**فصل 13- رگرسیون خطی**

**13.0 مقدمه**

رگرسیون خطی یکی از ساده‌ترین الگوریتم‌های یادگیری تحت نظارت در جعبه ابزار ما است. اگر تا به حال یک دوره مقدماتی آمار را در کالج گذرانده باشید، احتمالاً آخرین موضوعی که به آن پرداخته‌اید رگرسیون خطی بوده است. در واقع، آنقدر ساده است که گاهی اوقات اصلاً به عنوان یادگیری ماشین در نظر گرفته نمی‌شود! باور داشته باشید یا خیر، واقعیت این است که رگرسیون خطی - و بسط‌های آن - زمانی که بردار هدف، مقدار کمی ‌است (به عنوان مثال، قیمت خانه، سن) یک روش رایج و مفید برای پیش‌بینی است. در این فصل ما انواع روش‌های رگرسیون خطی (و برخی بسط‌های آن) را برای ایجاد مدل‌های پیش‌بینی با عملکرد خوب بررسی خواهیم کرد.

**13.1 تطبیق یک خط**

**مسئله**

می‌خواهید مدلی را آموزش دهید که نشان دهنده رابطه خطی بین بردار ویژگی و بردار هدف باشد.

**راه‌حل**

از LinearRegression در scikit-learn استفاده کنید:



**بحث**

رگرسیون خطی فرض می‌کند که رابطه بین ویژگی‌ها و بردار هدف تقریباً خطی است. یعنی اثر (که ضریب، وزن یا پارامتر نیز نامیده می‌شود) ویژگی‌ها بر بردار هدف ثابت است. در راه‌حل ما، برای توضیح، مدل خود را تنها با استفاده از دو ویژگی آموزش داده‌‌ایم. این بدان معناست که مدل خطی ما به صورت زیر خواهد بود:

که در آن هدف ما است، ابعاد مختلف ویژگی ، و ضرایبی هستند که با آموزش مدل شناسایی می‌شوند، و خطا است.

پس از اینکه مدل خود را آموزش دادیم، می‌توانیم مقدار هر پارامتر را مشاهده کنیم. به عنوان مثال، ، که به آن سوگیری(bias) یا رهگیری (intercept) نیز گفته می‌شود، می‌تواند با استفاده از intercept\_ مشاهده شود:





و و به عنوان coef\_ نشان داده می‌شوند:





در مجموعه داده (دیتاست) ما، مقدار هدف، ارزش متوسط یک خانه در بوستون (در دهه 1970) به هزار دلار است. بنابراین قیمت خانه اول در مجموعه داده عبارت است از:





که با استفاده از روش پیش‌بینی، می‌توانیم مقداری را برای این خانه پیش‌بینی کنیم:





بد نیست! مدل ما فقط 560.24 دلار تخفیف (تفاوت قیمت) داشت!

مزیت اصلی رگرسیون خطی تفسیرپذیری آن است، تا حد زیادی به این دلیل که ضرایب مدل، اثر یک تغییر یک واحدی بر بردار هدف هستند. به عنوان مثال، اولین ویژگی در راه‌حل ما تعداد جرایم به ازای هر فرد ساکن در خانه است. ضریب مدل ما برای این ویژگی بود، به این معنی که اگر این ضریب را در 1000 ضرب کنیم (از آنجایی که بردار هدف قیمت خانه به هزار دلار است)، تغییر قیمت خانه را برای هر جرم سرانه اضافی خواهیم داشت:





این عبارت می‌گوید که سرانه هر جنایت، قیمت خانه را تقریباً 350 دلار کاهش می‌دهد!

**13.2 مدیریت تاثیرات تعاملی**

**مسئله**

شما یک ویژگی دارید که تأثیر آن بر متغیر هدف، به ویژگی دیگری بستگی دارد.

**راه‌حل**

با استفاده از ویژگی‌های چندجمله‌ای scikit-learn یعنی PolynomialFeatures یک اصطلاح تعاملی[[1]](#footnote-1) ایجاد کنید تا این وابستگی را نشان دهد:



