bagging از روش مجموعه داده‌هایی که با استفاده از Bootstrap تولید شده‌اند کار می‌کند و درخت‌ها براساس این داده‌های مختلف ساخته و رشد می‌کنند. RF می‌تواند روی مجموعه داده‌ای با تعداد ویژگی‌های (بعاد) بسیار بالا کار کند در حالی که این مسئله نفرین ابعاد<sup>۱</sup> ممکن است بسیاری از الگوریتم‌های دیگر یادگیری ماشین را از کار انداخته و یا با افت کارایی مواجه کند. در مقایسه با سایر روش‌ها مجمع RF راحت‌تر مورد استفاده قرار می‌گیرد و به بیش برازش کمتر حساس است و در داده‌های حجمی کارایی بالایی دارد. به دلیل توانایی بالا و همه منظوره بودن، RF سریعاً به یکی از مشهورترین روش‌های یادگیری ماشین تبدیل شده است. این روش با داده‌های نویزی<sup>۲</sup> و گمشده<sup>۳</sup> نیز کارایی دارد. روی ویژگی‌های پیوسته و طبقه‌بندی شده نیز کارایی دارد.

## ۱۶-۳- افزایش کارایی در R از نظر زمان اجرا

زبان R از نظر پردازشی کند و نیز نیاز به حجم حافظه بالایی جهت اجرا دارد. این پدیده در کاربردهای معمولی با سیستم‌های کامپیوتی مدرن قابل لمس نیست ولی در کاربردهایی که اندازه مجموعه داده‌ها به میلیون‌ها رکورده است به ویژه وقتی که هر نمونه از داده دارای ویژگی‌های خیلی زیاد بوده و یا الگوریتم یادگیری مورد استفاده پیچیدگی محاسباتی بسیار بالایی داشته باشد، مشاهده می‌شود. این مسئله از دو جهت در بسته‌های جدید توسعه یافته با استفاده از R مورد توجه قرار گرفته است.

۱. برخی از بسته‌های نرم‌افزاری توانایی جدیدی برای مدیریت مجموعه داده‌های خیلی بزرگ اضافه کرده‌اند. با سریع‌تر کردن پردازش داده‌ها و یا با اجازه دادن به اینکه اندازه داده‌ها از حجم حافظه بیشتر باشد.

1. Curse of dimensionality

2. Noise

3. Missing value

۲. برخی دیگر از بسته‌ها با ایجاد امکان پردازش بر روی کامپیوتر با پردازنده‌های اضافی یا با استفاده از سیستم‌های سختافزاری ویژه و یا با بهینه‌کردن الگوریتم‌های یادگیری برای موارد مربوط به داده‌های عظیم به این هدف نائل شده‌اند که برخی از این بسته‌های نرم‌افزاری در زیر آورده شده‌اند.

#### ۱۶-۴- مدیریت مجموعه داده‌های خیلی بزرگ

اجرای مجموعه داده‌ای خیلی بزرگ می‌تواند منجر شود که اجرای برنامه‌های مبتنی بر R به دشواری کار کنند یا متوقف شوند. به ویژه وقتی حافظه کافی برای ذخیره داده‌ها موجود نباشد. حتی زمانی که کل داده‌ها در حافظه جای بگیرد.

حافظه اضافی RAM جهت انتقال داده از حافظه ثانویه مانند هارد دیسک<sup>۱</sup> به آن مورد نیاز است یعنی حجم حافظه باید بیشتر از حجم مجموعه داده باشد. به علاوه در حالت داده‌های خیلی بزرگ پردازش آنها نیز بسیار زمان‌بر بوده به ویژه اگر یک عملیات پیچیده بخواهد روی این داده‌ها صورت گیرد.

امروزه چندین بسته نرم‌افزاری برای مدیریت برنامه‌ها برای داده‌های عظیم توسعه یافته‌اند. به عنوان مثال بسته نرم‌افزاری data.table یک گونه بهبود یافته از data frame به نام data frame را که اشیاء آن معمولاً خیلی سریع‌تر از data frame برای برخی از عملیات‌ها پردازش می‌شوند عرضه شده است. اشیاء نرم‌افزاری مبتنی بر data frame از نظر اجراء سازگار با برنامه‌های data frame می‌باشند این بسته هنوز مانند data frame از نظر حافظه موجود در سیستم محدودیت دارد.

بسته نرم‌افزاری FF یک راه حل جایگزین به data frame می‌باشد که قادر است مجموعه داده‌هایی بیش از دو میلیارد ردیف را مدیریت کند، حتی اگر حافظه موجود سیستم کافی نباشد. ساختار FFDF دارای مؤلفه‌های فیزیکی است که می‌تواند داده را به صورت مؤثر روی یک مؤلفه مجازی که به مانند یک data frame معمولی R کار می‌کند، ذخیره کند. با این قابلیت که به صورت شفاف می‌تواند به داده ذخیره شده روی مؤلفه فیزیکی دسترسی داشته باشد، می‌توان تصور کرد که یک شیء FFDF به عنوان نگاشتی از نقاط به هر محل روی دیسک عمل می‌کند.

اشکال ساختار داده‌های FFDF این است که ذاتاً با اکثر توابع R سازگار نیستند یعنی داده‌ها باید در قطعه‌های کوچک پردازش و نتایج بعداً ترکیب شوند. مزیت قطعه‌بندی کردن داده‌ها این است که عملیات می‌توانند روی چندین پردازنده با روش‌های مختلف پردازش موازی همزمان اجرا شوند.

#### ۱۶-۵- یادگیری سریع‌تر با پردازش موازی<sup>۲</sup>

با استفاده از روش‌های پردازش موازی حتی چنانچه کامپیوتر دارای یک پردازنده باشد ولی با توجه به اینکه، هر پردازنده دارای چند هسته می‌باشد، امکان پردازش موازی روی پردازه‌ها را می‌دهد.

1. Hard disk  
2. Parallel processing

سیستم‌های کامپیوتی خوشه‌ای نیز می‌توانند قابلیت پردازش موازی ارائه دهند. زمانی که بتوان عملیات پردازشی را به صورت غیروابسته به هم تبدیل کرد پردازش موازی می‌تواند خیلی مؤثرتر عمل کند. به عنوان مثال در حین اجرای k-fold-cv وقتی نمونه‌ها تهیه شدن هر یک از این k عملیات مستقل بوده و می‌تواند روی یک گره از خوشه اجرا شده و نیاز به نتایج یکدیگر حین پردازش نهایی ندارند. بنابراین سرعت به خوبی می‌تواند افزایش یابد.

## ۱۶-۶- روش اندازه‌گیری زمان اجرا

برای درک افزایش سرعت پردازش برنامه‌ها به زبان R تابعی به نام system.time() وجود دارد که می‌تواند زمان اجرای برنامه را اندازه‌گیری نماید. به عنوان مثال در دستور زیر اجرای این تابع مشخص می‌کند که ۰.۱۶ ثانیه برای تولید یک میلیون عدد اتفاقی زمان کامپیوت صرف شده است.

```
> system.time(rnorm(1000000))
 user  system elapsed
 0.16   0.00   0.16
```

این تابع می‌تواند در زمانی که امکانات سخت‌افزاری یا بهبود در توسعه بسته R جهت افزایش سرعت داده شده است زمان اجرای جدید را محاسبه نماید.

بسته foreach شاید ساده‌ترین راه برای اجرا به صورت موازی کردن باشد به ویژه زمانی که R روی ویندوز پیاده‌سازی و اجرا خواهد شد.

بسته multicore امکان پردازش موازی روی یک کامپیوتی که دارای چندین پردازنده یا چندین هسته است را فراهم می‌نماید.

بسته snow امکان پردازش روی چند پردازنده، چند هسته‌ای و نیز چند کامپیوتی را فراهم می‌کند.

## ۱۶-۷- پردازش موازی در محاسبات ابری<sup>۱</sup>

مدل برنامه‌نویسی Mapreduce ارائه شده توسط google روشی برای پردازش داده‌ها روی کامپیوت‌های خوشه‌ای شبکه‌ای می‌باشد که از دو مرحله map و reduce تشکیل شده است:

۱. در مرحله map مسئله به پردازنده‌ای کوچک که قابلیت توزیع شدن بر روی کامپیوت‌های خوشه‌ای را دارد تقسیم می‌شوند.

۲. در مرحله reduce نتایج پردازش قطعه‌های کوچک جمع‌آوری می‌شوند تا براساس آنها پردازش نهایی برنامه اصلی به دست آید.

یک روش جایگزین به نرم‌افزار اختصاصی Mapreduce گوگل، نرم‌افزار منبع باز معروف به Apache hadoop است. نرم‌افزار hadoop از مفهوم mapreduce استفاده کرده و امکان اضافه‌ای جهت توزیع کردن فایل‌ها را برای ذخیره کردن حجم عظیمی از داده‌ها روی کامپیوت‌های یک سیستم خوشه‌ای فراهم می‌کند.

چندین پروژه R وجود دارند که امکان انجام پردازش با روش hadoop را به وجود آورده‌اند که یکی از آنها RHipe است که همان ایده تقسیم و ترکیب مجدد موجود در hadoop را مدیریت می‌کند.

پروژه Rhadoop نیز امکان واسط بین R و Hadoop را فراهم می‌کند. این پروژه بسته‌ای با نام rmr تولید کرده است که جهت یک راه حل ساده برای توسعه‌دهندگان R برای نوشتن برنامه‌های مبتنی بر Mapreduce ارائه داده‌اند. بسته‌های دیگر Rhadoop امکان دسترسی توابع R به داده‌های ذخیره شده مبتنی بر hadoop را فراهم می‌کنند.

## ۱۶-۱- واحد پردازش گرافیکی<sup>۱</sup> (GPU)

بسته gpuTools چندینتابع R مانند عملیات ماتریسی R، خوشه‌بندی، رگرسیون را با استفاده از زبان CUDA در بستر GPU شرکت Nvidia پیاده‌سازی کرده است.

## ۱۶-۲- پیاده‌سازی مؤثرتر الگوریتم یادگیری ماشین با R

برخی از الگوریتم‌های یادگیری ماشین قابلیت اجرای موازی را ذاتاً دارند به عنوان مثال در مجمع امکان پردازش موازی یادگیرنده‌ها وجود دارد. از طرفی برخی از مدل‌ها نیاز به اعمال تغییراتی روی داده یا الگوریتم یا طراحی مجدد جهت پردازش موازی دارند که چند نمونه در زیر آورده شده است.

بسته bigr اجرای عملیات یادگیری RF را برای کلاس‌بندی یا رگرسیون روی داده‌های خیلی بزرگ که در حافظه جا نمی‌گیرند با استفاده از اشیاء bigmemory مدیریت می‌کند. این بسته امکان پردازش موازی سریع‌تر با استفاده از بسته bigmemory را فراهم می‌کند. در مدل RF درخت‌ها می‌توانند به صورت موازی رشد کنند. همچنین درخت‌ها<sup>۲</sup> می‌توانند به جنگل<sup>۳</sup> افروده و یا با جنگل‌های دیگری ادغام شوند.

در سال‌های آینده، جالب خواهد بود که چگونه دیدگاه بشر تغییر می‌کند از آنجا که مسیر بین یادگیری ماشین و یادگیری انسان کم‌رنگ‌تر می‌شوند.

کلام آخر اینکه همان‌گونه که ما امروزه از کامپیوترها برای انجام بسیاری از عملیات‌هایی که انسان‌ها به راحتی نمی‌توانند انجام دهنده بهره می‌بریم. احتمالاً روزی خواهد رسید که کامپیوترها انسان‌ها را برای انجام کارها استخدام نمایند و این با پیشرفت و توسعه الگوریتم‌های یادگیری ماشین احتمالاً امکان‌پذیرتر خواهد بود. یعنی احتمالاً در آینده انسان در اختیار کامپیوتر خواهد بود!

1. Graphical processing unit

2. Tree

3. Forest

۱  
پیوست

## مقدمه‌ای بر جبر خطی (مبانی عملیات ماتریس‌ها)

## مقدمه‌ای بر جبر خطی (مبانی عملیات ماتریس‌ها)

جبر خطی<sup>۱</sup> شاخه‌ای از ریاضیات است که کانون توجه آن فضاهای برداری، ماتریس‌ها، تبدیلات خطی و دستگاه‌های معادلات خطی می‌باشد و همچنین یکی از ابزارهای اساسی در حل معادلات مربوط به الگوریتم‌های یادگیری ماشین است. در این بخش به صورت مقدماتی جبر خطی را معرفی خواهیم کرد که بر این اساس محاسبات ارائه شده در فصل‌های بعدی قابل درک باشند و اگر با این مباحث آشنایی باشید در این صورت می‌توانید بدون مطالعه این فصل به قسمت‌های دیگر این کتاب مراجعه کنید.

### تعريف بردار<sup>۲</sup>

بردار یک ماتریس  $n \times 1$  که  $n$  سطر و ۱ ستون دارد. مثال زیر نشان‌دهنده یک بردار است که در آن  $n=4$  می‌باشد به عبارتی این بردار دارای چهار سطر است و بردارها همیشه یک ستون را دارا می‌باشند.

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \quad (\text{مثال})$$

$$y_4 = 178 \quad y_3 = 315 \quad y_2 = 232 \quad y_1 = 460$$

### تعريف ماتریس<sup>۳</sup>

یک ماتریس آرایه‌ای از اعداد می‌باشد که در درون دو برآکت نوشته می‌شوند و عناصر آن در چند سطر و چند ستون قرار می‌گیرند و به عبارتی و همچنین می‌توان گفت یک ماتریس مشتمل از چند بردار است.

نکته: در ماتریس شروع شماره هر بردار از اندیس یک می‌باشد، ولی ممکن است که در بعضی از برنامه‌ها اولین عنصر بردار یا اولین عناصر آرایه را با صفر شروع کنند.

$$\begin{bmatrix} 14 & 8 \\ 15 & 100 \\ 2 & 12 \end{bmatrix} \quad \text{یا} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad (\text{مثال})$$

ماتریس نشان‌دهنده یک ماتریس با ۲ سطر و ۳ ستون است و همچنین برای

$$\begin{bmatrix} 8 & 5 \\ 2 & 9 \\ 3 & 50 \end{bmatrix} \quad (\text{مثال}) \quad \Leftrightarrow IR^{3 \times 2}$$

- 
1. Linear algebra
  2. Vector
  3. Matrix

مثال) فرض کنید ماتریسی مانند  $A = \begin{bmatrix} 402 & 201 \\ 971 & 821 \\ 849 & 1437 \\ 247 & 1358 \end{bmatrix}$  داریم که  $A_{ij}$  نشان‌دهنده هر کدام از عناصر آرایه

می‌باشد و  $A_i$  به عنصر  $i$ ام در سطر و  $A_j$  به عنصر  $j$ ام در ستون اشاره دارد یعنی در ماتریس  $A$  عدد ۲۰۱ در محل  $A_{12}$  می‌باشد و همچنین روابط زیر را خواهیم داشت.

$$A_{41} = 247, A_{32} = 1437, A_{11} = 402, A_{12} = 201$$

### جمع و تفریق دو ماتریس

برای جمع یا تفریق دو ماتریس باید هر دو از نظر سطر و ستون مانند هم باشند و در غیر این صورت این عمل امکان‌پذیر نیست. برای درک بهتر به دو مثال زیر توجه نمایید.

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix}_{3 \times 2} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{bmatrix}_{3 \times 2} = \begin{bmatrix} 5 & 0.5 \\ 4 & 10 \\ 3 & 2 \end{bmatrix}_{3 \times 2} \quad (\text{الف})$$

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix}_{3 \times 2} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \end{bmatrix}_{2 \times 2} = \text{خطا} \quad (\text{ب})$$

در مثال الف، دو ماتریس  $2 \times 3$  با هم جمع می‌شوند که این عملیات با موفقیت انجام می‌شود ولی در مثال ب، به دلیل اینکه دو ماتریس از نظر سطر و ستون با هم تفاوت دارند انجام عمل جمع امکان‌پذیر نیست.

### ضرب ماتریس در یک عدد

در برخی محاسبات نیاز است که یک عدد را در یک بردار و یا ماتریس ضرب کنیم که در این حالت تک‌تک عناصرهای ماتریس در عدد ضرب خواهد شد و نتیجه هر ضرب را به صورت یک عنصر از ماتریس نتیجه در نظر می‌گیریم.

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} \times 3 = \begin{bmatrix} 3 & 0 \\ 6 & 15 \\ 9 & 3 \end{bmatrix} \quad \text{یا} \quad 3 \times \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 6 & 15 \\ 9 & 3 \end{bmatrix} \quad (\text{مثال})$$

توجه داشته باشید که اگر ماتریس را در عدد یا عدد را در ماتریس ضرب کنیم نتیجه یکسانی خواهد داشت.

### تقسیم ماتریس در عدد

تقسیم ماتریس مانند عمل ضرب می‌باشد که هر عنصر ماتریس را در عدد تقسیم خواهیم کرد.

$$\begin{bmatrix} 4 & 0 \\ -6 & 3 \end{bmatrix} / 4 = \frac{1}{4} \begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{3}{2} & \frac{3}{4} \end{bmatrix} \quad (\text{مثال})$$

## اولویت عملگرها در روابط جبری

برخی موقع چند ماتریس را نیاز است که درهم ضرب و تقسیم یا جمع و تفریق کنیم که در این حالت اولویت عملگرها ابتدا با ضرب و تقسیم و سپس با جمع و تفریق است.

مثال) به نحوه محاسبه رابطه زیر توجه نمایید:

$$3 \times \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix} / 3 = ?$$

توجه شود که اولویت ضرب و تقسیم نسبت به جمع و تفریق بالاتر است و بر این اساس رابطه بالا در دو مرحله محاسبه می‌شود، در مرحله اول عمل ضرب و تقسیم دو ماتریس کناری انجام شده و بعد در مرحله دوم نتایج آنها با هم جمع و تفریق می‌شوند.

$$\begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix} / 3 = \begin{bmatrix} 1 \\ 0 \\ \frac{2}{3} \end{bmatrix} \quad \text{و} \quad 3 \times \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 12 \\ 6 \end{bmatrix}$$

مرحله اول)

و در ادامه جمع و تفریق را به صورت زیر خواهیم داشت.

$$\begin{bmatrix} 3 \\ 12 \\ 6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ \frac{2}{3} \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 10^{\frac{1}{3}} \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 2.1544 \end{bmatrix}$$

مرحله دوم)

## ضرب ماتریس در بردار

در این بخش به شما نحوه ضرب یک ماتریس در یک بردار را نشان خواهیم داد. برای ضرب یک ماتریس در یک بردار باید تعداد ستون‌های یک ماتریس با تعداد سطرهای برداری که می‌خواهیم ضرب کنیم برابر باشند.

مثال) نحوه ضرب ماتریس  $\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix}$  در بردار  $\begin{bmatrix} 1 \\ 5 \end{bmatrix}$  به صورت زیر می‌باشد.

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \end{bmatrix} = ?$$

$$(1 \times 1) + (3 \times 5) = 16$$

$$(4 \times 1) + (0 \times 5) = 4$$

$$(2 \times 1) + (1 \times 5) = 7$$

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}$$

چنانچه مشاهده فرمودید هر سطر از ماتریس به ترتیب با بردار ضرب شده و جواب آن به صورت یک بردار می‌باشد. حال به شما یک مثال کاربردی از عملیات ماتریس برای حل معادلات رگرسیون نشان خواهیم داد.

## ضرب دو ماتریس

اگر بخواهیم دو ماتریس که دارای تعدادی سطر و ستون می‌باشند را به هم ضرب کنیم باید سطر اول ماتریس اول را در ستون اول ماتریس دوم ضرب کنیم و سطر دوم ماتریس اول را به ستون دوم ماتریس دوم ضرب کنیم و بقیه سطرها و ستون‌های دیگر نیز با همین ترتیب بر هم ضرب می‌شوند.

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \quad (\text{مثال})$$

در این مثال ماتریس اول دارای دو سطر و سه ستون است و ماتریس دوم دو ستون دارد که ما می‌توانیم هرستون را به صورت یک بردار در نظر گرفته و ماتریس اول را در هر کدام از بردارها ضرب کرده و نتیجه ضرب ماتریس اول و دوم را به هم افزوده و جواب را به صورت یک ماتریس بسازیم که در روابط زیر مشاهده می‌نمایید.

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix} = \begin{cases} \begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 11 \\ 9 \end{bmatrix} \\ \begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 14 \end{bmatrix} \end{cases} \Rightarrow \begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}$$

به روش دیگری برای ضرب دو ماتریس توجه نمایید.

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 \times 0 + 3 \times 3 & 1 \times 1 + 3 \times 2 \\ 2 \times 0 + 5 \times 3 & 2 \times 1 + 5 \times 2 \end{bmatrix} = \begin{bmatrix} 9 & 7 \\ 15 & 12 \end{bmatrix} \quad (\text{مثال})$$

همچنین لازم به ذکر است که همان‌طور که چند بردار را به هم افزودیم و یک ماتریس ساختیم، یک ماتریس را می‌توان به چند بردار تفکیک کرد.

## خصوصیات ماتریس‌ها

در ضرب ماتریس‌ها رابطه زیر برقرار است.

$$A \times B \neq B \times A$$

در ضرب دو ماتریس اگر ماتریس اول را با ماتریس دوم جابجا کنیم، نتیجه ضرب دو ماتریس متفاوت خواهد بود که برای این موضوع به مثال زیر توجه نمایید.

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \quad (\text{مثال})$$

$$\begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \end{bmatrix}$$

برای درک بهتر به روابط زیر توجه نمایید:

اگر دو ماتریس به صورت  $A_{m \times n}$  و  $B_{n \times m}$  داشته باشیم،  $A \times B$  به صورت  $n \times n$  و  $B \times A$  به صورت  $m \times m$  خواهد بود و رابطه زیر برقرار خواهد بود.

$$A \times (B \times C) = (A \times B) \times C \quad \text{داریم} \quad A \times B \times C$$

مثال) اگر سه عدد ۳، ۵، ۲ را داشته باشیم که باید بر هم ضرب شوند آن را می‌توان به دو صورت  $(5 \times 3) \times 2$  یا  $2 \times (5 \times 3)$  نوشت.

$$(2 \times 5) \times 3 = 2 \times (5 \times 3) \quad \text{مثال)$$

در ادامه یک نوع ماتریس مهم را نشان خواهیم داد که به آن ماتریس یکانی گویند.  
این ماتریس همیشه به صورت مربعی می‌باشد و قطر اصلی آن عدد یک و بقیه صفر می‌باشد.  
برای ماتریس یکانی داریم:

$$(I_{n \times n}) \Leftrightarrow 1 \times z = z \times 1 = z \quad \text{مثال) } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ و غیره ...}$$

هر ماتریسی در ماتریس یکانی ضرب شود جواب خود همان ماتریس می‌باشد.

$$A \cdot I = I \cdot A = A \Leftrightarrow I_{n \times n} \text{ و } A_{m \times n}$$

در ادامه خواهید دید که چگونه یک ماتریس را به ماتریس معکوس و ترانهاده ماتریس تبدیل می‌کنیم.

### ماتریس معکوس:

اگر یک ماتریس با نام  $A$  داشته باشیم که  $m \times m$  یا مربعی باشد و معکوس آن را نیز داشته باشیم رابطه زیر برقرار است.

$$AA^{-1} = A^{-1}A = I$$

به مثال‌های عددی زیر برای درک مفهومی رابطه بالا توجه نمایید.

$$3 \times 3^{-1} = 112 \times (12^{-1}) = 12 \times \left(\frac{1}{12}\right) = 1$$

به یک مثال دیگر از ماتریس توجه نمایید:

اگر یک ماتریس به صورت  $A = \begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix}$  داشته باشیم، معکوس آن به صورت زیر خواهد بود.

$$A^{-1} = \begin{bmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{bmatrix}$$

و در ادامه خواهیم دید که ضرب یک ماتریس در معکوس خود، یک ماتریس یکانی را به وجود می‌آورد.

$$AXA^{-1} = \begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix} \begin{bmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_{2 \times 2} \quad (\text{مثال})$$

### ترانهاده‌ی یک ماتریس: $A^T$

اگر یک ماتریس با نام  $A$  داشته باشیم و آن را به ترانهاده‌اش تبدیل کنیم آن را به صورت  $A^T$  نشان می‌دهیم، در این عمل جبری تمامی ستون‌های یک ماتریس تبدیل به سطر و یا تمام سطرهای آن به ستون تبدیل می‌شود. برای درک بهتر این موضوع به مثال زیر توجه نمایید.

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 5 & 9 \end{bmatrix} \Leftrightarrow A^T = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix} \quad (\text{مثال})$$

سطر اول ماتریس  $A$  به ستون اول ماتریس  $A^T$  و سطر دوم  $A$  به ستون دوم  $A^T$  تبدیل شده و در ماتریس‌هایی با ابعاد بیشتر به همین شکل ادامه می‌دهیم یعنی سطرهای ماتریس را به ترتیب به جای ستون‌های ماتریس ترانهاده‌ی خودش قرار می‌دهیم.





پیوست

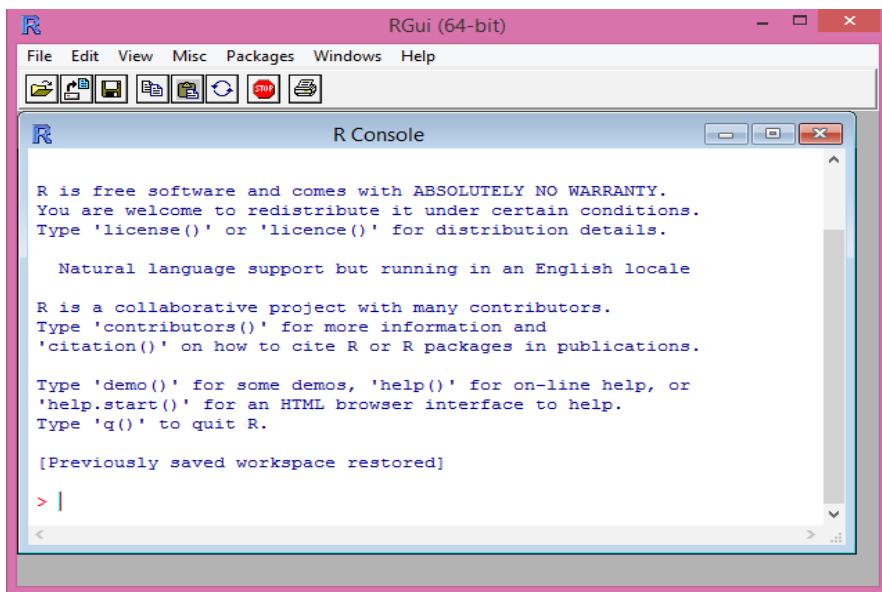
# آموزش زبان برنامه‌نویسی R

## آموزش زبان برنامه‌نویسی R<sup>۱</sup>

در این بخش ما شما را با زبان برنامه‌نویسی R آشنا خواهیم کرد و مطالب به گونه‌ای ارائه شده تا با حداقل دانش برنامه‌نویسی بتوان به صورت خودآموز آن را به راحتی فرا گرفت. مطالب به صورت مقدماتی ارائه شده و مطالب پیشرفته‌تر در بخش‌های دیگر این کتاب با پیاده‌سازی هر الگوریتم مربوط به آن بخش معرفی شده است که شما بتوانید به صورت کاربردی با نحوه عملکرد و چگونگی پیاده‌سازی الگوریتم‌های یادگیری ماشین توسط این زبان آشنا شویید. زبان R در سال ۱۹۹۹ در دانشگاه اولکلند<sup>۲</sup> توسط دو دانشمند با نام‌های راس<sup>۳</sup> و رابرت<sup>۴</sup> ایجاد شده و به صورت رایگان برای استفاده عموم در نظر گرفته شد و به دلیل منبع باز بودن آن می‌توان اقدام به توسعه بخش‌هایی از آن کرد. زبان R توسط تمامی محققین رشته‌های مختلف که به یک زبان کامل برای محاسبات آماری پیشرفته احتیاج دارند، مورد توجه قرار گرفته و در حیطه داده‌کاوی و یادگیری ماشین محبوبیت رو به رشدی را کسب کرده است.

### نصب نرم‌افزار R

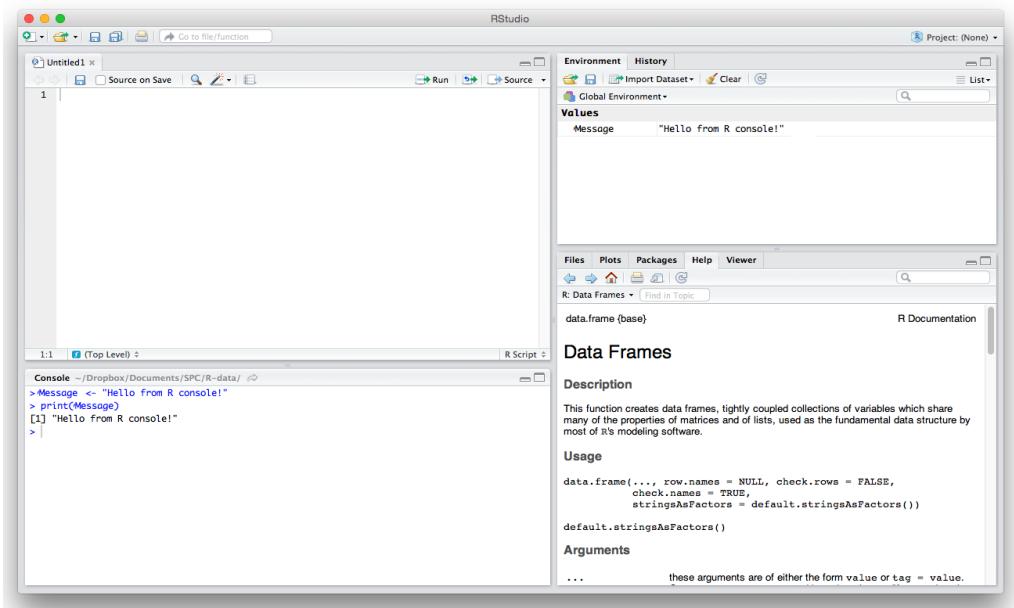
برای نصب R ابتدا باید فایل قابل نصب آن را از سایت منبع آن یعنی <http://www.r-project.org> به صورت رایگان دریافت کنید. محیط برنامه‌نویسی پیش‌فرض R چنانکه در شکل (۱) مشاهده می‌شود بسیار ساده است اما می‌توانید با استفاده از محیط‌های برنامه‌نویسی پیشرفته‌تری که برای این زبان ایجاد شده است مدیریت بهتری برروی کدهای خود داشته باشید. برخی از این محیط‌ها عبارتند از: RStudio و Tinn-R و RWinEdt از:



شکل (۱): رابط کاربری R

- 
1. R
  2. Oakland university
  3. Ross
  4. Robert

در این کتاب شما را با R-Studio آشنا خواهیم کرد که آن را می‌توانید از سایت <http://www.rstudio.com> دانلود و نصب کنید، اما نکته‌ای که باید به آن توجه شود این است که برای نصب این محیط پیشرفته‌تر ابتدا باید محیط پایه‌ای و مقدماتی R را در رایانه مورد نظر نصب کرده باشید. شکل (۲) نمای کلی از محیط R-Studio را نشان می‌دهد.



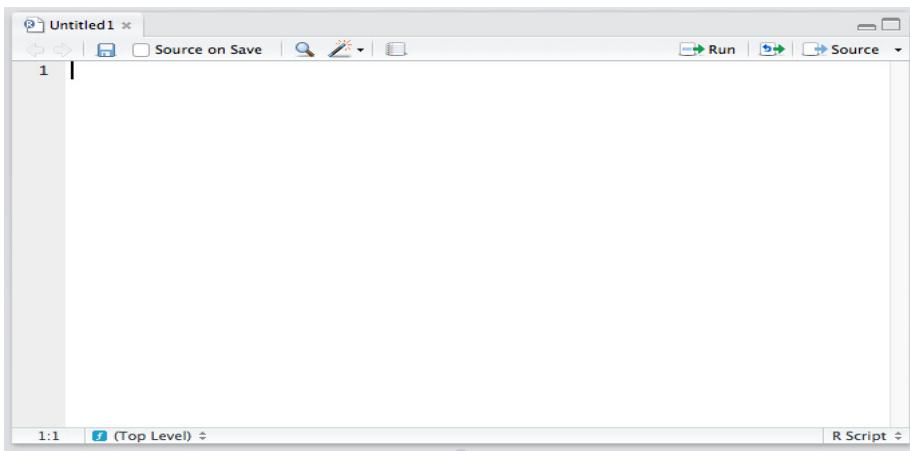
شکل (۲): رابط کاربری و محیط برنامه‌نویسی R-Studio

## R-studio بخش‌های مختلف

محیط R-Studio می‌تواند از طریق مسیر tools-> global option براساس سلیقه کاربر ویرایش شود، ولی ما در این کتاب محیط پیش‌فرض آن را مدنظر قرار داده‌ایم.

### ویرایشگر<sup>۱</sup>

در قسمت بالا و سمت چپ محیط برنامه پنجره‌ای وجود دارد که در آن کدهای R نوشته می‌شود که به صورت شکل (۳) می‌باشد.



شکل (۳): پنجره ویرایشگر خطوط برنامه نوشته شده به زبان R.

در این بخش کدها می‌تواند به صورت یکپارچه نوشته و اجرا شوند و همچنین می‌توان در آن یک تابع، کلاس، یا بسته<sup>۱</sup> ایجاد و با نام مناسب برای استفاده‌های بعدی ذخیره کرد. گزینه‌ی Source on save را بروای این پنجره اگر تیک بزنید تمامی محتوای ورودی به این ویرایشگر به صورت لحظه به لحظه ذخیره می‌شود و از طریق دکمه Run می‌توانید برنامه را اجرا کنید.

## کنسول<sup>۲</sup>

در قسمت پایین و سمت چپ محیط اصلی برنامه R-Studio پنجره کنسول قرار دارد که در شکل (۴) می‌توانید آن را مشاهده نمایید. این پنجره محیطی است که قطعه برنامه‌های کوچک و توابع، ابتدا در این بخش نوشته و آزمایش می‌شوند و سپس در صورت لزوم به ویرایشگر کپی می‌شود. در این پنجره دستورات با فشار دادن دکمه اینتر اجرا شده و نتیجه آن در خط بعد ارائه می‌شود.

```

Console ~/Dropbox/Documents/SPC/R-data/
> Message <- "Hello from R console!"
> print(Message)
[1] "Hello from R console!"
>

```

شکل (۴): کنسول اصلی برنامه R.

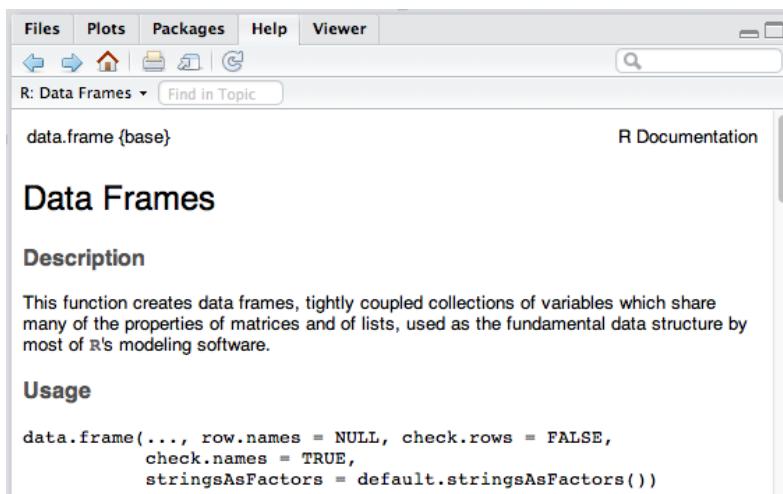
## پنجره محیط و تاریخچه<sup>۱</sup>

همان‌گونه که در شکل (۵) مشاهده می‌کنید در بخش بالا و راست محیط اصلی، پنجره‌ای وجود دارد که دارای دو بخش<sup>۲</sup> و History Environment می‌باشد که بخش محیط، تمامی قسمت‌هایی را که از طریق کنسول تعریف می‌شود را در خود نشان می‌دهد؛ مانند متغیرهای تعریف شده، توابع، مجموعه داده‌ها و غیره و همچنین نکته اصلی در پنجره تاریخچه این است که صرف‌نظر از درستی یا نادرستی دستورات وارد شده، هر خطی که برنامه‌نویس در کنسول نوشته و اجرا کرده را در خود نگهداری کرده و نشان می‌دهد.



شکل (۵): پنجره محیط و تاریخچه.

و یک بخش مهم این ویراستار زبان برنامه‌نویسی یک پنجره با پنج بخش است که آن را می‌توانید در شکل (۶) مشاهده کنید این پنجره در سمت پایین و راست محیط برنامه قرار دارد.



شکل (۶): پنجره مربوط به بخش‌های فایل، نمودارها، بسته‌های برنامه، راهنمای برنامه و بخش مشاهده‌گر.

- 
1. Environment
  2. History
  3. Tab

## فایل<sup>۱</sup>

از این بخش می‌توان فایل‌های موجود درون کامپیوتر را جستجو کرد.

**بسته‌های نرم‌افزاری<sup>۲</sup>:** تمامی بسته‌های برنامه نصب شده در این بخش قرار دارد که می‌توان از اینجا آنها را فعال یا غیرفعال کرد.

**نمودارها<sup>۳</sup>:** نمودارهای ایجاد شده از طریق کدهای نوشته شده در این قسمت به تصویر کشیده می‌شوند.

**راهنما<sup>۴</sup>:** در این قسمت راهنمای دستورات و تمامی جزئیات مربوط به R ارائه می‌شود. به عنوان مثال اگر علامت ? و بلافاصله بعد از آن نام دستوری را در کنسول تایپ کنید توضیحات مربوط به آن دستور در این قسمت به نمایش در می‌آید.