**بحث**

گاهی اوقات تأثیر یک ویژگی بر متغیر هدف ما حداقل تا حدی به ویژگی دیگری وابسته است. به عنوان مثال، یک مثال ساده مبتنی بر قهوه را تصور کنید که در آن ما دو ویژگی دوتایی داریم - وجود شکر (قند) و اینکه آیا هم زده‌ایم (هم‌زده) یا نه - و می‌خواهیم پیش‌بینی کنیم که آیا قهوه طعم شیرینی دارد یا خیر. فقط گذاشتن شکر در قهوه (شکر=1، هم‌زدن=0) طعم قهوه را شیرین نمی‌کند (همه شکر تَهِ لیوان است!) و فقط هم‌زدن قهوه بدون افزودن شکر (شکر=0، هم زده=1) هم آن را شیرین نمی‌کند. در عوض، این تعامل بین قرار دادن شکر در قهوه و هم‌زدن قهوه (شکر=1، هم‌زدن=1) است که طعم قهوه را شیرین می‌کند. تاثیر شکر و هم زدن بر شیرینی به یکدیگر بستگی دارد. در این مورد می‌گوییم اثر متقابلی بین ویژگی‌های شکر و هم‌زدن وجود دارد.

ما می‌توانیم اثرات متقابل را با گنجاندن یک ویژگی جدید که حاصل ارزش‌های متناظر از ویژگی‌های متقابل را تشکیل می‌دهد، توضیح دهیم:

که در آن و به ترتیب مقادیر شکر و هم‌زدن هستند و نشان دهنده تعامل بین این دو است.

در این راه حل، ما از مجموعه داده‌ای استفاده کردیم که فقط شامل دو ویژگی بود. در اینجا مقادیر اولین مشاهده برای هر یک از آن ویژگی‌ها آمده است:





برای ایجاد یک عبارت تعاملی، ما به سادگی آن دو مقدار را برای هر مشاهده ضرب می‌کنیم:



سپس می‌توانیم عبارت تعامل برای اولین مشاهده را مشاهده کنیم:





با این حال، در حالی که اغلب ما دلیلی اساسی برای باور وجود تعامل بین دو ویژگی خواهیم داشت، گاهی اوقات چنین نیست. در این موارد استفاده از ویژگی‌های چند جمله ای (PolynomialFeatures) در scikit-learn برای ایجاد اصطلاحات تعاملی برای همه‌ی ترکیب‌های ویژگی‌ها می‌تواند مفید باشد. سپس می‌توانیم از استراتژی‌های انتخاب مدل برای شناسایی ترکیبی از ویژگی‌ها و اصطلاحات تعاملی که بهترین مدل را تولید می‌کنند، استفاده کنیم.

برای ایجاد اصطلاحات تعاملی با استفاده از PolynomialFeatures، سه پارامتر مهم وجود دارد که باید تنظیم کنیم. مهم‌تر از همه، interaction\_only=True به PolynomialFeatures می‌گوید که فقط عبارات تعامل را برگرداند (و نه ویژگی‌های چند جمله‌ای، که در دستور العمل 13.3 در مورد آن صحبت خواهیم کرد). به طور پیش‌فرض، PolynomialFeatures ویژگی‌هایی به نام سوگیری را اضافه می‌کند. ما می‌توانیم با include\_bias=False از آن جلوگیری کنیم. در نهایت، پارامترِ درجه، حداکثر تعداد ویژگی‌ها را برای ایجاد عبارت‌های تعاملی تعیین می‌کند (این مورد در صورتی استفاده می‌شود که می‌خواهیم یک عبارت تعاملی ایجاد کنیم که آن عبارت تعاملی ترکیبی از سه ویژگی باشد). ما می‌توانیم خروجی PolynomialFeatures از راه‌حل خود را با بررسی اینکه آیا مقادیر ویژگی اولین مشاهده و مقدار مدت تعامل با نسخه محاسبه‌شده دستی ما مطابقت دارد یا خیر، ببینیم:





**13.3 آموزش یک رابطه غیرخطی**

**مسئله**

شما می‌خواهید یک رابطه غیرخطی را مدل کنید.

**راه‌حل**

یک رگرسیون چند جمله‌ای با گنجاندن ویژگی‌های چند جمله‌ای در مدل رگرسیون خطی ایجاد کنید:



**بحث**

تا اینجا ما فقط در مورد مدل‌سازی روابط خطی بحث کردیم. نمونه‌ای از یک رابطه خطی می‌تواند تعداد طبقات یک ساختمان و ارتفاع ساختمان باشد. در رگرسیون خطی، ما تأثیر تعداد طبقات و ارتفاع ساختمان را تقریباً ثابت فرض می‌کنیم، به این معنی که ارتفاع یک ساختمان 20 طبقه تقریباً دو برابر یک ساختمان 10 طبقه است که تقریباً دو برابر بلندتر از یک ساختمان 5 طبقه است. با این حال، بسیاری از روابط واقعی کاملاً خطی نیستند.