**مشاهده‌گر<sup>۵</sup>:** در این قسمت اطلاعات در قالب فرا متن<sup>۶</sup> به تصویر کشیده می‌شود.

## برنامه‌نویسی با زبان R

در این بخش می‌خواهیم شما را با نحوه نوشتن دستورات زبان R آشنا کنیم. این زبان نیز مانند زبان‌های دیگر دارای ساختار دستوری مربوط به خود می‌باشد که در عین سادگی، بسیار پرقدرت و انعطاف‌پذیر است. در این بخش نحوه نوشتن و اجرای کدهای برنامه در کنسول<sup>۷</sup>، مانند ایجاد متغیر<sup>۸</sup>، محاسبات عددی، کار با بردار، کار با ماتریس و ایجاد لیست‌ها<sup>۹</sup> و مواردی دیگر توضیح داده شده است.

### توضیحات خطوط برنامه

برای اینکه کدهای برنامه قابل فهم‌تر باشند باید برای آنها توضیحاتی را افزود که این کار را با قرار دادن کاراکتر # در ابتدای خطوط برنامه R انجام می‌دهیم. به مثال زیر توجه نمایید.

#this is my first code comment.

```
#####
#####
```

### ایجاد متغیر و اختصاص نام

برای آن که مقداری را به یک نام اختصاص دهیم به صورت زیر عمل می‌کنیم.

variable=5    یا    Variable<-5

1. File
2. Packages
3. Plots
4. Help
5. Viewer
6. Hyper text
7. Console
8. Variable
9. Lists

با هر دو دستور بالا می‌توان متغیری را ایجاد کرد و مقداری را به آن اختصاص داد و با نوشتن نام در کنسول که در اینجا همان variable می‌باشد و زدن کلید اینتر، مقدار درون آن چاپ می‌شود.

&gt;variable

[1] 5

همچنین می‌توان هر کاراکتر یا رشته‌ای را به یک نام اختصاص داد.

&gt;name = 'we use r for machine learning'

&gt;name

[1] "this is my first string"

خط دستور بالا رشته‌ای را به نام name اختصاص می‌دهد و در ردیف دوم با تایپ کلمه name و فشردن کلید اینتر، رشته اختصاص داده شده به نام، چاپ می‌شود.

## محاسبات ریاضی

عملیات ساده ریاضی را می‌توان تنها با نوشتن اعداد و قرار دادن عملگر جمع، ضرب و یا غیره و فشار دادن کلید اینتر انجام داد که بلافاصله جواب در کنسول نوشته می‌شود.

> 5+6	> 5-2	> 4*2	> 4^2	> 4/2
[1] 11	[1] 3	[1] 8	[1] 16	[1] 2

نحوه اختصاص اعداد و همچنین اختصاص یک رابطه ریاضی به یک متغیر به صورت زیر انجام می‌شود.

اول	دوم	سوم
>x=2 >x [1] 2	>y=2+2 >y [1] 4	>z=x+y >z [1] 6

در روابط بالا و قسمت اول عدد ۲ را به متغیر x اختصاص می‌دهد و با نوشتن اسم متغیر و فشار دادن کلید اینتر، عدد درون آن چاپ می‌شود. همچنین در قسمت دوم مقدار جمع دو عدد به متغیر y اختصاص می‌یابد و با نوشتن y و فشار دادن کلید اینتر عدد درون y چاپ می‌شود. در قسمت سوم برای جمع مقادیر درون دو متغیر، آنها را با هم جمع کرده و درون متغیری به نام z ریخته‌ایم که می‌توان آن را یا چاپ کرد و یا در عملیات دیگری استفاده کرد.

به چند مثال دیگر نیز توجه نمایید.

>x=2	>y=4*5	>z=x/2+y*x
>x	>y	>z
[1] 2	[1] 2	[1] 41

## ایجاد دنباله‌ای منظم از داده‌ها

دنباله‌ای از اعداد طبیعی ۱ تا ۲۰ را می‌توان با دستور زیر به وجود آورد که در واقع این دنباله عددی را می‌توان از هر عددی شروع و در هر عددی به پایان رساند مثلاً از ۲۵ شروع و تا ۲۳۵۰ ادامه داد.

```
>n=1:20
>n
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

با دستور زیر دنباله اعداد از منفی شروع می‌شوند.

```
>n=1:-7:-3
>n
[1] -2 -1 0 1 2 3 4
```

در دستور بالا وجود  $-^3$  - یعنی دنباله اعداد ۱ تا ۱۰ را از سه شماره قبل‌تر شروع کن و اگر  $(10-3)$  را داخل پارانتز قرار دهیم، سه عدد آخر دنباله اعداد حذف می‌شود. در مثال زیر دنباله عددی ایجاد شده از ۱ تا ۷ است.

```
>1:(10-3)
[1] 1 2 3 4 5 6 7
```

## تابع‌ها

یک تابع تکه برنامه‌ای می‌باشد که به منظور انجام عمل مشخصی نوشته می‌شود و با نام مناسبی ذخیره می‌شود که هر بار جای نوشتمن مجدد آن تکه برنامه در صورت لزوم تنها نام آن تابع فراخوانی می‌شود. به مانند زبان‌های برنامه‌نویسی دیگر در زبان R توابع زیادی وجود دارد که می‌توان از آنها استفاده کرد و یا اینکه برنامه‌نویس می‌تواند توابع خود را نوشته و اجرا کند. در ادامه این بخش تعدادی از پرکاربردترین تابع‌ها را برای آشنایی بیشتر شما با عملکرد آنها معرفی کرده‌ایم.

## ایجاد تابع

براساس ساختار دستوری زیر می‌توان یک تابع ایجاد کرد و در هر کجای برنامه آن تابع را فراخوانی و استفاده کرد.

```
name.of.function <- function(argument1, argument2) {
  statement
  return(something)
}
```

همان‌طور که در خطوط برنامه بالا مشاهده می‌کنید نام تابع به صورت name.of.function می‌باشد.

عبارت function(argument1, argument2) نیز تابع را ایجاد می‌کند که دربر دارنده آرگومان<sup>۱</sup> نیز می‌باشد.  
عبارت محاسباتی درون تابع که در اصل همان عملکرد اصلی تابع است نیز در statement نوشته می‌شود و عبارت return(something) مقدار محاسبه شده در بخش statement را برمی‌گرداند.

مثال زیر نحوه عملکرد تابعی را نشان می‌دهد که عدد دریافتی را به مربيع آن عدد تبدیل می‌کند.

```
> square.it <- function(x) {
  square <- x * x
  return(square)
}
```

عملکرد تابع بالا به گونه‌ای است که عدد دریافتی در آرگومان تابع را به مربيع آن عدد تبدیل می‌کند. به عبارتی اگر  $x$  یک مقدار عددی باشد در ردیف دوم برنامه، با استفاده از ضرب آن مقدار در خودش، مربيع آن مقدار را ایجاد می‌کند.

پس از ایجاد تابع فقط کافیست مقدار عددی را به عنوان آرگومان به تابعی که ایجاد کرده‌ایم اختصاص دهیم و کلید اینتر<sup>۲</sup> را بزنیم. به مثال زیر توجه نمایید.

```
> square.it(2)
```

[1] 4

## معرفی تعدادی تابع

تابع() مقدار عددی، رشته متنی و یا مقدار درون یک متغیر را در صفحه نمایش چاپ می‌کند. به مثال زیر توجه نمایید.

```
> print(10)
[1] 10
```

- 
1. Argument
  2. Enter

```
>variable<-5
>print(variable)
[1] 5

>name = 'we use r for machine learning'
>print(name)
[1] "we use r for machine learning"
```

تابع () seq() دنباله‌ای از اعداد حقیقی را تولید می‌کند.

```
>seq(1,5,0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

تابع () rep() که دو آرگومان می‌گیرد، اولی یک دنباله اعداد ایجاد می‌کند و دومی تعداد تکرار کل دنباله ایجاد شده را مشخص می‌کند.

```
>rep(1:5,2)
[1] 1 2 3 4 5 1 2 3 4 5
```

این تابع به روش‌های زیر نیز عمل می‌کند.

```
>rep('*',3)
[1] "*" "*" "*"
```

تابع () rnorm() می‌توان با آن دنباله‌ای از اعداد تصادفی ایجاد کرد.

```
>rnorm(5)
[1] -0.27966611 -0.20409732 -0.22561419 0.34702845 0.03236784
```

عدد درون تابع بالا نشان‌دهنده تعداد دنباله اعداد تصادفی می‌باشد که در اینجا ما آن را ۵ قرار داده‌ایم؛ ولی شما می‌توانید هر عددی را انتخاب کنید.

## مسیر جاری<sup>۱</sup>

در زبان R برای خواندن از یک فایل و نوشتن بر روی آن، آدرس پیش‌فرضی وجود دارد که به آن مسیر جاری گویند. با دستور زیر می‌توان مسیر پیش‌فرض را مشاهده کرد.

```
>getwd()
[1] "C:/Users/ali/Documents"
```

برای تغییر مسیر جاری به مسیر دلخواه از دستور زیر استفاده گردد که ما از مسیر "d://R projects" در سیستم خود استفاده کردایم.

```
>setwd("d://R projects")
```

و پس از اجرای مجدد دستور مشاهده دایرکتوری جاری، نتیجه به صورت زیر خواهد بود.

```
>getwd()
[1] "d:/R projects"
```

## خواندن و نوشتن داده از فایل

در زبان R داده‌ها را می‌توان از روی فایل خواند که چند حالت مختلف آن به صورت زیر می‌باشد:

1. خواندن از فایل‌هایی با فرمات txt

```
DataLoad<- read.table("<FileName>.txt", header = TRUE)
```

مثال:

```
>DataLoad<-read.table("d://dataset.txt",header = TRUE)
>dataLoad
[1] X25. X28. X6. X32. X56. X55. X98. X64. X66. X25..1 X98..1 X34. X28..1
<0 rows> (or 0-length row.names)
```

2. خواندن از فایل‌هایی با نوع صفحه گسترده<sup>۱</sup>

```
DataLoad<- read.csv("FILE DIRECTORY", header = TRUE, quote=""", stringsAsFactors=TRUE, strip.white = TRUE)
```

## نوشتن در فایل

دستور پایین محتويات درون یک متغیر را به درون یک فایل می‌نویسد. X همان متغیری است که داده را در خود دارد و آرگومان دوم مربوط به مسیر و نام فایل موردنظر برای ذخیره می‌باشد.

```
write.table(x,"d://dataset.txt")
```

## بسته‌های<sup>۲</sup> نرم‌افزاری

بسته‌های نرم‌افزاری برای انجام عمل خاصی در زبان R برنامه‌نویسی و ایجاد می‌شوند که از توابع، کلاس‌ها و اشیاء تشکیل یافته‌اند. هر بسته یک نام و توضیحات مربوط به کارایی آن را دارد که می‌توان از طریق منبع مستندات<sup>۳</sup> و برنامه‌های زبان R و یا جستجوی ساده درون اینترنت به آن دست یافت. برای نصب بسته خاصی بعد از اطلاع از اینکه کدام بسته و با چه نامی برای کار موردنظر مناسب است، می‌توان آن را از طریق دستور زیر نصب کرد.

```
>install.packages("abc")
```

- 
1. Csv
  2. Package
  3. Cran

abc در دستور فوق اشاره به نام بسته دارد که شما می‌توانید بسته یا بسته‌های دیگری را در آرگومان تابع قرار داده و نصب کنید. توجه شود که به هنگام اجرای این دستور باید به اینترنت متصل باشید زیرا دستور فوق بسته مشخص شده در آرگومان را از درون منبع مربوطه که در اینترنت است یافته و به درون سیستم شما دانلود کرده و نصب می‌کند.

## بارگذاری بسته موجود

با استفاده از تابع زیر و نوشتن نام بسته نرم‌افزاری موجود در آرگومان آن که ما آن را قبلاً بارگیری<sup>۱</sup> کرده‌ایم می‌توان از آن بسته استفاده کرد.  
>library()

## مجموعه داده‌های موجود در برنامه R

در زبان R تعدادی مجموعه داده آماده وجود دارد که می‌توان از آنها استفاده کرد. از طریق تابع () data() می‌توان فهرست کاملی از مجموعه داده‌ها را یافت که ما تعدادی را در اینجا نشان داده‌ایم.

AirPassengers: Monthly Airline Passenger Numbers 1949-1960
BJsales: Sales Data with Leading Indicator
BJsales.lead (BJsales): Sales Data with Leading Indicator
BOD: Biochemical Oxygen Demand
CO2: Carbon Dioxide Uptake in Grass Plants
ChickWeight :Weight versus age of chicks on different diets
DNase: Elisa assay of DNase
EuStockMarkets: Daily Closing

اگر در درون آرگومان تابع نام مجموعه داده مورد نظر نوشته شود می‌توان داده‌های آن مجموعه داده را فراخوانی کرد و بعد از فراخوانی آن با اجرای نام مجموعه داده، محتویات مجموعه داده مورد نظر را مشاهده کرد.

```
>data("airmiles")
>airmiles
Time Series:
Start = 1937
End = 1960
Frequency = 1
[1] 412 480 683 1052 1385 1418 1634 2178 3362 5948 6109 5981 6753 8003
10566 12528 14760 16769 19819 22362 25340
[22] 25343 29269 30514
```

اگر لیستی از مجموعه داده‌ها را به صورت بسته نرم‌افزاری<sup>۱</sup> نصب کنید می‌توانید آن را با استفاده از دستور زیر فراخوانی کنید.

```
>library(gstat)
```

Warning message:

```
package 'gstat' was built under R version 3.2.5
```

اکنون می‌توان با دستور زیر فهرست داده‌های مندرج در درون بسته مورد نظر را مشاهده نمایید. توجه شود که این بسته در بر دارنده مجموعه داده است و با بسته‌هایی که درون خود دارای خطوط برنامه اجرایی هستند متفاوت است.

```
>data(package="gstat")
```

تعدادی از مجموعه داده‌ها برای این بسته به صورت زیر می‌باشد که با استفاده از دستوراتی که گفته شد می‌توان محتویات هر کدام از این مجموعه داده‌ها را مشاهده و استفاده کرد.

TULLNREG, coalash, demstd, fulmar,...

برای مشاهده بخشی از مجموعه داده‌ها می‌توان از دستور زیر استفاده کرد:

```
>head(datasetName,number)
```

آرگومان اول نام مجموعه داده و آرگومان دوم عدد مربوط به تعداد سطری است که می‌خواهیم نمایش داده شود.

مثال:

```
>head(airmiles,4)
```

```
[1] 412 480 683 1052
```

و با استفاده از دستور زیر می‌توان داده‌های انتهایی را مشاهده کرد که قوانین مربوط به آرگومان‌ها مانند دستور فوق می‌باشد.

```
>tail()
```

## بردار

برای ایجاد بردار از تابع () c استفاده می‌شود که می‌تواند دنباله‌ای از اعداد را کنار هم قرار دهد.

## ایجاد یک بردار

```
>c(1,2,3,4,5,6,7,8,9)
```

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

همچنین می‌توان با استفاده از دستور زیر دنباله‌ای از اعداد تکراری را ایجاد کرد که براساس آرگومان‌های تابع `c()` تعداد تکرار هر عدد مشخص می‌شود. به این نکته توجه شود که تعداد آرگومان‌های تابع `c()` باید مساوی تعداد اعدادی باشد که در قسمت ۱:۴ مشخص گردیده است.

```
>rep(1:4,c(3,3,4,2))
[1] 1 1 1 2 2 2 3 3 3 4 4
```

تابع `c(3,3,3,1)` در کد بالا مشخص کننده این است که هر کدام از اعداد تولید شده که توسط آرگومان اول `(1:4)` تولید شده است، باید چند بار تکرار شوند.

## ایجاد بردار و اختصاص آن به یک متغیر

```
>v<-c(2,5,7,9,33,4,20)
```

```
>v
[1] 2 5 7 9 33 4 20
```

در مثال زیر یک بردار ایجاد شده و به یک متغیر با نام `v` اختصاص داده شده است. در ردیف بعد یک بردار دیگر ایجاد شده که متغیر `v2` آخرین آرگومان آن می‌باشد و این بردار جدید به متغیری با نام `v2` اختصاص داده شده است. حال اگر متغیر `v2` را چاپ کنیم تنها بردار اختصاص داده شده در خط اول چاپ می‌شود ولی اگر متغیر `v2` را چاپ کنیم در این حالت بردار اول را در ادامه بردار دوم چاپ می‌کند.

```
>v<-c(7,9,33,4,20)
```

```
>v2<-c(0.50,0.23,0.38,v)
```

```
>v
[1] 7 9 33 4 20
>v2
[1] 0.50 0.23 0.38 7.00 9.00 33.00 4.00 20.00
```

به مثالی دیگر از ایجاد بردار توجه کنید که در آن کلمات، محتوای بردار می‌باشند.

```
>carNames<-c("benz","bmw","porsche")
```

```
>carNames
```

```
[1] "benz"   "bmw"    "porsche"
```

برای مشخص کردن تعداد کلمات درون بردار از تابع زیر استفاده می‌شود:

```
>length(carNames)
[1] 3
```

برای مشخص کردن تعداد حروف هر کلمه درون بردار ازتابع زیر استفاده می‌شود:

```
> nchar(carNames)
```

```
[1] 4 3 7
```

## محاسبات ریاضی بر روی بردارها

اگر برداری را به یک متغیر اختصاص دهیم و آن متغیر را در خودش ضرب کنیم با این کار تک‌تک عناصر بردار در خودش ضرب خواهد شد.

```
>d<-c(10,2,4,3)
```

```
>m=d*d
```

```
>m
```

```
[1] 100 4 16 9
```

با روش زیر می‌توان لگاریتم<sup>1</sup> تک‌تک عناصر بردار را حساب کرد و برای این کار بردار را در آرگومانتابع مربوطه قرار می‌دهیم.

```
>m
```

```
[1] 100 4 16 9
```

```
>log(m)
```

```
[1] 4.605170 1.386294 2.772589 2.197225
```

ریشه اعداد را نیز از طریق تابع زیر می‌توان به دست آورد.

```
>m
```

```
[1] 100 4 16 9
```

```
>sqrt(m)
```

```
[1] 10 2 4 3
```

عمل جمع، منفی، ضرب و تقسیم دو بردار به صورت زیر می‌باشد.

```
>y<-c(2,3,2,5)
```

```
>x<-c(1,2,4,1)
```

```
>x
```

```
[1] 1 2 4 1
```

```
>y
```

```
[1] 2 3 2 5
```

```
>x+y
```

```
[1] 3 5 6 6
```

```
>x-y
```

```
[1] -1 -1 2 -4
```

```
>x*y
[1] 2 6 8 5
>x/y
[1] 0.5000000 0.6666667 2.0000000 0.2000000
```

## ماتریس‌ها

ماتریس در اصل از اجتماع چند بردار به وجود می‌آید و دارای تعدادی سطر و ستون می‌باشد. در ماتریس هم مانند بردار داده‌ها همنوع هستند. برای ایجاد ماتریس به دستور زیر توجه نمایید.

```
>m<-matrix(c(2,5,6,4,7,5),ncol = 3,nrow = 2)
```

```
>m
 [,1] [,2] [,3]
 [1,] 2 6 7
 [2,] 5 4 5
```

تابع matrix() برای ایجاد ماتریس می‌باشد که آرگومان اول آن تابع c() می‌باشد که کار تولید عناصر درون ماتریس را به عهده دارد، آرگومان دوم ncol تعداد ستون‌ها و آرگومان سوم nrow تعداد ردیف‌ها را مشخص می‌کند.

حال برای دستیابی به عناصر ماتریس m ای که ایجاد شده از دستور m[i,j] استفاده می‌کنیم. به مثال زیر توجه نمایید.

```
>m
 [,1] [,2] [,3]
 [1,] 2 6 7
 [2,] 5 4 5
```

برای دستیابی به سطر دوم از ستون سوم ماتریس m، دستور زیر را اجرا می‌کنیم.

```
>m[2,3]
[1] 5
```

با دستورات زیر می‌توان به کلیه مقادیر یک سطر مشخصی از ماتریس دست یافت.

مثال) سطر دوم ماتریس:

```
>m[2,]
[1] 5 4 5
```

با دستورات زیر می‌توان به کلیه مقادیر یک ستون مشخصی از ماتریس دست یافت.

مثال) ستون سوم ماتریس:

```
>m[,3]
```

```
[1] 7 5
```

## روش تبدیل ماتریس به بردار

با استفاده از دستور as.vector() می‌توان ماتریس را به بردار تبدیل کرد.

```
>mat<-matrix(c(1:6),ncol = 3)
```

```
>mat
```

```
[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
```

```
>matvector<-as.vector(mat)
```

```
>matvector
```

```
[1] 1 2 3 4 5 6
```

## ترانهاده ماتریس

ترانهاده ماتریس را با استفاده از تابع () t به دست می‌آوریم. در مثال زیر mat یک ماتریس است که اگر آن را درون تابع () قرار دهیم ترانهاده ماتریس mat را برمی‌گرداند.

```
mat
```

```
[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
```

```
> t(mat)
```

```
[,1] [,2]
[1,] 1 2
[2,] 3 4
[3,] 5 6
```

## ماتریس یکانی

با استفاده از تابع `diag(k)` در صورتی که  $k$  یک عدد باشد می‌توانیم یک ماتریس یکانی  $k \times k$  ایجاد کنیم.

```
> k=5
> diag(k)
 [,1] [,2] [,3] [,4] [,5]
[1,] 1 0 0 0 0
[2,] 0 1 0 0 0
[3,] 0 0 1 0 0
[4,] 0 0 0 1 0
[5,] 0 0 0 0 1
```

## معکوس ماتریس

تابع `solve(A)` معکوس ماتریس  $A$  را می‌دهد به شرطی که  $A$  یک ماتریس مربعی باشد.

```
> mat=matrix(c(1:4),ncol = 2)

> mat
 [,1] [,2]
[1,] 1 3
[2,] 2 4

> solve(mat)
 [,1] [,2]
[1,] -2 1.5
[2,] 1 -0.5
```

## داده‌های چارچوب دار (data frame)

داده‌های چارچوب دار دارای ستون‌هایی با نوع داده‌های مختلف است که کاربرد زیادی در تحلیل داده‌ها دارد. هر ستون از داده‌ها دارای نوع یکسان هستند ولی ستون‌های مختلف می‌تواند دارای نوع داده متفاوتی باشند که طول تمامی بردارهایی که ستون‌ها را تشکیل می‌دهند یکسان است.

برای ایجاد داده‌های چارچوب دار به دستورات زیر توجه نمایید:

```
>x = 11:20
```

```
>y = letters[1:10]
```

```
>z = rep(c('male', 'female'), 5)
```

```
>df = data.frame(id = x, status = y, JENSIYAT = z)
```

```
>df
```

	id	status	JENSIYAT
1	11	a	male
2	12	b	female
3	13	c	male
4	14	d	female
5	15	e	male
6	16	f	female
7	17	g	male
8	18	h	female
9	19	i	male
10	20	j	female

در ردیف اول به تعداد ۱۰ عدد از ۱۱ تا ۲۰ به صورت سری تولید شده و در x ذخیره می‌شود و در ردیف دوم ۱۰ حرف از a تا z تولید و در y ذخیره می‌شود. در ردیف سوم کلمه male و female به ترتیب به تعداد ۵ تکرار و به صورت بردار در z ذخیره می‌شود. در اصل سه بردار با نامهای x y و z ایجاد کرده‌ایم که می‌خواهیم به صورت داده‌های چارچوب‌دار در کنار هم قرار داده و نمایش دهیم. در نهایت در ردیف چهارم از طریق دستور() data.frame() بردارها را کنار هم قرار می‌دهیم به این صورت که در آرگومان تابع مربوطه، سه بردار را مساوی سه اسم دلخواه قرار می‌دهیم که همان اسامی ستون‌ها می‌باشند.

## فاکتورها (factors)

بعضی از متغیرها در زبان R به صورت دسته‌ای مدنظر قرار می‌گیرند که آن را با فاکتور نشان می‌دهند. در حالات مختلف به این متغیرها، متغیرهای مرتب و متغیرهای نامرتب گویند. مثلاً در انواع خاک‌ها که به شنی، ماسه‌ای، رُسی و غیره تفکیک می‌شوند فاکتورهای نامرتب گفته می‌شود و اگر خاک را به ضعیف، قوی و خیلی قوی تفکیک کنیم به اینها فاکتورهای مرتب گویند.

به مثال نامرتب زیر توجه نمایید.

```
>soil.types<-c("clay","loam","sand","loam","clay")
```

```
>soil.types<-factor(soil.types)
```

```
>soil.types
```

```
[1] clay loam sand loam clay
```

```
Levels: clay loam sand
```

همان‌گونه که در خط آخر در قسمت levels مشاهده می‌نمایید به سه نوع خاک اشاره کرده است.

حال به یک مثال از مرتب‌ها توجه نمایید.

```
>soil.degree<-c("weak","average","stronge","weak","stronge")
>soil.degree<-ordered(soil.degree,levels=c("weak","average","stronge"))
>soil.degree
[1] weak average stronge weak stronge
Levels: weak < average < stronge
```

در سطر آخر ترتیب سطوح مختلف را نشان می‌دهد و همچنین می‌توان آن را به صورت عددی نیز نشان داد که می‌توانید در مثال زیر مشاهده نمایید.

```
>soil.degree
[1] weak average stronge weak stronge
Levels: weak < average < stronge
```

```
>soil.numeric<-as.double(soil.degree)
>soil.numeric
[1] 1 2 3 1 3
```

همان‌گونه که در بخش نتیجه برنامه بالا می‌بینید سطح سختی خاک را به صورت عددی نشان داده است مثلاً برای ضعیف مقدار ۱ و برای قوی مقدار ۳ را نشان داده است.

## سری‌های زمانی<sup>۱</sup>

در زبان R برای ایجاد شیء سری زمانی از تابع ts() استفاده می‌شود که دو مؤلفه داده‌ها و تاریخ آنها در آن وجود دارد.

**داده‌ها:** بردار یا ماتریسی از داده‌های عددی می‌باشند که هر ستون یک سری زمانی مجزا را تشکیل می‌دهد.  
**تاریخ:** فواصل مشخص تاریخ داده‌ها می‌باشد.

به مثال زیر توجه نمایید.

```
>my.ts<-ts(matrix(rnorm(40), ncol = 4), start=c(2000), frequency = (10))
>my.ts
Time Series:
Start = c(2000, 1)
End = c(2000, 10)
Frequency = 10
```

	Series 1	Series 2	Series 3	Series 4
2000.0	-0.7959544	0.04913729	-0.2573231	1.06024324
2000.1	-1.3132317	0.44909315	-1.1130598	-0.83623997
2000.2	1.5411079	-0.38512047	0.6120174	1.66359849
2000.3	-0.131785	-0.67738206	-1.4845356	0.05344508
2000.4	-1.4664533	0.42014288	-0.9139451	2.22688956
2000.5	-1.5073629	1.52651412	0.6956207	1.40289627
2000.6	-1.1468381	0.78016616	1.6209827	-0.64619568
2000.7	0.5858550	-2.94297087	-0.9755637	-0.61633378
2000.8	-2.0787991	-0.53713480	2.0263469	0.47775019
2000.9	-0.1582376	1.56290855	-0.9676026	-0.70530537

**لیست**

یک لیست در زبان R برداری متتشکل از چندین شیء است. در مثال پایین سه بردار تشکیل داده شده و در نهایت یک بردار با نام x ایجاد گردیده است که در بر دارنده سه بردار تشکیل شده می‌باشد. براساس توضیحات داده شده در خط چهارم برنامه، متغیر x در بر دارنده یک لیست می‌باشد که با این روش می‌توان از ترکیب بردارهای مختلف لیستی تهیه کرد و به صورت یکجا بر روی آنها کار کرد و همچنین می‌توان به صورت گزینشی برروی بخش خاصی از داده‌های بردارهای مختلف نیز کار کرد.

```
>n = c(20, 13, 15)
>s = c("aa", "bb", "cc", "dd", "ee")
>b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
>x = list(n, s, b, 3)
```

>x

```
[[1]]
[1] 20 13 15
```

```
[[2]]
[1] "aa" "bb" "cc" "dd" "ee"
```

```
[[3]]
[1] TRUE FALSE TRUE FALSE FALSE
```

```
[[4]]
[1] 3
```

## تکه کردن لیست

در این قسمت می‌توان با مشخص کردن اندیس برای متغیری که لیست را درون آن ذخیره کرده‌ایم تنها همان قسمت از جواب را استخراج کنیم که متناسب با اندیس مربوطه در جواب کلی بوده است. مثلاً اندیس ۲ مربوط به حروف می‌باشد.

&gt;x[2]

[[1]]

[1] "aa" "bb" "cc" "dd" "ee"

اگر اندیس را با بردار نشان دهیم می‌توانیم تکه‌های بیشتری را همزمان استخراج کنیم.

&gt;x[c(2, 4)]

[[1]]

[1] "aa" "bb" "cc" "dd" "ee"

[[2]]

[1] 3

همچنین با استفاده از دو براکت که اولی تودرتو می‌باشد می‌توان به ردیف و اندیس مشخصی در لیست اشاره کرد و آن را استخراج کرد. عدد درون دو براکت تودرتو نشان‌دهنده شماره بردار و عدد درون براکت دوم نشان‌دهنده محل درون بردار مورد نظر می‌باشد. به مثال‌های زیر توجه نمایید.

&gt;x[[1]][2]

[1] 13

&gt;x[[3]][5]

[1] FALSE

&gt;x[[2]][5]

[1] "ee"

## تصورسازی داده‌ها:

زبان R امکانات بسیار زیادی برای ایجاد نمودارهای مختلف و تصورسازی داده‌های آماری فراهم نموده است که باید برای حالات مختلف ترسیم داده‌ها از توابع درون خود R-Studio و یا بسته‌های نرم‌افزاری متناسب برای انجام آن کار را دانلود، نصب و اجرا کرد. نمودارهای مختلفی برای به تصویر کشیدن داده‌ها وجود دارند که ما تعدادی از آنها را در این بخش معرفی می‌کنیم.

## نمودار میله‌ای

یکی دیگر از نمودارهایی که رابطه بین سری‌های مختلف را نشان می‌دهد نمودار میله‌ای است. در این نوع نمودار ارتفاع میله‌ها فراوانی یا متغیر اندازه‌گیری شده است: هرچه میله بلندتر مقدار متغیر نیز بیشتر است.

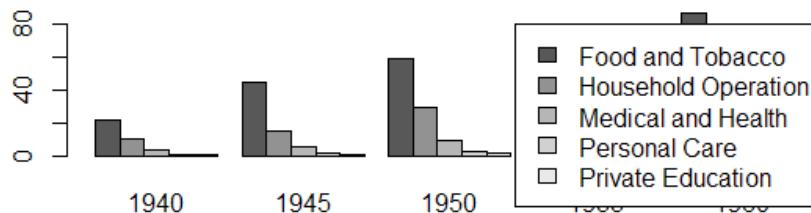
مجموعه داده USPersonalExpenditure مربوط به هزینه زندگی در کشور آمریکا بین سال‌های ۱۹۴۰ تا ۱۹۶۰ است که با تابع data() آن را فراخوانی می‌کنیم.

&gt; data("USPersonalExpenditure")

با استفاده از تابع barplot() نمودار میله‌ای زیر را ترسیم می‌کنیم.

>barplot(USPersonalExpenditure,beside=TRUE,legend.text=TRUE,main="وضعیت هزینه‌ها")

### وضعیت هزینه‌ها



:beside=TRUE سبب می‌شود که مقادیر هر ستون کنار هم رسم شوند.  
:legend.text=TRUE راهنمای تصویر را در بالا و سمت راست اضافه می‌کند.  
"وضعیت هزینه‌ها" main= عنوان اصلی نمودار را مشخص می‌کند.

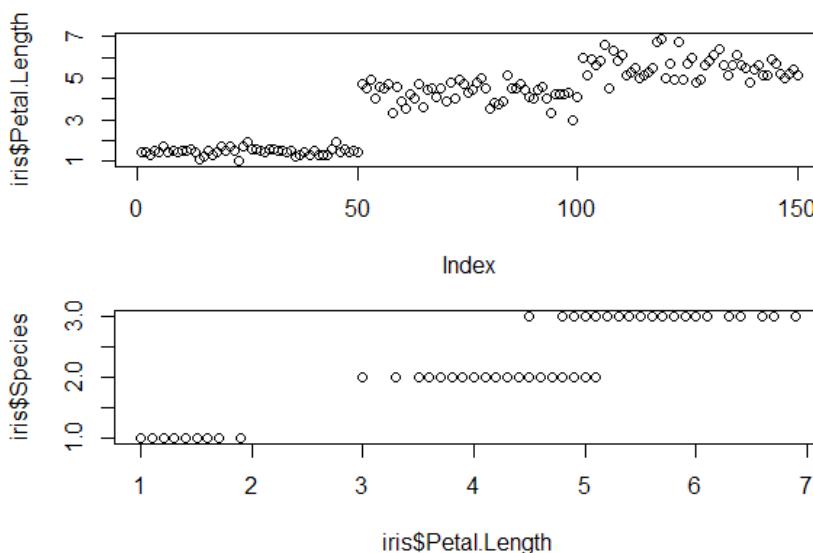
### نمودار پراکندگی

هنگامی که از آمار استفاده می‌کنیم در اغلب مسائل ارتباط بین اندازه‌های مختلف مهم است. برای بررسی این موضوع یکی از رایج‌ترین نمودارها، نمودار پراکندگی است که با تابع plot() رسم می‌شود. دو مدل مختلف از نمودار پراکندگی با دو خط دستور زیر ایجاد گردیده است. خط اول دستور زیر مربوط به نمودار پراکندگی تک متغیره ساده و خط دوم نمودار پراکندگی چند متغیره می‌باشد.

>plot(x=iris\$Petal.Length)

>plot(x=iris\$Petal.Length,y=iris\$Species)

دو شکل زیر اولی مربوط به تک متغیره و دومی مربوط به دو متغیره می‌باشد.



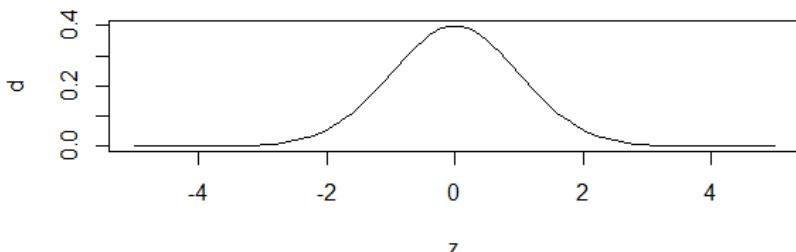
برای رسم نمودار چگالی نرمال استاندارد به مثال زیر توجه نمایید.

ازتابع() برای ایجاد نقاطی با فاصله یکسان بین -۳ و ۳ با فاصله (0.1) که شامل ۶۱ نقطه می‌باشد استفاده می‌کنیم. سپس تابع() را برای محاسبه چگالی نرمال (تراکم) و استاندارد ارزیابی شده در آن نقاط فراخوانی نموده، آن را ترسیم کرده و تیتری را به رنگ آبی کم رنگ به آن می‌افزاییم. توجه داشته باشید که ما می‌توانیم تیتر را در یک فراخوانی جداگانه به طرح فعلی اضافه نماییم.

```
> z <- seq(-5.5,0.1)
> d <- dnorm(z)
> plot(z,d,type="l")
> title("The Standard Normal Density",col.main="cornflowerblue")
```

در زبان برنامه‌نویسی R آرگومان‌های یک تابع را می‌توان براساس موقعیت یا نام مشخص نمود.

### The Standard Normal Density



تابع plot انتظار دارد که دو آرگومان اول، بردارهایی باشند که مختصات x و y نقاطی را که قرار است ترسیم شوند، نشان دهند و در اینجا ما نوع گراف را نیز مشخص نمودیم. از آنجایی که نوع گراف یکی از چندین پارامتر اختیاری می‌باشد (برای مشاهده جزئیات بیشتر، plot را تایپ نمایید)، آن را به صورت "l" type="l" (حرف ال) مشخص کردیم که شما می‌توانید از انواع مختلف دیگر استفاده کنید در اصل ما به جای حالت پیشفرض یعنی ترسیم نقاط گسسته، خواستار به هم پیوستن نقاط برای ایجاد یک خط می‌اشیم. توجه داشته باشید که نرم‌افزار R از علامت تساوی برای تعیین آرگومان‌های نامدار یک تابع استفاده می‌نماید.

تابع title انتظار دارد که یک رشته کلارکتری برای تیتر را به عنوان آرگومان اول دریافت نماید ما همچنین آرگومان اختیاری "col.main="cornflowerblue" را برای تنظیم رنگ تیتر، تعیین نمودیم.

## نمودار هیستوگرام

هیستوگرام نوع خاصی از نمودار میله‌ای است که در آن از مستطیل‌ها برای نشان دادن فراوانی توزیع یک مجموعه از اعداد استفاده می‌شود. هر مستطیل میزان مقادیر x را در هر کدام از پایه‌هاییش نشان می‌دهد و معمولاً تمام

مستطیل‌ها دارای پهنای یکسانی می‌باشند و ارتفاع هر مستطیل به عدد مشاهدات مربوط به بازه مورد نظر بستگی دارد. اگر مستطیل‌ها پهنای متفاوتی داشته باشند آنگاه مساحت آن مستطیل متناسب با مقادیر آن است و ارتفاع مستطیل نشان‌دهنده چگالی داده است.

توسط دستورهای زیر شش نمودار هیستوگرام ترسیم گردیده که پارامترهای هر کدام را به ترتیب مورد توجه قرار دهید.