اغلب ما می‌خواهیم یک رابطه غیر‌خطی را مدل کنیم - برای مثال، رابطه بین تعداد ساعات مطالعه یک دانش‌آموز و نمره‌ای که در آزمون می‌گیرد. به طور شهودی، می‌توانیم تصور کنیم که تفاوت زیادی در نمرات آزمون بین دانش‌آموزانی که یک ساعت مطالعه می‌کنند در مقایسه با دانش‌آموزانی که اصلا مطالعه نکرده‌اند وجود دارد. با این حال، بین دانش‌آموزی که 99 ساعت درس خوانده و دانش‌آموزی که 100 ساعت مطالعه کرده است، تفاوت بسیار کمتری در نمرات آزمون وجود دارد. تأثیر یک ساعت مطالعه بر نمره آزمون دانش‌آموز با افزایش تعداد ساعات کاهش می‌یابد.

رگرسیون چند جمله‌ای توسعه‌ای از رگرسیون خطی است تا به ما امکان مدل‌سازی روابط غیرخطی را بدهد. برای ایجاد یک رگرسیون چند جمله‌ای، تابع خطی را که در دستور العمل 13.1 استفاده کردیم تبدیل کنید:

که با افزودن ویژگی‌های چند جمله‌ای به یک تابع چند جمله‌ای می‌رسیم:

که در آن d درجه چند جمله‌ای است. چگونه می‌توانیم از رگرسیون خطی برای یک تابع غیرخطی استفاده کنیم؟ پاسخ این است که ما نحوه تناسب رگرسیون خطی با مدل را تغییر نمی‌دهیم، بلکه فقط ویژگی‌های چند جمله‌ای را به آن اضافه می‌کنیم. یعنی رگرسیون خطی «نمی‌داند» که تبدیل درجه دوم است. فقط آن را یک متغیر دیگر در نظر می‌گیرد.

ممکن است توضیح کاربردی‌تری لازم باشد. برای مدل‌سازی روابط غیرخطی، می‌توانیم ویژگی‌های جدیدی ایجاد کنیم که یک ویژگی موجود، را تا یک مقدار توانی افزایش می‌دهد: ، ، و غیره. هر چه بیشتر از این ویژگی‌های جدید اضافه کنیم، «خط» ایجاد شده توسط مدل ما انعطاف‌پذیرتر است. برای واضح‌تر شدن این موضوع، تصور کنید که می‌خواهیم یک چندجمله‌ای تا درجه سوم ایجاد کنیم. به منظور سادگی، ما فقط بر روی یک مشاهده (اولین مشاهده در مجموعه داده) تمرکز خواهیم کرد، :





برای ایجاد یک ویژگی چندجمله‌ای، مقدار مشاهده اول را به درجه دوم یعنی افزایش می‌دهیم:





این ویژگی جدید ما خواهد بود. سپس مقدار مشاهده اول را به درجه سوم یعنی نیز افزایش می‌دهیم:





با گنجاندن هر سه ویژگی (، و ) در ماتریس ویژگی‌ها و سپس اجرای یک رگرسیون خطی، یک رگرسیون چند جمله‌ای انجام داده‌ایم:





PolynomialFeatures دو پارامتر مهم دارد. ابتدا، درجه حداکثر تعداد درجه را برای ویژگی‌های چندجمله‌ای تعیین می‌کند. برای مثال degree = 3، و را ایجاد می‌کند. در نهایت، به طور پیش‌فرض، PolynomialFeatures شامل یک ویژگی حاوی تنها یک‌ها[[2]](#footnote-2) می‌شود (به نام سوگیری). ما می‌توانیم آن را با تنظیم include\_bias=False حذف کنیم.

**13.4 کاهش واریانس با منظم سازی**

**مسئله**

شما می‌خواهید واریانس مدل رگرسیون خطی خود را کاهش دهید.

**راه‌حل**

از یک الگوریتم یادگیری استفاده کنید که شامل جریمه انقباض[[3]](#footnote-3) (همچنین منظم سازی هم نامیده می‌شود) است، مانند روش‌های ridge regression و lasso regression:



**بحث**

در رگرسیون خطی استاندارد، مدل برای به حداقل رساندن مجموع مجذور خطا بین مقادیر واقعی () و پیش‌بینی، () مقادیر هدف، یا مجموع باقیمانده مربع‌ها (RSS) آموزش می‌بیند:

روش‌های یادگیری رگرسیون که تنظیم شده هستتد نیز مشابه همین هستند، با این تفاوت که سعی می‌کنند RSS و مقدار جریمه برای اندازه کل مقادیر ضرایب را به حداقل برسانند، که جریمه انقباض نامیده می‌شود؛ زیرا سعی می‌کند مدل را «کوچک» کند. دو نوع متداول از یادگیرندگان منظم برای رگرسیون خطی وجود دارد: رگرسیونridge و lasso. تنها تفاوت رسمی‌ در نوع جریمه انقباض مورد استفاده است. در رگرسیون ridge، جریمه انقباض یک ابرپارامتر تنظیم ضرب در مجذور همه ضرایب است:

که در آن ضریب ام ویژگی p و α یک ابرپارامتر است (در ادامه بحث می‌شود). Lasso نیز مشابه همین است، با این تفاوت که جریمه انقباض یک ابرپارامتر تنظیم ضرب در مجموع قدر مطلق همه ضرایب است:

که در آن n تعداد مشاهدات است. پس از کدام یک استفاده کنیم؟ به عنوان یک قاعده کلی، رگرسیون ridge اغلب پیش‌بینی‌های کمی ‌بهتری نسبت به lasso ایجاد می‌کند، اما lasso (به دلایلی که در دستور العمل 13.5 در مورد آن صحبت خواهیم‌کرد) مدل‌های قابل تفسیرتری تولید می‌کند. اگر بخواهیم بین توابع جریمه ridge و lasso تعادل برقرار کنیم، می‌توانیم از شبکه الاستیک (elastic net) استفاده کنیم که به سادگی یک مدل رگرسیونی است که هر دو جریمه را شامل می‌شود. صرف نظر از اینکه از کدام یک استفاده می‌کنیم، هر دو رگرسیون ridge و lasso می‌توانند مدل‌های بزرگ یا پیچیده را با گنجاندن مقادیر ضرایب در تابع زیانی که در تلاش برای به حداقل رساندن آن هستیم، جریمه کنند.

ابرپارامتر α، به ما اجازه می‌دهد تا میزان جریمه کردن ضرایب را با استفاده از مقادیر بالای ساخت مدل‌های ساده کنترل کنیم. مقدار ایده آل α باید مانند هر‌ ابرپارامتر دیگری تنظیم شود. در scikit-learn، α با استفاده از پارامتر آلفا تنظیم می‌شود.

scikit-learn شامل یک روش RidgeCV است که به ما امکان می‌دهد مقدار ایده آل α را انتخاب کنیم:





سپس می‌توانیم به راحتی مقدار α برای بهترین مدل را مشاهده کنیم:





نکته پایانی: چون در رگرسیون خطی مقدار ضرایب تا حدی توسط مقیاس ویژگی تعیین می‌شود و در مدل‌های منظم شده همه ضرایب با هم جمع می‌شوند، باید قبل از آموزش از استانداردسازی ویژگی اطمینان حاصل کنیم.

**13.5 کاهش ویژگی‌ها با رگرسیون lasso**

**مسئله**

شما می‌خواهید مدل رگرسیون خطی خود را با کاهش تعداد ویژگی‌ها ساده کنید.

**راه‌حل**

از رگرسیون lasso استفاده کنید:



**بحث**

یکی از ویژگی‌های جالب جریمه رگرسیون lasso این است که می‌تواند ضرایب یک مدل را به صفر کاهش دهد و به طور موثر تعداد ویژگی‌های مدل را کاهش دهد. به عنوان مثال، در حل ما α را روی 0.5 قرار می‌دهیم و می‌بینیم که بسیاری از ضرایب 0 هستند، به این معنی که ویژگی‌های مربوط به آنها در مدل استفاده نمی‌شوند:





با این حال، اگر α را به مقدار بسیار بالاتری افزایش دهیم، می‌بینیم که به معنای واقعی کلمه از هیچ یک از ویژگی‌ها استفاده نمی‌شود:





مزیت عملی این اثر این است که ما می‌توانیم 100 ویژگی را در ماتریس ویژگی‌های خود بگنجانیم و سپس با تنظیم ابرپارامتر α برای lasso، مدلی تولید کنیم که تنها از 10 (مثلا) ویژگی از مهمترین ویژگی‌ها استفاده می‌کند. این روش به ما امکان می‌دهد واریانس را کاهش دهیم و در عین حال تفسیرپذیری مدل خود را بهبود ببخشیم (زیرا ویژگی‌های کمتر توضیح را آسان‌تر می‌کند).

1. - Interaction term [↑](#footnote-ref-1)
2. - ones [↑](#footnote-ref-2)
3. - shrinkage penalty [↑](#footnote-ref-3)