خط زیر بسته RColorBrewer نصب می‌کند که برای افزودن الگوهای رنگی متنوع به نمودار می‌توان از این بسته استفاده کرد

```
> install.packages('RColorBrewer')
```

خط زیر بسته نصب شده را بارگذاری می‌کند که بتوان از آن استفاده کرد.

```
> library(RColorBrewer)
```

```
> data(USPersonalExpenditure)
```

خط زیر تعدادی از ردیف‌های مجموعه داده را نمایش می‌دهد.

```
> head(USPersonalExpenditure)
```

تعدادی از ردیف‌های مجموعه داده به صورت زیر می‌باشد.

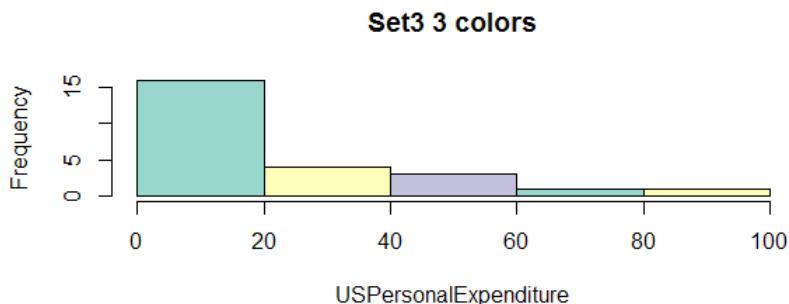
Year	1940	1945	1950	1955	1960
Food and Tobacco	22.200	44.500	59.60	73.2	86.80
Household Operation	10.500	15.500	29.00	36.5	46.20
Medical and Health	3.530	5.760	9.71	14.0	21.10
Personal Care	1.040	1.980	2.45	3.4	5.40
Private Education	0.341	0.974	1.80	2.6	3.64

حال با استفاده از خطوط دستور زیر می‌خواهیم نمودارهای متنوعی را از مجموعه داده مورد نظر به تصویر بکشیم. نمودار مورد استفاده، نمودار هیستوگرام می‌باشد که حالت‌های مختلف آن را خواهید دید. در خط برنامه زیرتابع() برای ترسیم نمودار هیستوگرام استفاده شده است که آرگومان اول آن اشاره به مجموعه داده VADeaths دارد.

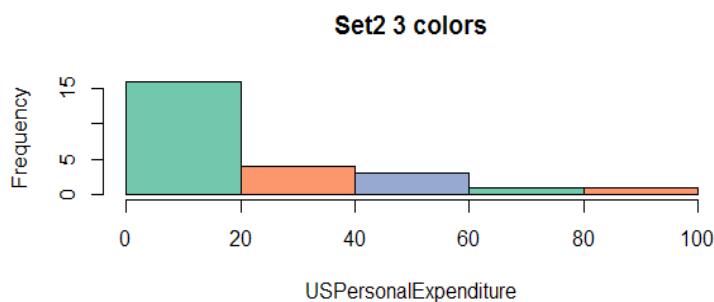
آرگومان col=brewer.pal(3,"Set3") نشان‌دهنده تعداد و نوع رنگ مستطیل‌ها می‌باشد که در بسته RColorBrewer وجود دارد. برای آگاهی از جزئیات خیلی بیشتر این بسته به مستندات آن در بخش راهنمای مراجعه نمایید. آرگومان بعدی یعنی main برچسب بالای نمودار را ایجاد می‌کند که برای مثال زیر همان "Set3 3 colors" چاپ گردیده است. برای آشنایی با تمامی آرگومان‌های تابع هستوگرام می‌توانید به بخش راهنمای محیط برنامه R-studio مراجعه

کرده و نام تابع را در بخش جستجو وارد کنید. به مثال‌های زیر توجه نمایید.

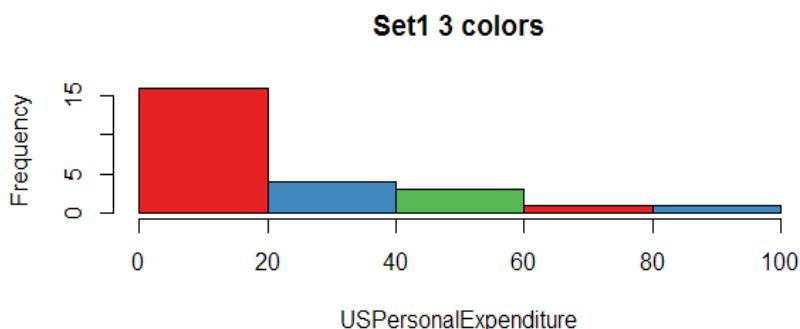
```
> hist(USPersonalExpenditure, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
```



```
> hist(USPersonalExpenditure,col=brewer.pal(3,"Set2"),main="Set2 3 colors")
```



```
> hist(USPersonalExpenditure, col=brewer.pal(3,"Set1"),main="Set1 3 colors")
```



برای مصورسازی داده‌ها می‌توان از ابزار پیشرفته‌تری که به صورت بسته‌های نرم‌افزاری به داخل R منتقل می‌یابند استفاده کرد که در ادامه مطالب با برخی از این بسته‌ها آشنا خواهید شد.

## بسته نرم افزاری در R

امروزه هزاران بسته نرم افزاری R برای استفاده عموم تولید شده است که جستجو در میان آنها برای پیدا کردن بسته موردنظر که بتواند مناسب عمل کند و نیاز شما را پاسخ گوید کمی مشکل است. برای این موضوع ما تعدادی از بسته‌هایی که در حال حاضر توسط استفاده‌کنندگان محبوبیت دارند را به شما معرفی می‌کنیم ولی شما خودتان نیز می‌توانید با مراجعه به منابع مربوطه بسته نرم افزاری خود را انتخاب کنید. سعی ما بر آن بوده که برای هر حیطه از عملکرد R تعدادی بسته نرم افزاری معرفی کنیم.

### بسته‌های نرم افزاری مصورسازی

بسته نرم افزاری مصورسازی پیشرفت‌تری نیز ایجاد شده‌اند که می‌توان گفت استاندارد جدید برای تجسم داده‌ها در R توسط RStudio است. به عنوان مثال می‌توان به `ggplot` و `ggvis` اشاره کرد.

بسته نرم افزاری `ggplot`: این بسته نرم افزاری یکی دیگر از ابزارهای مصورسازی پیشرفت‌ه است که برای تولید نمودارهای گرافیکی پیچیده چند لایه مورد استفاده قرار می‌گیرد.

بسته نرم افزاری `ggvis`: این بسته نرم افزاری به شما اجازه ایجاد نمودار ایستا و تعاملی برای نمایش، توزیع، نمایش روابط و غیره را با استفاده از دستور زبان گرافیک می‌دهد. این بسته نرم افزاری با تهیه یک چارچوب شهودی به شما اجازه توصیف یا ایجاد هرگونه طرح که فکر آن را در ذهن دارید می‌نماید و با یادگیری اجزاء دستور زبان آن، شما خود را توانمند به ساخت هزاران نوع مختلف مصورسازی داده‌ها خواهید کرد. نکته مثبت سازگاری `ggvis` برای اسناد وب می‌باشد. شما از طریق ذخیره آن به صورت فایل پی‌ان‌جی<sup>۱</sup> قابلیت انتشار و اشتراک‌گذاری را در محیط اینترنت خواهید داشت و با یک یا دو خط کد، می‌توانید یک طرح `ggvis` را به یک آنلاین یا یک ابزار اکتشاف داده‌های تعاملی برای تجزیه و تحلیل تبدیل نمایید.

### بسته‌هایی برای بارگذاری داده

بسته‌های RSQLite, RPostgreSQL, RMySQL, RODBC به شما این امکان را می‌دهند که بتوانید به پایگاه داده خاصی از طریق R متصل شوید که اسمی بسته‌ها خودشان تقریباً معرف عملکردشان می‌باشند.

بسته‌های XLConnect و `xlsx` امکان اتصال و کار با داده‌های مایکروسافت اکسل را به شما می‌دهند.

بسته foreign امکان خواندن داده‌های نرم افزار SPSS و SAS را می‌دهد.

و همچنین برای خواندن از فایل متنی ساده با قالب TXT می‌توانید بدون استفاده از بسته نرم افزاری و با استفاده از خود زبان R اقدام کنید.

## کار با داده‌ها و دست کاری آنها

با استفاده از بسته dplyr می‌توانید مجموعه‌ای از داده‌ها را از هم جدا کرده، خلاصه‌ای از آنها را به دست آورده و مجددآ آنها را سازماندهی کرده و به هم پیوندشان دهید و با بسته lubridate می‌توانید بر روی تاریخ و زمان و تغییر آنها کار کنید.

## مدل کردن داده‌ها

با استفاده از تابع Anova از بسته car می‌توانید جدول انووای<sup>۱</sup> نوع یک و نوع دو را ایجاد کنید.

بسته randomForest مدل RF را برای کاربرد در یادگیری ماشین ایجاد می‌کند.

با بسته mgcv می‌توانید یک مدل تعمیم‌یافته عمومی برای داده‌ها بسازید.

با بسته lme4/nlme می‌توان یک مدل تلفیقی خطی و غیرخطی ایجاد کرد.

بسته multicomp یک ابزار ارزیابی مقایسه‌ای چندگانه برای داده می‌باشد.

## ایجاد گزارش برای نتایج

بسته shiny یک روش مناسب برای جستجوی داده‌ها و به اشتراک گذاشتن آن‌ها است که برای افراد غیر برنامه‌نویس در سطح وب مورد استفاده قرار می‌گیرد.

بسته R Markdown برای ایجاد یک گزارش قابل ویرایش بسیار مناسب می‌باشد به این صورت که شما می‌توانید نتایج خود را در قالب فایل پی‌دی‌اف<sup>۲</sup> یا مایکروسافت ورد<sup>۳</sup> ذخیره نمایید.

بسته xtable یک شیء R را دریافت می‌کند و آن را برای نمایش به صورت یک صفحه وبی<sup>۴</sup> تبدیل می‌کند.

## کار با داده‌های مربوط به سری‌های زمانی

بسته‌های zoo و xts و quantmod برای کار با داده‌های آماری و اقتصادی مربوط به سری‌های زمانی و تحلیل آنها مورد استفاده قرار می‌گیرد.

## بسته‌هایی برای بالا بردن سرعت اجرای الگوریتم‌ها

با بسته Rcpp می‌توانید کدهای C++ را فراخوانی کنید که با آن برنامه‌های سبک و پرسرعت بنویسید.

با بسته parallel می‌توانید یک پردازش موازی برای بالا بردن سرعت پردازش به وجود آورید.

1. Anova table

2. Random forest

3. Pdf

4. Ms word

5. Html

## ایجاد ارتباط با قالب‌های داده‌ای عمومی وب

بسته‌های xml و json برای خواندن و نوشتن در فرمتهای معروف قابل ارائه در وب مورد استفاده قرار می‌گیرد.

**بسته‌ی rjson:** بسته‌بندی rjson از طریق Alex Couture-Beil نمادگذاری زبان مقصد جاوا اسکریپت به کار می‌رود. JSON یک فرمت استاندارد و بدون رمز است که اکثراً برای ساختارها و مقاصد داده در وب به کار می‌رود. اخیراً به دلیل کاربرد این فرمت در ایجاد برنامه‌های کاربردی وب، معروف شده است اما علی‌رغم اسمی که دارد، به مرورگرهای وب محدود نشده است.

پس از نصب بسته‌ی rjson، برای تبدیل JSON به R، به روش زیر عمل می‌کنیم:

```
>library(rjson)
>r-object<-formJSON(json-string)
```

برای تبدیل R به JSON به روش زیر عمل می‌کنیم:

```
>Json-string<-toJSON(r-object)
```

در نهایت می‌توانید با استفاده از بسته‌های testthat و ya roxygen2 بسته نرمافزاری خود را نوشه و ارزیابی کرده و منتشر کنید. در ادامه با توجه به اینکه این بسته‌ها هر کدام خصوصیات درونی متفاوتی دارند، با معرفی یک مجموعه بسته بسیار پرکاربرد در یادگیری ماشین به نام کرت<sup>۱</sup>، مقداری در رابطه با نحوه استفاده از آن به عنوان یک بسته نرمافزاری صحبت خواهیم کرد تا شما به صورت کلی با فرایند استفاده بسته‌ها آشنا شوید.

## مقدمه‌ای کوتاه درباره مجموعه بسته کرت

بسته نرمافزاری بسیار پرکاربرد و کارآمد و معروفی برای زبان R به نام کرت وجود دارد که حاوی توابعی برای ساده‌سازی فرآیند آموزش برای مسائل پیچیده دسته‌بندی و رگرسیون است. این بسته از تعداد زیادی از بسته‌های R استفاده می‌کند اما برای بالا بردن سرعت کار در هنگام شروع به کار همه آنها را فراخوانی نمی‌کند و بر حسب نیاز این کار انجام می‌شود.

بسته کرت را با استفاده از کد زیر نصب کنید که ابتدا لیست کاملی از بسته‌هایی که از آنها بهره می‌برد را نشان داده و همه آنها را نصب می‌کند.

```
> install.packages("caret", dependencies = c("Depends", "Suggests"))
```

البته می‌توانید با استفاده از دستور زیر فقط تعداد محدودی از بسته‌های ضروری‌تر را به همراه کرت نصب کنید و بقیه را به هنگام نیاز به آنها نصب و استفاده کنید. برای این کار فقط کافی است که دستور زیر را اجرا کنید.

```
> install.packages("caret")
```

برای اینکه مطمئن شوید که همه بسته‌های موردنیاز نصب شده‌اند صفحه‌های اصلی مربوط به help بسته، در آدرس زیر است:

<http://caret.r-forge.r-project.org/>

در اینجا، مثال‌های متعدد و حجم زیادی از اطلاعات وجود دارد که بیشتر در شکل بسته یافت شده است.

مجموعه بسته کرت چندین تابع با ارزش دارد که کارشناس ایجاد مدل‌هایی برای محاسبه، ارزیابی نتایج، و همچنین برای کارهایی مانند انتخاب ویژگی و غیره می‌باشد. در ادامه یک مثال ساده آورده‌ایم تا شما با نحوه استفاده تابع‌های درون کرت یا هر بسته نرم‌افزاری دیگر آشنا شوید و در ادامه تعدادی از توابع مورد استفاده در کرت معرفی شده است.

### تقسیم مجموعه داده‌ها با استفاده از کرت:

در تحلیل داده‌ها براساس الگوریتم‌های یادگیرنده، همیشه مجموعه داده‌های اولیه را معمولاً به دو دسته تقسیم می‌کنیم. برای این کار در بسته کرت می‌توان از روش زیر استفاده کرد.

داده‌ها باید به یک مجموعه آموزشی و یک مجموعه آزمون<sup>۱</sup> تقسیم شوند و برای این کار نیز از تابع `createDataPartition` که در بسته کرت وجود دارد استفاده می‌کنیم.

```
> library(caret)
> library(mlbench)
> data(Sonar)
> set.seed(107)
> inTrain <- createDataPartition(y = Sonar$Class, p = .75, list = FALSE)
> str(inTrain)
```

```
int [1:157, 1] 1 2 3 6 7 9 10 11 12 13 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
```

همان‌گونه که در برنامه بالا مشاهده می‌نمایید در خط اول و دوم بسته‌های caret و mlbench و در خط سوم برنامه مجموعه داده Sonar فراخوانی شده است. داده‌های Sonar شامل ۲۰۸ نقطه داده است که در ۶۰ پیش‌گویی کننده، جمع‌آوری شده است.

در خط پنجم یک متغیر با نام `inTrain` ایجاد شده که محتوای `createDataPartition` را در خود نگهداری می‌کند. در تابع `createDataPartition` که وظیفه‌اش تقسیم داده‌های اولیه به دو دسته داده‌های آموزشی و داده‌های

آزمون است. آرگومان اول محل نگهداری نتیجه تابع و آرگومان دوم مشخص کننده درصد تقسیم کل داده موجود و اختصاص آن به داده آموزشی و آرگومان سوم مشخص کننده قالب‌بندی نتیجه می‌باشد. خط آخر نیز نشان‌دهنده ساختار اشیاء در زبان R می‌باشد که در آنجا داده‌های آزمون است.

به صورت پیش‌فرض، `createDataPartition`، یک عمل تجزیه تصادفی داده‌ها برای دسته‌بندی آنها انجام می‌دهد. برای درک بهتر نحوه تقسیم داده توسط این تابع، برای تقسیم داده‌ها به داده‌های آموزشی و آزمون از دستور زیر استفاده می‌کنیم:

```
> training <- Sonar[ inTrain,]
```

```
> testing <- Sonar[-inTrain,]
```

```
> nrow(training)
```

```
[1] 157
```

```
> nrow(testing)
```

```
[1] 51
```

## توابع در مجموعه بسته نرم‌افزاری کرت

در ادامه به تعدادی از تابع‌های موجود در بسته کرت اشاره می‌کنیم که شما می‌توانید با مراجعه به مستندات مربوطه درباره آرگومان‌های این تابع‌ها اطلاعات آن‌ها را به دست آورید و از آنها به صورت بهینه استفاده کنید. توجه نمایید که اطلاعات بیشتر در رابطه با هر بسته توسط ناشر آن در دسترس عموم قرار داده می‌شود که باید توسط برنامه‌نویسان مطالعه شوند تا از اسمی توابع و آرگومان‌های درون بسته آگاهی یابند.

### تابع : `rain(x, ...)`

برای تخمین پارامترهای بهینه در مدل‌های رگرسیونی و همچنین مدل‌های دسته‌بندی بهینه مورد استفاده قرار می‌گیرد.

### تابع : `trainControl()`

از این تابع برای کنترل پارامترها در مرحله آموزش استفاده می‌شود.

### تابع : `cars()`

مجموعه داده مربوط به فروش کتاب می‌باشد.

### تابع : `BloodBrain()`

یک مجموعه داده برای غلظت خون است.

**تابع : confusionMatrix()**

برای ایجاد یک ماتریس درهم ریختگی<sup>۱</sup> مورد استفاده قرار می‌گیرد.

**تابع : createDataPartition()**

با استفاده از این تابع می‌توان داده‌های آزمون و آموزشی را بخش‌بندی کرد.

**تابع : downSample()**

داده‌های نامتعادل را زیر نمونه‌برداری می‌کند.

**تابع : findCorrelation()**

این تابع در درون ماتریس همبستگی جستجو می‌کند تا یک برداری از اعداد صحیح را برگرداند که مشخص کننده ستون‌هایی است که می‌توان با حذف آنها همبستگی‌ها را بهینه کرد.

**تابع : calibration()**

برای مدل‌های مختلف دسته‌بندی، این تابع یک نمودار تعاملی ایجاد می‌کند که نشان‌دهنده چگونگی سازگاری مدل با نرخ رویداد مشاهده شده است.

**تابع : avNNet.default()**

برای ایجاد یک شبکه عصبی مورد استفاده قرار می‌گیرد که میانگینی از چند مدل مختلف از شبکه عصبی را ارائه می‌دهد.

براساس مثال‌های بالا مشاهده فرمودید که در درون مجموعه بسته کرت انواع مختلفی از توابع وجود دارند که برای کارهای متنوعی طراحی گردیده‌اند. باید در نظر داشته باشید که ما به دلیل محدودیت این کتاب قادر به بازتاب تمام مستندات عظیم موجود درباره بسته‌های نرم‌افزاری مختلف نیستیم و همچنین این بسته‌های نرم‌افزاری و امکانات آنها به صورت مداوم در حال ارتقاء و بهروز شدن می‌باشند پس بر این اساس هدف ما این بود که شما به وسیله مثال‌هایی با ساختار این بسته‌های نرم‌افزاری و نحوه استفاده آنها در R آشنا شوید.



پیوست

# آموزش مقدماتی پایتون

## آموزش مقدماتی پایتون

در این بخش هدف ما معرفی اصول اولیه برنامه‌نویسی با پایتون می‌باشد. پایتون به عنوان یک زبان چند منظوره برای برنامه‌نویسان است که از محبوبیت بسیاری در میان متخصصین یادگیری ماشین برخوردار است. این زبان برخلاف زبان‌های برنامه‌نویسی مشابه دیگر همانطور که در داده کاوی پرقدرت است در تحلیل و پردازش تصویر نیز بسیار کارآمد می‌باشد.

## نصب و راهاندازی پایتون

پایتون را در محیط‌های مختلفی می‌توان نصب و اجراء نمود. این زبان در سیستم عامل‌های مختلفی مانند لینوکس، ویندوز و غیره قابل اجراء می‌باشد. همچنین IDE‌های بسیاری برای این زبان وجود دارد که از معروفترین‌های آن Spider و Pycharm را می‌توان نام برد که به صورت رایگان قابل دریافت از وب سایت مربوطه می‌باشد که به سادگی می‌توانید با جستجوی آن در اینترنت لینک مربوطه را بیابید. به دلیل اینکه در اینجا هدف ما معرفی نحوه برنامه‌نویسی زبان پایتون می‌باشد، پس شما به سادگی می‌توانید بدون نصب این IDE‌ها، تنها با نصب برنامه اصلی پایتون کار خود را شروع نمایید. برای دریافت فایل نصی پایتون می‌توانید به وبسایت <https://www.python.org> مراجعه نموده و نسخه مورد نظر خود را دریافت و نصب نمایید. همچنین پس از اجرای پایتون می‌توانید برای آگاهی از نسخه نصب شده دستور `help()` را در خط دستور تایپ نمایید و کلید Enter را بزنید.

برای نصب بسته‌های نرم افزاری ابتدا به اینترنت متصل شوید و سپس با اجرای نرم افزار پایتون در آن خط دستور زیر را برای نصب بسته‌های مختلف تایپ نمایید.

`Pip install PackageName`

در خط دستوری بالا `PackageName` اشاره به نام بسته مورد نیاز برای نصب دارد که یک مثال کاربردی را در زیر می‌توانید مشاهده نمایید. در این مثال بسته نرم‌افزاری `numpy` نصب می‌شود.

`Pip install numpy`

اگر به اینترنت متصل باشید و دستور بالا را بدون خطای نوشتاری تایپ نمایید، پایتون به صورت اتومات به اینترنت متصل شده و نرم افزار مربوطه را نصب می‌نماید.

## محاسبات عددی مقدماتی در پایتون

برای انجام عملیات جمع، تفریق، ضرب، تقسیم، باقیمانده، توان، به ترتیب از علامت‌های `+`, `-`, `*`, `/`, `%` استفاده می‌شود. به مثال‌های زیر که در خط دستور پایتون قابل اجراء هست توجه نمایید.

<code>&gt;&gt;&gt;25-5 20</code>	<code>&gt;&gt;&gt;25+5 30</code>	<code>&gt;&gt;&gt;25/5 5.0</code>	<code>&gt;&gt;&gt;25*5 125</code>
<code>&gt;&gt;&gt;25%5 0</code>	<code>&gt;&gt;&gt;27%5 2</code>	<code>&gt;&gt;&gt;2**2 4</code>	<code>&gt;&gt;&gt;2.20*6 13.200000000000001</code>

محاسبه معادل عددی با استفاده از توابع درونی پایتون نیز در مثال‌های زیر آورده شده است. همانطور که مشاهده می‌نمایید تابع `hex()` مقدار ۱۰۲۴ را به معادل آن در نوع هگزادسیمال تبدیل می‌نماید و تابع `bin()` مقدار عددی داده شده درون آرگومان خود را معادل آن در قالب باینری تبدیل می‌نماید.

```
>>> hex(1024)
```

```
'0x400'
```

```
>>> bin(1024)
```

```
'0b100000000000'
```

## نحوه محاسبه عبارت عددی

در زبان پایتون عبارت عددی قابل محاسبه می‌باشد و ابتدا اعداد داخل پرانتز محاسبه شده و سپس نتیجه آن با بیرون پرانتز محاسبه می‌شود. اولویت عملگرها مانند سایر زبان‌های برنامه‌نویسی استاندارد می‌باشد که اولویت محاسبه به ترتیب به صورت «داخل پرانتز، علامت ضرب و تقسیم، علامت جمع و تفریق» می‌باشد.

```
>>> (20.0+4)/6
```

```
4
```

```
>>> (2+3)*5
```

```
25
```

## نحوه استفاده از متغیرها

مقادیر را می‌توان همواره در درون یک متغیر ذخیره نمود و آن متغیر را در محاسبات مورد استفاده قرار داد. یک متغیر می‌توان یک حرف و یا یک کلمه باشد. به مثال‌های زیر توجه نمایید.

ابتدا متغیرهای `a` و `b` را با مقادیر ۲ و ۳ مقدار دهی می‌نماییم و سپس عملیات مختلف را بر روی آنها انجام می‌دهیم. نکته قابل توجه این است که برخی اسامی به عنوان کلید واژه درونی پایتون هستند که باید توجه نمود که کلمات انتخاب شده با آنها تشابه نداشته باشد. به عنوان مثال نمی‌توان کلمه `if` را به عنوان متغیر انتخاب نمود. همچنین باید توجه داشته باشید که پایتون یک زبان حساس به بزرگ یا کوچک بودن حروف است و باید در نام‌گذاری متغیرها به این نکته توجه داشت.

```
>>> a=2
```

```
>>> b = 3
```

```
>>> a+b
```

```
5
```

```
>>> a-b  
-1
```

```
>>> a**2 + b**2  
13
```

```
>>> a>b  
False
```

## توابع ریاضی

تابع ریاضی دیگری در پایتون وجود دارد که می‌توان با آنها مواردی مانند مقادیر سینوس، کسینوس، ریشه اعداد و مواردی مانند اینها را محاسبه نمود. برای استفاده این تابع باید آنها را از طریق ماثول Math بارگذاری یا فراخوانی نمود.

از طریق خط اول تکه کد زیر می‌توان تابع ریاضی را فراخواند و چنانچه مشاهده می‌نمایید با تابع درون ماثول مورد استفاده محاسبات را انجام داد که در خط دوم با تابع sqrt() می‌توان ریشه عدد وارد شده در آرگومان تابع را به دست آورده سپس با استفاده از تابع floor() و قرار دادن مقدار به دست آمده در آرگومان آن می‌توان به عدد صحیح تبدیل نمود.

```
>>> from math import *
```

```
>>> sqrt(2)  
1.4142135623730951
```

```
>>> floor(1.4142135623730951)  
1
```

باید توجه نمود که مقدار به دست آمده از طریق تابع floor همیشه مقدار پایه می‌باشد، یعنی اگر مقدار قرار گرفته در آرگومان برابر ۱,۹۹ بود نیز این تابع مقدار ۱ را در خروجی ارائه می‌نمود.

در خط زیر با استفاده از تابع hex() می‌توان عدد صحیح ۲۵۷۲ را به مقدار معادل آن در هگزادسیمال تبدیل نمود.

```
>>> hex(2572)  
'0xa0c'
```

مثال‌های بالا تنها با هدف معرفی نحوه عملکرد تابع ریاضی موجود در پایتون ارائه می‌گردد، اما شما می‌توانید تابع بسیاری را در پایتون فراخوانی نمایید تا محاسبات ریاضی مختلف را به سادگی انجام دهید.

## نوشتن اسکریپت پایتون

در کد نویسی پایتون معمولاً کدها در قالب یک اسکریپت هستند و اگر هدف نوشتن کد تنها محاسبات ریاضی ساده و کوچک نیست پس بهتر است کدهای خود را در قالب اسکریپت بنویسید تا بر این اساس بتوانید یک برنامه کامل را در قالب یک اسکریپت اجراء و برای استفاده های بعدی ذخیره نمایید. این اسکریپت ها را معمولاً در درون یک IDE مخصوص پایتون می نویسند که در ابتدای این بخش مثالی از آنها را مطرح نموده ایم.

یک تکه برنامه کوچک که به صورت یک اسکریپت می توان آن را اجراء نمود به صورت زیر می باشد، این برنامه مساحت یک دایره با قطر ۱۰,۰ را محاسبه و در خروجی چاپ می نماید.

```
from math import *
d = 10.0 # diameter
A = pi * d**2 / 4

print "diameter =", d
print "area =", A
```

در برنامه بالا علامت # برای نوشن توضیحات مورد استفاده قرار می گیرد که در اصل هر عبارتی در پشت آن قرار گیرد تبدیل به یک توضیح یا کامنت می شود و در اینصورت زبان پایتون آن را در محاسبات نادیده می گیرد و این توضیحات تنها برای توصیف کدها و برنامه مورد استفاده قرار می گیرد. برای چاپ مقادیر محاسبه شده نیز از کلمه print استفاده می نماییم که مشاهده می نمایید عبارت درون (") را نیز دقیقاً به عنوان کاراکتر در خروجی چاپ نموده و مقادیر درون متغیرها را در ادامه چاپ می نماید.

## اعلان نوع متغیر و اشیاء

در پایتون مقادیر در اصل در درون اشیاء ذخیره می شوند. به مثال زیر توجه نمایید.

اعشاری D = 10.0

صحیح d = 10

در مثال بالا d به عنوان یک شیء ایجاد می شود که نوع این شیء بسته به نوع مقدار اختصاص داده شده به آن تغییر می کند. به عبارت دیگر اگر به d یک عدد اعشاری بدهیم در این حالت شیء d از نوع اعشاری می شود و اگر عدد یا مقدار آن را یک عدد صحیح در نظر بگیریم شیء d یک شیء از نوع صحیح می شود. در اصل این خاصیت در پایتون باعث می شود که دیگر نیاز به مشخص نمودن نوع متغیر در زمان اعلان متغیر نباشیم و همین موضوع باعث بالا رفتن انعطاف پذیری بیشتر این زبان برنامه نویسی شده است.

## تبدیل نوع داده ورودی

برای اینکه پایتون بتواند داده‌های ورودی ما را از نوع رشته در نظر بگیرد و آن را در خروجی نشان دهد از تابع `Input()` استفاده می‌نمایید. به مثال زیر توجه نمایید.

`Input ("Input your string")`

حال اگر بخواهید یک مقدار عددی را که از نوع صحیح یا اعشار می‌باشد از خروجی درخواست نمایید تا آن عدد وارد شده را در نوع مورد نظر دریافت و در یک متغیر ذخیره نمایید می‌توانید با قرار دادن تابع `input()` درون یک تابع `int()` برای عدد صحیح و تابع `float()` برای عدد اعشاری اقدام نمایید.

به مثال‌های زیر توجه نمایید.

```
X = int(input("Input an integer: "))
print(x)
```

```
y = float(input("Input a float: "))
print(y)
```

تکنیک بالا کاربردهای بسیاری دارد که یکی از آنها را می‌توان در اسکریپت زیر مشاهده نمود. برنامه زیر با دریافت قطر یک دایره می‌تواند مساحت دایره را محاسبه و در خروجی چاپ نماید.

```
From math import *
d = float(input("Diameter: "))
A = pi * d**2 / 4
print("Area = ", A)
```

توضیحات برنامه بالا: در خط اول توابع مربوط به محاسبات ریاضی بارگذاری می‌شود. در خط دوم با استفاده از تابع `input()` از خروجی اندازه قطر دایره را از کاربر دریافت می‌نمایید در نوع اعشاری ذخیره می‌نماید و با استفاده از فرمول موجود در خط سوم مساحت دایره را محاسبه می‌نماید و در خط آخر با استفاده از تابع `print()` مقدار محاسبه شده را در خروجی چاپ می‌نماید.

به شرط اینکه مقدار قطر دایره را ۵ وارد کنیم خروجی برنامه برابر با ۱۹,۶۳ می‌گردد که مقدار دقیق آن در زیر مشخص گردیده است.

Diameter: 5

Area = 19.634954084936208

## حلقه‌های برنامه‌نویسی در پایتون

ما همواره از طریق کدهای پایتون می‌توانیم کامپیوتر را برای انجام محاسبات طولانی مورد استفاده قرار دهیم. حلقه‌های برنامه‌نویسی می‌توانند راه حل مناسبی برای برخی از این محاسبات باشند. یکی از پرکاربردترین حلقه‌ها حلقه While می‌باشد.

برنامه زیر می‌تواند ریشه تمام اعداد صحیح بین ۰ تا ۱۰۰ را محاسبه نماید برای این منظور در اینجا ما از حلقه while استفاده می‌نماییم.

```
From math import *
I = 0
while i<= 100:
    print (I, "\t\t", sqrt(i))
    I = I + 1
print ("READY!")
```

خط دوم برنامه شمارنده را مقداردهی اولیه می‌نماید که همان  $i=0$  می‌باشد، و در خط سوم برنامه حلقه تعریف می‌شود و شرط حلقه برای ایستادن برنامه از ادامه اجرای حلقه تعریف می‌شود که همان  $i<=100$  می‌باشد و در خط چهارم برنامه با استفاده از تابع `sqrt()` ریشه عدد جاری را محاسبه و در خروجی چاپ می‌نماید. و در خط بعد آن یک عدد به شمارنده اضافه می‌شود تا عدد بعدی را محاسبه نماید که پس از برقرار شدن شرط حلقه while که همان رسیدن عدد به ۱۰۰ است، برنامه به خط بعد می‌رود که همان خط آخر برنامه است و از این طریق در خروجی READY! را چاپ می‌نماید.

اگر برنامه بالا را در یک IDE اجراء نماییم خروجی زیر را مشاهده می‌نماییم.

عدد صحیح	ریشه عدد صحیح پس از اجرای حلقه
0	0.0
1	1.0
2	1.4142135623730951
3	1.7320508075688772
4	2.0
....	....
95	9.746794344808963
96	9.797958971132712
97	9.848857801796104
98	9.899494936611665
99	9.9498743710662
100	10.0
READY!	
Process finished with exit code 0	

## شرطهای اجرای برنامه

برای آنکه بتوان شرطهای منطقی برای کنترل جریان جرای برنامه‌ها به وجود آورد می‌توان از if، elif و else استفاده نمود. در مثال زیر می‌توان نحوه عملکرد if را مشاهده نمود که در آن در خط اول یک رشته از کاربر دریافت می‌گردد که در خط دوم برنامه توسط عبارت شرطی همانند بودن ورودی مقایسه می‌گردد و در خط آخر برنامه در صورت برقراری شرط ورودی کاربر در خروجی چاپ می‌شود.

```
S = input ("Input your name: ")
```

```
if s == "Tom":
```

```
    print ("Hello ", s)
```

توجه داشته باشید که با اجرای برنامه در کنسول از شما خواسته می‌شود که یک اسم را وارد نمایید و اگر این اسم برابر با Tom نبود برنامه بدون چاپ چیزی پایان می‌یابد، اما اگر کلمه Tom را در کنسول تایپ نمایید و اینتر را فشار دهید، در خروجی عبارت Hello Tom نمایش داده می‌شود.

حال می‌توانیم برنامه بالا را کمی توسعه دهیم. در برنامه زیر با استفاده از کلمه else می‌توانیم برنامه را به گونه‌ای بنویسیم که اگر شرط برقرار نبود دیگر از برنامه خارج نشود و در خروجی رشته Hello unknown را چاپ نماید.

```
S = input ("Input your name: ")
```

```
if s == "Tom":
```

```
    print ("Hello ", s)
```

```
else:
```

```
    print ("Hello unknown")
```

همچنین همانگونه که در برنامه زیر مشاهده می‌شود می‌توان با استفاده از کلمه elif بیش از یک شرط را در برنامه به صورت پی در پی به اجرا درآورد.

```
S = input ("Input your name: ")
```

```
if s == "Tom":
```

```
    print ("Hello ", s)
```

```
elif s == "John":
```

```
    print ("I'm so glad to see you ", s)
```

```
elif s == "Rebeca":
```

```
    print ("I didn't expect you ", s)
```

```
else:
```

```
    print ("Hello unknown")
```

## Tuple ها

در پایتون متغیرها می توانند تحت یک نام در یک گروه قرار گیرند که این حالت با چندین روش انجام می شود.  
به مثالی در زیر توجه نمایید.

در تکه کد زیر مقدار ۵ و ۳ به x و y اختصاص داده شده و در خروجی مقادیر عددی مربوطه چاپ می شود.

```
(x,y) = (5, 3)
```

```
coordinates = (x,y)
```

```
print(coordinates)
```

خروجی برنامه به صورت زیر خواهد بود.

```
(5, 3)
```

در تکه کد زیر در خط اول برنامه مقادیر عددی به متغیر dimensions اختصاص داده می شود و سپس می توان چنانچه در خط دوم مشاهده می نمایید همه مقادیر را به یکباره در خروجی چاپ نمود و یا اینکه با مشخص نمودن اندیس هر عدد مقدار همان اندیس را توسط متغیر مربوطه در خروجی چاپ نمایید. برای چاپ هر اندیس چنانچه از خط سوم به بعد در تکه کد زیر مشاهده می نمایید می توان متغیری را که توان اعداد را در خود جای داده است را با دستور print اجراء نمود و حتما باید اندیس مقدار عددی مورد نظر را نیط برای آن مشخص نمود.

```
dimensions = (8, 5.0, 3.14)
```

```
print(dimensions)
```

```
print(dimensions[0])
```

```
print(dimensions[1])
```

```
print(dimensions[2])
```

خروجی برنامه به صورت زیر خواهد بود.

```
(8, 5.0, 3.14)
```

```
8
```

```
5.0
```

```
3.14
```

## لیست ها

لیست ها همان توالی مشخص شده ای از داده هایی با نوع مختلف هستند. به عبارتی می توان انواع داده های متفاوت مانند آدرس افراد در قالب رشته، مقدار حقوق دریافتی را در قالب یک عدد اعشاری و همچنین انواع داده های دیگر را به صورت یک لیست ذخیره نمود و سپس آنها را با استفاده از یک نام مشخص ذخیره و بازیابی نمود.

برای تعریف و اختصاص دادن آیتم‌ها به یک لیست می‌توان براساس دستور زیر عمل نمود.

```
my_list = [item_1, item_2, item_3]
```

همچنین در پایتون امکان ایجاد لیست خالی نیز وجود دارد.

```
my_list = []
```

یک مثال از ایجاد یک لیست از اسمی رنگ‌ها و چاپ هر کدام از آیتم‌های درون آن را به صورت جداگانه در زیر می‌توانید مشاهده نمایید. توجه نمایید که اندیس اول از عدد صفر شروع می‌شود و تنها کافی است که برای چاپ اسم یک رنگ از درون لیست، نام لیست را با شماره اندیس رنگ در خروجی چاپ نمایید.

```
colors = ["red", "orange", "yellow", "green", "indigo", "white"]
```

```
>>> print(colors[0])
```

red

همچنین برای چاپ نام رنگ پنجم باید از دستور زیر استفاده نمایید.

```
>>> print(colors[4])
```

Indigo

حتی می‌توانید برای مشاهده تعداد رنگ‌های درون لیست ازتابع len() استفاده نمایید که برای این منظور به مثال زیر توجه نمایید.

```
>>> len(colors)
```

6

حال فرض کنید که نیاز دارید تا تمامی رنگ‌هایی که یک رنگین کمان را ایجاد می‌نمایند را انتخاب نمایید. برای این منظور ابتدا از لیست رنگ ایجاد شده در بالا می‌خواهیم استفاده نماییم. برای این منظور باید ابتدا انواع رنگ موجود در این لیست را مشاهده نمود و هر رنگی که کم داشت به لیست اضافه و هر رنگی که برای رنگین کمان مورد استفاده نمی‌باشد را از لیست حذف نماییم.

با دستور زیر می‌توان محتوای درون لیست را مشاهده نمود.

```
>>> print(colors)
```

```
['red', 'orange', 'yellow', 'green', 'indigo', 'white']
```

حال با توجه به خروجی بالا ما باید رنگ white را که در رنگین کمان وجود ندارد را حذف می‌نماییم و همچنین رنگ violet را که در رنگین کمان وجود دارد به آن اضافه می‌نماییم.

برای حذف یک رنگ که در اصل یکی از آیتم‌های لیست می‌باشد با دستور زیر عمل می‌نماییم.

```
>>> colors.remove("white")
```

برای افزودن یک رنگ به لیست خود از دستور زیر استفاده می‌نماییم.

```
>>> colors.append("violet")
```

و در نهایت با استفاده از دستور زیر مجدداً آیتم‌های درون لیست را چاپ و بررسی می‌نماییم.

```
>>> print(colors)
```

## تعريف توابع توسيط کاربر (User-defined function)

یک تابع مجموعه‌ای از عبارتها است که یک کار مشخصی را انجام می‌دهد. توابع ایجاد شده می‌توانند در هر جایی از یک برنامه با فراخوانی نام تابع همان وظیفه مشخص را به راحتی انجام دهند و دیگر نیازی به نوشتن مجدد کدهای درون تابع نیست. همچنین توابع می‌توانند به عنوان آرگومان مورد استفاده قرار گیرند یا به یک متغیر اختصاص داده شوند.

تابع درون پایتون توسيط کلمه def تعريف می‌شوند که ساختار برنامه پایتون برای تعريف تابع به صورت زیر می‌باشد.

```
def <function_name(<argument1>, <argument2>, ....):
    <statements>
    ...
    return <returnvalue(s)>
```

در زیر یک تابع تعريف می‌نماییم که در صورت فراخوانی و چاپ آن در خروجی بتواند عبارت "Python is so popular" را چاپ نماید.

```
def python_popular():
    print ("Python is so popular!")
```

به مثالی دیگر توجه نمایید که در آن یک تابع تعريف شده که می‌تواند یک عبارت عددی را محاسبه نماید که این تابع می‌تواند مجدداً تنها با فراخوانی نام آن این محاسبات را مجدداً انجام دهد.

```
def calculator(x, y):
    return x * y + 2
```

در تابع بالا مقادير ورودي در معادله  $x * y + 2$  جايگذاري و محاسبه می‌شود. مقادير ورودي به عنوان x و y در نظر گرفته می‌شود که توسيط آرگومان تابع دریافت می‌شود.

می‌توانيد در ادامه با فراخوانی نام تابع مانند زیر محاسبه مورد نظر را برای هر مقدار مورد نظر انجام دهيد.

```
Print(calculator(2,6))
print(calculator(3,7))
```

خروجی مورد نظر برایتابع (calculator) که دو بار فراخوانی شده و هر بار با آرگومان‌های مختلف مقداردهی گردیده است به صورت زیر می‌شود.

14

23

## کتابخانه‌های پرکاربرد پایتون برای یادگیری ماشین

یکی از کاربردهای زبان برنامه‌نویسی پایتون در حوزه تحلیل داده و یادگیری ماشین است. یکی از برجسته‌ترین مزایای پایتون این است که در آن کتابخانه‌های منبع باز بسیاری وجود دارند که هر کدام برای وظیفه مشخصی تولید شده‌اند و این امر باعث بالا رفتن سرعت و کیفیت برنامه‌نویسی با پایتون گردیده است. بر این اساس در ادامه مطالب تعدادی از کتابخانه‌های معروف و پرکاربرد پایتون را که برای داده کاوی و علوم داده مورد استفاده قرار می‌گیرند را معرفی می‌نماییم تا بتوانید علاوه بر آشنایی با این برنامه‌ها و کاربرد آنها، این برنامه‌ها را به عنوان نمونه‌هایی از کاربرد بسته‌های نرم‌افزاری و کتابخانه‌ها یادگیری ماشین مدنظر قرار دهید و با نحوه استفاده پایتون از برنامه‌های درون این کتابخانه‌ها آشنا شوید.

### بسته نرم‌افزاری Scikit-learn

این کتابخانه در خود توابع بسیاری برای پیاده‌سازی الگوریتم‌های یادگیری ماشین دارد. در اصل Scikit-learn برای تازه کارها بیش از یک ابزار مبتدی برای پیاده‌سازی الگوریتم‌های تحلیل داده می‌باشد و در عین سادگی آن یک برنامه‌نویس می‌تواند با استفاده از توابع درون آن الگوریتم‌های مختلفی در حوزه رگرسیون، خوشبندی و همچنین دسته‌بندی داده‌های باناظر پیاده‌سازی نماید. این کتابخانه دو بسته نرم‌افزاری SciPy و NumPy را نیز درون خود مورد استفاده قرار می‌دهد. توجه داشته باشید که در این کتاب به دلیل توان بالای این کتابخانه در پیاده‌سازی الگوریتم‌ها، سادگی آن برای فهم افراد مبتدی ما در پیاده‌سازی الگوریتم‌های هر فصل برای مثال‌های پایتون از این کتابخانه استفاده می‌نماییم. توجه نمایید که برای نصب هر بسته نرم‌افزاری در پایتون اگر از IDE پایچرم<sup>1</sup> استفاده می‌نمایید می‌توانید ابتدا سیستم خود را به اینترنت متصل نمایید و سپس با مراجعه به منوی زیر و کلیک بر روی علامت مثبت موجود در صفحه مربوطه در پایچرم بسته‌های مورد نیاز خود را به راحتی نصب نمایید.

File -> Default Settings -> Project Interpreter

در صفحه باز شده باید بسته مورد نیاز خود را یا از درون لیست موجود انتخاب نمایید یا با تایپ نام دقیق آن در بخش جستجو آن را یافته و بر روی آن کلیک نموده و دکمه نصب را بزنید.

## Numpy بسته نرم افزاری

بسته NumPy یکی از بسته های متداول در پایتون می باشد که تقریبا در ۸۰ درصد پروژه های یادگیری ماشین مورد نیاز می باشد. این بسته نرم افزاری بسیار مهم کارش ایجاد آرایه های چند بعدی می باشد. این بسته به دلیل ارتباط نزدیک با لایه ساخت افزاری سیستم می تواند با سرعت بالایی محاسبات ماتریسی را انجام دهد و همچنین این زبان به دلیل طراحی آن برای حل معادلات عددی یک ابزار پرقدرت برای محاسبات علمی و مهندسی می باشد.

## SciPy بسته نرم افزاری

بسته scipy حاوی جعبه ابزارهای مختلف طراحی شده برای حل مسائل رایج در محاسبات علمی است. مازول های مختلف این بسته نرم افزاری می توانند با سایر برنامه ها به راحتی ادغام شده و اعمال مختلفی را در حوزه بهینه سازی، پردازش تصویر، آمار و غیره انجام دهند. باید توجه داشت که یک برنامه نویس پایتون می تواند بدون استفاده این بسته ها نیز برنامه خود را کدنویسی نماید اما استفاده از توابع درون این بسته های برنامه نویسی می تواند بسیاری از مشکلات برنامه نویسی مانند خطای برنامه نویسی، عدم بهینه بودن کدها و سایر مشکلات رایج را از میان بردارد.

## Panda بسته نرم افزاری

بسته نرم افزاری پاندا با ارائه توابع بسیاری می تواند داده ها را به صورت یک ساختمان داده سطح بالا مدیریت نماید که در عین حال این عملیات با حداقل پیچیدگی در برنامه نویسی و به راحتی بتواند قابل انجام باشد. این بسته روش های بسیاری برای گروه بندی، ترکیب و فیلتر کردن داده ها و همچنین انجام تجزیه و تحلیل های سری زمانی استفاده می کند. این بسته نرم افزاری به راحتی می تواند داده ها را از منابع مختلف از جمله پایگاه داده SQL، CSV، JSON، اکسل، Barگیری کنند و داده ها را برای آماده سازی برای انجام عملیات مختلف بر روی آنها دستکاری نماید.

## Matplotlib بسته نرم افزاری

با استفاده از این بسته نرم افزاری که یکی دیگر از کتابخانه های استاندارد پایتون می باشد خروجی های الگوریتم های یادگیری ماشین یا نتایج هرگونه تحلیل داده ای را می توان به صورت بصری در قالب نمودار ارائه نمود که این بسته نرم افزاری توابع بسیاری را برای این منظور درون خود دارد. این کتابخانه توابع می تواند هر نوع نمودار دو بعدی و گرافی را ترسیم نماید. این ابزار به دلیل سطح پایین بودن آن مقداری کدنویسی بیشتری نیاز دارد تا بتوان نمودارهایی با ظاهر زیباتر تولید نمود اما سرعت و انعطاف پذیری آن در تولید هر گونه نمودار و عدم محدودیت آن این ابزار را به یکی دیگر از ابزارهای تفکیک ناپذیر در تحلیل داده تبدیل نموده است.

همچنین از دیگر خصوصیات این بسته نرم افزاری این است که تمامی رابط گرافیکی سیستم عامل های مختلف با آن سازگار هستند یعنی می توان از این ابزار در محیط های مختلف بدون مشکل استفاده نمود و همچنین خروجی هایی با فرمت مختلف را از قبیل PDF, SVG, JPG, PNG, BMP, GIF می تواند ارائه نماید.

کتابخانه‌ها و بسته‌های نرم‌افزاری بالا تنها تعدادی از معروفترین‌ها هستند که جهت آشنایی شما با چگونگی تعامل زبان پایتون با بسته‌های تولید شده برای این زبان است. در اصل کتابخانه‌های بسیار زیادی وجود دارند که در هر پروژه پیاده‌سازی الگوریتم یادگیری ماشین یا داده کاوی می‌توان از مناسبترین‌ها برای برنامه مورد نظر استفاده نمود. همچنین یادگیری این روش‌های متدالو و پایه‌ای شما را برای استفاده از کتابخانه‌های بزرگتر و پیشرفته‌تر آماده می‌سازد که برخی از آنها عبارتند از TensorFlow، PyTorch، Keras، MXNet، Caffe، CNTK.

برای پیاده‌سازی الگوریتم‌های شبکه عصبی عمیق و محاسبات پیچیده‌تر مورد استفاده قرار می‌گیرند.

## **نمایه (انگلیسی و فارسی)**

## «انگلیسی به فارسی»

<b>A</b>		
Agent graph	۲۱۵	گراف عامل
Anomaly detection	۱۶۰	تشخیص ناهنجاری
Artificial neural network	۹۴	شبکه عصبی مصنوعی
Auto encoder	۲۲۲	خود رمز کننده
Auto-encoder	۲۲۲	خود رمز کننده
Average square projection error	۱۵۴	میانگین مربعات خطای افکنش
<b>B</b>		
Back propagation	۱۰۰	پس انتشار
Bagging	۲۳۹	بگینگ
Bayesian network	۲۰۹	شبکه های بیزین
Belief network	۲۲۰	شبکه های باور
Bias	۸۳	بایاس
Boltzman machine	۲۲۳	ماشین بولتزمن
Boosting	۲۴۰	تقویت کننده
Bootstrap	۲۳۷	بوت استرالپ
<b>C</b>		
Centroid	۱۳۵	مرکز (خوشه یا کلاس)
Class	۷۵	دسته
Classifier	۲۱۷	دسته بند
Clique	۲۱۱	کلیک
Cluster	۱۳۴	خوشه
Clustering	۱۳۴	خوشه بندی
Collaborative filtering	۱۸۷	پالایش گروهی
Compressed feature vector	۲۲۲	بردار ویژگی های فشرده
Concept drift	۱۹۲	رانش مفهوم
Conditional probability distribution	۲۱۰	توزیع احتمال شرطی
Confusion matrix	۱۳۲	ماتریس در هم ریختگی
Content based	۱۸۷	بر اساس محتوا

Convex	۴۳	محدب
Convolution layer	۲۳۱	لایه کانولوشن
Convolutional neural network	۲۲۰	شبکه عصبی کانولوشن
Correlation	۱۸۷	همبستگی
Cosine similarity	۱۸۹	شیاهت کسینوسی
Cost function	۲۹	تابع هزینه
Covariance	۱۵۳	کواریانس
Covariance matrix	۱۵۳	ماتریس کوواریانس
Cross validation	۸۷	اعتبار سنجی تقاطعی
Curse of dimensionality	۲۴۱	نفرین ابعاد
<b>D</b>		
Data frame	۷۷	قابل داده
Data set	۲۶	مجموعه داده
Data visualization	۱۵۰	تصویرسازی داده
Decision boundary	۶۶	مرز تصمیم‌گیری
Decoder	۲۲۲	رمزگشا
Deduction	۲۱۲	استنتاج
Deep belief network	۲۲۰	شبکه باور عمیق
Demographic	۱۸۷	جمعیتی
Denial of services	۱۶۲	انکار سرویس
Density estimation	۱۶۵	تخمین چگالی
Density function	۲۲۱	تابع چگالی
Derivative	۲۱۴	مشتق
Discount factor	۱۷۶	عامل تخفیف
Discriminative	۲۱	تمایزی
Discriminative model	۲۲۰	مدل تمایزی
Distribution model	۱۳۵	مدل توزیع
Dynamic Bayesian network	۲۱۰	شبکه‌های پویای بیزین

<b>E</b>		
Elbow	۱۴۲	آرنج
Encoder	۲۲۲	رمز کننده
Ensemble	۲۳۷	مجموع
Epoch	۹۷	دوره
Error	۴۵	خطا
Evolving granular network	۱۹۵	شبکه عصبی داده دانه تغییرپذیر
<b>F</b>		
False negative	۲۰۰	منفی اشتباه
False negative rate	۲۰۱	نرخ منفی اشتباه
False positive	۲۰۱	مثبت اشتباه
False positive rate	۲۰۱	نرخ مثبت اشتباه
Feature scaling	۵۴	مقیاس‌گذاری ویژگی‌ها
Feed forward	۹۹	پیش‌خور
Final class score	۲۳۳	امتیاز نهایی کلاس
Flat	۱۴۸	تخت
Fraud detection	۱۶۱	تشخیص تقلب
Fully connected	۲۳۱	کاملاً متصل
Function	۲۹	تابع
<b>G</b>		
Gaussian distribution	۴۸	توزیع نرمال
Generalization	۸۲	تعیین
Generative model	۲۲۰	مدل تولیدی
Global flat subspace	۱۴۸	زیر فضای تخت عمومی
Gradient descent	۳۷	گرادیان نزولی
Graphical model	۲۰۸	مدل گرافی
<b>H</b>		
Hadoop	۲۴۳	هدوپ
Hard clustering	۱۳۴	خوشبندی سخت
Hidden layer	۱۰۱	لایه مخفی

Hidden markove model	۲۰۹	مدل مخفی مارکوف
Hidden space	۲۲۲	فضای مخفی
Hierarchical Gaussian mixture	۱۹۵	محلوط گوسی سلسله مراتبی
Histogram	۲۷۶	هیستوگرام
Hofding tree	۱۹۵	درخت هافدینگ
Hybrid model	۲۲۰	مدل ترکیبی
Hybrid recommender system	۱۸۸	سیستم توصیه‌گر ترکیبی
Hyper	۲۵۸	فوق
Hypothesis	۴۵	فرضیه
<b>I</b>		
Industrial damage detection	۱۶۱	تشخیص خرابی صنعتی
Inner product	۱۱۸	ضرب داخلی
Input	۲۲۲	ورودی
Input space	۲۲۲	فضای ورودی
Instance	۲۸	نمونه
Intensity	۲۳۰	شدت روشنایی
Item	۱۸۴	آیتم
<b>J</b>		
Joint probability function	۲۲۰	تابع احتمال مشترک
<b>K</b>		
kernel	۱۲۲	هسته
Kernel function	۱۲۲	تابع هسته
k-fold cross validation	۸۷	اعتبارسنجی تقاطعی k گروهی
<b>L</b>		
Land mark	۱۲۵	نشانه
Learning rate	۳۸	نرخ یادگیری
Likelihood	۲۱۵	شباهت
Linear	۶۸	خطی
Linear regression	۴۵	رگرسیون خطی
Local minimum	۳۷	کمینه محلی
Logistic regression	۶۴	رگرسیون لاجستیک

<b>M</b>		
Map	۲۱۲	نگاشت
Margin	۱۱۵	حاشیه
Markov model	۲۰۹	مدل مارکوف
Markov network	۲۰۹	شبکه مارکوف
Max-product	۲۱۶	بیشینه ضرب‌ها
Mean	۵۶	متوسط
Mean normalization	۵۶	نرمالسازی متوسط
Model selection	۹۰	انتخاب مدل
Momentum	۱۰۲	مومنتوم
Multi layer perceptron	۹۹	پرسپیترون چند لایه
<b>N</b>		
Natural language processing	۱۹۲	پردازش زبان طبیعی
Noise	۲۴۱	نویز
None linear	۶۸	غیرخطی
<b>O</b>		
One-verses-one	۷۴	یکی در مقابل یک
One-versus-all	۷۴	یکی در مقابل همه
Over fitting	۸۳	بیش برآش
<b>P</b>		
Parallel processing	۹۴	پردازش موازی
Pearson correlation coefficient	۱۸۹	ضریب همبستگی پیرسون
Perception	۳۷	درک
Polling	۲۱۴	ادغام
Pooling layer	۲۳۱	لایه ادغام
Positive example	۸۲	نمونه مثبت
Posterior probability	۲۱۲	احتمال پسین
Precision	۲۰۱	دقت
Prediction	۲۰	پیش‌بینی
Predictor	۲۰	پیش‌بینی کننده
Principal component analysis	۱۴۸	تجزیه و تحلیل مؤلفه‌های اصلی

Prior probability distribution	۲۰۸	توزیع احتمال پیشین
Probabilistic graphical models	۲۰۸	مدل‌های گرافی احتمالاتی
Projection	۱۱۹	افکنش
<b>Q</b>		
Qlearning	۱۷۹	یادگیری Q
Query	۲۱۲	پرس و جو
<b>R</b>		
Recall	۲۰۱	بازیابی
Receiver operator characteristic	۲۰۲	منحنی مشخصه عملکرد سیستم
Receptive field	۲۲۱	میدان ادراکی
Reconstruction	۲۲۶	بازسازی
Rectified linear unit	۲۲۲	واحد خطی تصحیح شده است
Rectified linear unit	۲۲۲	واحد خطی یکسو شده
Recurrent network	۲۲۰	شبکه‌های بازگشتی
Recurrent network	۲۲۰	شبکه بازگشتی
Recurrent neural network	۲۲۷	شبکه عصبی بازگشتی
Regression	۲۶	رگرسیون
Regularization	۸۸	تنظیم
Reinforcement	۲۰	تقویتی
Reinforcement learning	۲۰	یادگیری تقویتی
Representation	۲۲۸	بازنمایی
Re-Sampling	۲۳۷	نمونه‌برداری مجدد
Restricted boltzman machine	۲۲۴	ماشین بلتزمن محدود
<b>S</b>		
Sample	۴۴	نمونه
Scaling	۵۴	مقیاس‌گذاری
Sensitivity	۲۰۱	حساسیت
Similarity computation	۱۲۵	محاسبه شباهت
Similarity distance	۱۸۹	فاصله شباهت
Simultaneous update	۳۸	بهروزرسانی همزمان

Soft clustering	۱۳۴	خوشه‌بندی نرم
Step function	۹۶	تابع پله‌ای
Stochastic	۲۲۴	اتفاقی
Streaming data	۱۹۲	داده جریانی
Sum product	۲۱۶	مجموع حاصل ضرب
Supervised learning	۲۰	یادگیری با نظارت
Support vector machine	۱۱۲	ماشین بردار پشتیبان
Symmetric bipartite graph	۲۲۶	گراف دوبخشی متقارن
<b>T</b>		
Temporal difference learning	۱۷۳	یادگیری تفاوت زمانی
Test	۴۴	آزمون
Threshold	۶۴	آستانه
Training	۳۰۵	آموزش
Transaction	۱۸۶	تراکنش
True negative rate	۲۰۱	نرخ منفی درست
True positive	۲۰۰	مثبت درست
True positive rate	۲۰۱	نرخ مثبت درست
<b>U</b>		
Under fitting	۸۳	کم برازش
Unsupervised learning	۲۰	یادگیری بدون نظارت
<b>V</b>		
vapnik chervonenkis dimension (VC)	۸۲	بعد ظرفیت فرضیه (VC)
Variance	۸۳	واریانس
Very fast decision tree	۱۹۸	درخت تصمیم خیلی سریع
Voting	۷۴	رأی‌گیری
<b>W</b>		
Weak learner	۲۳۷	یادگیرنده ضعیف
Weight	۱۰۹	وزن

## «فارسی به انگلیسی»

الف		
Elbow	۱۴۲	آرچ
Test	۴۴	آزمون
Threshold	۶۴	آستانه
Training	۳۰۵	آموزش
Item	۱۸۴	آیتم
Stochastic	۲۲۴	اتفاقی
Posterior probability	۲۱۲	احتمال پسین
Polling	۲۱۴	ادغام
Deduction	۲۱۲	استنتاج
Cross validation	۸۷	اعتبارسنجی تقاطعی
Projection	۱۱۹	افکنش
Final class score	۲۳۳	امتیاز نهایی کلاس
Model selection	۹۰	انتخاب مدل
Denial of services	۱۶۲	انکار سرویس
ب		
Reconstruction	۲۲۶	بازسازی
Representation	۲۲۸	بازنمایی
Recall	۲۰۱	بازیابی
Bias	۸۳	بایاس
Content based	۱۸۷	بر اساس محتوا
Compressed feature vector	۲۲۲	بردار ویژگی‌های فشرده
vapnik chervonenkis dimension (VC)	۸۲	بعد ظرفیت فرضیه (VC)
Simultaneous update	۳۸	به روزرسانی همزمان
Bootstrap	۲۳۷	بوت استریپ
Over fitting	۸۳	بیش برازش
Max-product	۲۱۶	بیشینه ضرب‌ها
پ		
Collaborative filtering	۱۸۷	پالایش گروهی

Natural language processing	۱۹۲	پردازش زبان طبیعی
Parallel processing	۹۴	پردازش موازی
Query	۲۱۲	پرس و جو
Multi layer perceptron	۹۹	پرسپترون چند لایه
Back propagation	۱۰۰	پس انتشار
Prediction	۲۰	پیش‌بینی
Predictor	۲۰	پیش‌بینی کننده
Feed forward	۹۹	پیش‌خور
ت		
Function	۲۹	تابع
Joint probability function	۲۲۰	تابع احتمال مشترک
Step function	۹۶	تابع پله‌ای
Density function	۲۲۱	تابع چگالی
Cost function	۲۹	تابع هزینه
Kernel function	۱۲۳	تابع هسته
Principal component analysis	۱۴۸	تجزیه و تحلیل مؤلفه‌های اصلی
Flat	۱۴۸	تخت
Density estimation	۱۶۵	تخمین چگالی
Transaction	۱۸۶	تراکنش
Fraud detection	۱۶۱	تشخیص تقلب
Industrial damage detection	۱۶۱	تشخیص خرابی صنعتی
Anomaly detection	۱۶۰	تشخیص ناهمجارتی
Generalization	۸۲	تعمیم
Boosting	۲۴۰	تقویت کننده
Reinforcement	۲۰	تقویتی
Regularization	۸۸	تنظیم
Prior probability distribution	۲۰۸	توزیع احتمال پیشین
Conditional probability distribution	۲۱۰	توزیع احتمال شرطی
Gaussian distribution	۴۸	توزیع نرمال

<b>ج</b>		
Demographic	۱۸۷	جمعیتی
<b>ح</b>		
Margin	۱۱۵	حاشیه
Sensitivity	۲۰۱	حساسیت
<b>خ</b>		
Error	۴۵	خطا
Linear	۶۸	خطی
Linear	۶۸	خطی
Auto-encoder	۲۲۲	خود رمز کننده
Auto encoder	۲۲۲	خود رمز کننده
Cluster	۱۳۴	خوشه
Clustering	۱۳۴	خوشه بندی
Hard clustering	۱۳۴	خوشه بندی سخت
Soft clustering	۱۳۴	خوشه بندی نرم
<b>د</b>		
Streaming data	۱۹۲	داده جریانی
Very fast decision tree	۱۹۸	درخت تصمیم خیلی سریع
Hofding tree	۱۹۵	درخت هافدینگ
Perception	۳۷	درک
Class	۷۵	دسته
Classifier	۲۱۷	دسته بند
Precision	۲۰۱	دقت
Epoch	۹۷	دوره
<b>ر</b>		
Concept drift	۱۹۲	رانش مفهوم
Voting	۷۴	رأی گیری
Regression	۲۶	رگرسیون
Linear regression	۴۵	رگرسیون خطی
Logistic regression	۶۴	رگرسیون لاجستیک

Encoder	۲۲۲	رمزکننده
Decoder	۲۲۲	رمزگشا
ز		
Global flat subspace	۱۴۸	زیر فضای تخت عمومی
س		
Hybrid recommender system	۱۸۸	سیستم توصیه‌گر ترکیبی
ش		
Likelihood	۲۱۵	شباهت
Cosine similarity	۱۸۹	شباهت کسینوسی
Recurrent network	۲۲۰	شبکه بازگشتی
Deep belief network	۲۲۰	شبکه باور عمیق
Recurrent neural network	۲۲۷	شبکه عصبی بازگشتی
Evolving granular network	۱۹۵	شبکه عصبی داده تغییرپذیر
Convolutional neural network	۲۲۰	شبکه عصبی کانولوشن
Artificial neural network	۹۴	شبکه عصبی مصنوعی
Markov network	۲۰۹	شبکه مارکوف
Hidden markov network	۲۰۹	شبکه مخفی مارکوف
Recurrent network	۲۲۰	شبکه‌های بازگشتی
Belief network	۲۲۰	شبکه‌های باور
Bayesian network	۲۰۹	شبکه‌های بیزین
Dynamic Bayesian network	۲۱۰	شبکه‌های پویای بیزین
Intensity	۲۳۰	شدت روشنایی
ض		
Inner product	۱۱۸	ضرب داخلی
Pearson correlation coefficient	۱۸۹	ضریب همبستگی پیرسون
ع		
Discount factor	۱۷۶	عامل تخفیف
غ		
None linear	۶۸	غیرخطی

ف		
Similarity distance	۱۸۹	فاصله شباهت
Hypothesis	۴۵	فرضیه
Hidden space	۲۲۲	فضای مخفی
Input space	۲۲۲	فضای ورودی
Hyper	۲۵۸	فوق
ق		
Data frame	۷۷	قاب داده
ک		
Fully connected	۲۳۱	کاملاً متصل
Clique	۲۱۱	کلیک
Under fitting	۸۳	کم برازش
Local minimum	۳۷	کمینه محلی
Covariance	۱۵۳	کوواریانس
گ		
Gradient descent	۳۷	گرادیان نزولی
Symmetric bipartite graph	۲۲۶	گراف دوبخشی متقارن
Agent graph	۲۱۵	گراف عامل
k-fold cross validation	۸۷	اعتبارسنجی تقاطعی k گروهی
ل		
Pooling layer	۲۳۱	لایه ادغام
Convolution layer	۲۳۱	لایه کاتولوشن
Hidden layer	۱۰۱	لایه مخفی
م		
Confusion matrix	۱۳۲	ماتریس در هم ریختگی
Confusion matrix	۱۳۲	ماتریس در هم ریختگی
Covariance matrix	۱۵۳	ماتریس کوواریانس
Support vector machine	۱۱۲	ماشین بردار پشتیبان
Restricted boltzman machine	۲۲۴	ماشین بلتزمن محدود
Boltzman machine	۲۲۳	ماشین بولتزمن

Mean	۵۶	متوسط
False positive	۲۰۱	مشتبه
True positive	۲۰۰	مثبت درست
Ensemble	۲۳۷	مجموع
Sum product	۲۱۶	مجموع حاصل ضرب
Data set	۲۶	مجموعه داده
Similarity computation	۱۲۵	محاسبه شباهت
Convex	۴۳	محدب
Hierarchical Gaussian mixture	۱۹۵	مخلوط گوسی سلسله مراتبی
Hybrid model	۲۲۰	مدل ترکیبی
Discriminative model	۲۲۰	مدل تمایزی
Distribution model	۱۳۵	مدل توزیع
Generative model	۲۲۰	مدل تولیدی
Graphical model	۲۰۸	مدل گرافی
Markov model	۲۰۹	مدل مارکف
Probabilistic graphical models	۲۰۸	مدلهای گرافی احتمالی
Decision boundary	۶۶	مرز تصمیم‌گیری
Centroid	۱۳۵	مرکز (خوشه یا کلاس)
Derivative	۲۱۴	مشتق
Data visualization	۱۵۰	تصویرسازی داده
Inversion	۲۵۰	معکوس
Feature scaling	۵۴	مقیاس‌گذاری ویژگی‌ها
Scaling	۵۴	مقیاس‌گذاری
Receiver operator characteristic	۲۰۲	منحنی مشخصه عملکرد سیستم
False negative	۲۰۰	منفی اشتباه
Momentum	۱۰۱	مومنتوم
Average square projection error	۱۵۴	میانگین مربعات خطای افکنش
Condition random field	۲۰۹	میدان اتفاقی شرطی
Receptive field	۲۳۱	میدان ادراکی
ن		
False positive rate	۲۰۱	نرخ مثبت اشتباه
True positive rate	۲۰۱	نرخ مثبت درست

False negative rate	۲۰۱	نرخ منفی اشتباه
True negative rate	۲۰۱	نرخ منفی درست
Learning rate	۳۸	نرخ یادگیری
Mean normalization	۵۶	نرمالسازی متوسط
Land mark	۱۲۵	نشانه
Curse of dimensionality	۲۴۱	نفرین ابعاد
Map	۲۱۲	نگاشت
Sample	۴۴	نمونه
Re-Sampling	۲۳۷	نمونه برداری مجدد
Positive example	۸۲	نمونه مثبت
Noise	۲۴۱	نویز
—۵—		
Hadoop	۲۴۳	هدوپ
kernel	۱۲۳	هسته
Correlation	۱۸۷	همبستگی
Histogram	۲۷۶	هیستوگرام
۶		
Rectified linear unit	۲۳۲	واحد خطی یکسو شده
Variance	۸۳	واریانس
Input	۲۲۲	ورودی
Weight	۳۷	وزن
۷		
Q-learning	۱۷۹	یادگیری Q
Weak learner	۲۳۷	یادگیرنده ضعیف
Supervised learning	۲۰	یادگیری با ناظرت
Unsupervised learning	۲۰	یادگیری بدون ناظرت
Temporal difference learning	۱۷۲	یادگیری تفاوت زمانی
Reinforcement learning	۲۰	یادگیری تقویتی
One-versus-all	۷۴	یکی در مقابل همه
One-verses-one	۷۴	یکی در برابر یکی

## مراجع و منابع

- [1] Andrew ng, "machine learning" standford university video lectures 2009 available at: [http://videolectures.net/stanfordcs229f07\\_machine\\_learning/](http://videolectures.net/stanfordcs229f07_machine_learning/)
- [2] Alpaydin, "Ethem. Introduction to machine learning", MIT press, 2014.
- [3] Schapire, Rob. "Machine learning algorithms for classification." Princeton University 10 2015.
- [4] James, Gareth, et al. "An introduction to statistical learning", Vol. 6. New York, springer, 2013.
- [5] Bishop, Christopher M. "Pattern recognition and Machine Learning", springer, 2006.
- [6] Lantz, Brett., "Machine learning with R", Packt Publishing Ltd, 2013.
- [7] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1, pp.1-127, 2009.
- [8] H. Anton, " Elementary Linear Algebra" 5e, John Wiley & Son Inc, 1987.
- [9] Bishop, Christopher, John Winn, and David J. Spiegelhalter. "Variational inference engine for probabilistic graphical models." U.S. Patent No. 6,556,960. 29 Apr. 2003.
- [10] Nowozin, Sebastian, and Christoph H. Lampert. "Structured learning and prediction in computer vision." Foundations and Trends® in Computer Graphics and Vision 6.3–4, pp.185-365, 2011.
- [11] Sutton, Charles, and Andrew McCallum. "An introduction to conditional random fields." Foundations and Trends® in Machine Learning 4.4, pp.267-373, 2012.
- [12] Getoor, Lise. "Introduction to statistical relational learning". MIT press, 2007.
- [13] Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining, "Introduction to linear regression analysis. ", John Wiley & Sons, 2015.
- [14] Berger, James O. "Statistical Decision Theory and Bayesian Analysis (Second edition. ed.)". New York, NY: Springer New York. ISBN 9781475742862, 1985.
- [15] Efron, Bradley. "Bayes' theorem in the 21st century." Science 340.6137, 2013.
- [16] Chandola V., Banerjee A., Kumar V., "Anomaly detection: A survey", ACM Computing Surveys, vol.41, No.3, p. 15, 2009.
- [17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436-444, 2015.
- [18] Y. Bengio, "Learning deep architectures for AI," Foundations and trends® in Machine Learning, vol. 2, pp. 1-127, 2009.

- [19] F. Rosenbaltt, "The Perceptron—a Perciving and Recognizing Automation," Report 85-460-1 Cornell Aeronautical Laboratory, Ithaca, 1957.
- [20] Hinton, G. "Deep belief networks". Scholarpedia. 4 (5): 5947. doi:10.4249/scholarpedia. 5947, 2009.
- [21] Hinton, G. E.; Osindero, S.; Teh, Y. W. , "A Fast Learning Algorithm for Deep Belief Nets" , Neural Computation. 18 (7); pp. 1527–1554, 2006.
- [22] Bengio, Yoshua, et al. "Greedy layer-wise training of deep networks." Advances in neural information processing systems 19:153, 2007.
- [23] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks", In Advances in Neural Information Processing Systems, 2007.
- [24] Singla, Parag, and Pedro Domingos. "Discriminative training of Markov logic networks." AAAI. Vol. 5. 2005.
- [25] Andrey Kurenkov, "A brief History of Neural Nets and Deep Learning". [<http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>]
- [26] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11, pp. 2278-2324, 1998.
- [27] Sethi, Ishwar K., and Anil K. Jain, eds. "Artificial neural networks and statistical pattern recognition: old and new connections". Elsevier, Vol. 11, 2014.
- [28] Clark, Christopher, and Amos Storkey. "Teaching deep convolutional neural networks to play go." arXiv preprint arXiv:1412.3409, 2014.
- [29] Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Recommender systems: Introduction and challenges." Recommender Systems Handbook. Springer US, pp.1-34, 2015.
- [30] Hofmann, M., and Klinkenberg, R, "RapidMiner: Data mining use cases and business analytics applications", CRC Press, 2013.
- [31] Leskovec, J., Rajaraman, A., and Ullman, J. D. "Mining of massive datasets", Cambridge University Press, 2014.
- [32] Aggarwal, C. C. "Data streams: An overview and scientific applications. in Scientific Data Mining and Knowledge Discovery", Springer, pp. 397–377, 2009.
- [33] Ricci, Francesco, Lior Rokach, and Bracha Shapira". Introduction to recommender systems handbook", springer US, 2011.
- [34] Geurts, Pierre, and L. Wehenkel. "An introduction to machine learning." Slide from Ulg Applied Inductive Learning lectures. iv 10, 2009.
- [35] Chase, Henry W., et al. "Reinforcement learning models and their neural correlates: an activation likelihood estimation meta-analysis." Cognitive, affective, & behavioral neuroscience 15.2, pp.435-459, 2015.

- [36] Schapire, Robert E. "The boosting approach to machine learning: An overview." Nonlinear estimation and classification. Springer New York, pp.149-171, 2003.
- [37] Gandrud, Christopher. "Reproducible research with R and R studio", CRC Press, 2013.
- [38] Venables, W. N., and D. M. Smith. "An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics Version 3.1",the R Core Team, 07-10, 2014.
- [39] Kuhn, M., et al. "caret: Classification and regression training. R package version 6.0-21." CRAN: Wien, Austria, 2014.
- [40] Team, R. Core. "R language definition." Vienna, Austria: R foundation for statistical computing, 2000.
- [41] Team, RStudio. "RStudio: Integrated Development for R (RStudio, Inc., Boston, MA, 2015)." URL: <https://www.rstudio.com/products/rstudio>, 2015.
- [42] Jones, Owen, Robert Maillardet, and Andrew Robinson., "Introduction to scientific programming and simulation using R",. CRC Press, 2014.
- [43] Sokolova, Marina, Nathalie Japkowicz, and Stan Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation." Australasian Joint Conference on Artificial Intelligence. Springer Berlin Heidelberg, 2006.
- [44] Altman DG, Bland JM. "Diagnostic tests 3: receiver operating characteristic plots." British Medical Journal, 1994.