پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

آشنایی با مدل‌های گرافیکی در نرم‌افزار R

محمدرضا یداللهی

استاد راهنما: زهرا رضائی قهرودی

۱۴۰۲

فهرست مطالب

خلاصه

شبکه‌های بیزی ابزاری برای مطالعه داده‌ها هستند. شبکه بیزی با ایجاد یک ساختار گرافیکی برای مدل‌سازی داده‌ها، به داده‌ها ساختار می‌دهد. سپس توزیع‌های احتمال را بر روی این متغیرها توسعه می‌دهد. این شبکه متغیرهای فضای مسئله را بررسی می‌کند و توزیع‌های احتمال مرتبط با آنها را مورد بررسی قرار می‌دهد. این شبکه با استفاده از استنباط آماری بر روی این توزیع‌های احتمال، معنا و مفهومی از آنها استخراج می‌کند. این روش‌ها برای بررسی مجموعه‌ی بزرگی از داده‌ها به صورت کارآمد به کار می‌روند تا استنباط‌ها را انجام دهند. تعدادی از کاربردهای واقعی این شبکه‌ها در دسترس هستند و همچنان در حال تحقیقات فعال هستند. این پروژه به بحث درباره‌ی نظریه و کاربردهای شبکه‌های بیزی و به برخی از مدل‌سازی‌های گرافیکی با استفاده از نرم‌افزار R می‌پردازد.

۱

مقدمه

مدل‌های گرافیکی چارچوب‌های مفیدی برای استنتاج و کشف روابط علّی بخصوص زمانی که با بُعد بالایی از متغیرها روبرو هستیم، فراهم می‌کنند. یک شبکه بیزی [۱] یک گراف غیرمدور جهت‌دار [۲] (باسو)(مسیری برای بازگشت به رئوس وجود نداشته باشد) است ، که مجموعه‌ای از متغیرهای تصادفی و نحوه ارتباط آن‌ها را نشان می‌دهد. یک شبکه بیزی با ایجاد یک سیستم گرافیکی برای مدل‌بندی داده‌ها، به داده‌ها ساختار می‌دهد. سپس توزیع‌های احتمال را روی این متغیرها توسعه می‌دهد. بررسی ارتباط بین علت بیماری‌ها با یکدیگر و تشخیص و برآورد احتمال یک بیماری خاص در یک فرد بیمار، یکی از کاربردهای استفاده از شبکه بیزی است.

در شبکه بیزی یا گراف غیرمدور باسو، رابطه علت و معلولی بین متغیرها از طریق مفهوم یادگیری تعیین می‌شود. در شبکه بیزی، یک گراف با استفاده از یادگیری فرایندها به صورت خودکار ایجاد می‌شود و عملکرد خود را بهبود می‌بخشد. یادگیری شبکه بیزی فرایندی دو مرحله‌ای است که شامل یادگیری ساختاری و یادگیری پارامتری است. در یادگیری ساختاری، یک شبکه بین متغیرهای تصادفی تعیین می‌شود تا روابط علی و معلولی که بیشترین تطابق را با داده‌ها داشته باشد، تعیین شود. هنگامی که بهترین ساختار گراف از نظر موقعیت گره‌ها و رابطه بین آن‌ها به دست آمد،

[۱] Bayesian network

[۲] DAG

در یادگیری پارامتری، براساس محتمل‌ترین ساختار، روابط علی و معلولی با استفاده از یادگیری پارامتری کمی‌سازی می‌شود و به براورد پارامترها پرداخته می‌شود.

تصور کنید که در حال انجام یک مطالعه هستید و برخی از داده‌ها ناقص هستند. یا تصور کنید در یک موقعیتی قرار دارید که تصمیم می‌گیرید یک بُعد جدید را به داده‌های خود اضافه کنید، اما قادر به بازگشت و پر کردن این خلاها در داده‌های گذشته نیستید زیرا این متغیرهای جدید در مجموعه داده قبلی ثبت نشده‌اند. در چنین موقعیتی، چگونه درباره داده‌های ناقص و قدیمی استدلال کنید؟ آیا آنها را ناکارآمد تلقی می‌کنید و رد می‌کنید؟ یا سعی می‌کنید با استدلال درباره آنها به نتایجی برسید؟ حتی اگر این استنباط‌ها کامل بودن کار را تضمین نکند، می‌توانید با درجه‌ای از اطمینان موضوعات را تفسیر و تحلیل کنید. این موضوع امیدوارکننده است. در چنین شرایطی، شبکه‌های بیزی می‌توانند به شما کمک کنند.

شبکه‌های بیز یک زیرحوزه در هوش مصنوعی [۳] است که به سرعت محبوبیت پیدا کرده است. این یک حوزه فعال در تحقیقات دانشگاهی و صنعتی است، زیرا قدرت آن در بهره‌برداری از داده‌ها شناخته شده است. تعداد زیادی از کاربردهای عملی شبکه‌های بیزی در قابلیت صنعتی کشف شده است. این منجر به این شده است که تعدادی زیادی از شرکت‌ها و پژوهشگران شبکه‌های بیز را برای پاسخ به سوالات مختلفی که در برابر آنها قرار دارند، استفاده کنند.

[۳] Artificial Intelligence

شبکه‌های بیز از نظریه گراف برای مدل‌سازی ساختار یک مسئله استفاده می‌کنند. گره‌ها همراه با توپولوژی شبکه، متغیرها را در فضای مسئله و روابط بین آنها رمزگذاری می‌کنند. در نظریه احتمال، به خصوص آمار بیزی و آمار استنباطی، برای کشف و کُدبندی کردن درجاتی که این روابط در فضای مسئله دارند، استفاده می‌کند.

شبکه‌های بیزی به ویژه در "کدگذاری دانش تخصصی نامشخص در سامانه‌های خبره" [۴] مفید هستند (هکرمن، ۲۰۰۸). طراحان مدل داده و شبکه بیزی می‌توانند مدل‌ها را برای نمایش سامانه‌های خبره ایجاد کنند و به شبکه اجازه دهند از داده‌های موجود یاد بگیرد. از تمام داده‌های موجود استفاده می‌کند تا بهترین برآوردها را درباره یک مسئله ارائه دهد.

مدل‌های گرافیکی در شکل مدرن خود از اواخر دهه ۱۹۷۰ وجود داشته و امروزه در بسیاری از زمینه‌های علوم ظاهر شده است. همراه با پیشرفت‌های مداوم مدل‌های گرافیکی، تعدادی از برنامه‌های نرم‌افزاری مدل‌سازی گرافیکی مختلف در طول سال‌ها نوشته شده‌اند. در سال‌های اخیر، بسیاری از این پیشرفت‌های نرم‌افزاری در نرم‌افزار R صورت گرفته است.

در این پروژه، به بررسی نظریه شبکه‌های بیزی می‌پردازیم و کاربردهای عملی این نظریه را بررسی می‌کنیم. در بخش ۲، مفاهیم نظری مورد نیاز برای درک شبکه‌های بیزی را بررسی می‌کنیم؛ به طور خاص، نظریه قضیه بیز و استنتاج بیزی را مورد مطالعه قرار می‌دهیم و سپس به مفاهیم نظریه گراف می‌پردازیم. بخش ۳ مفصل‌تر درباره شبکه بیزی، نحوه استفاده عمومی آن و موقعیت‌هایی که ویژه استفاده می‌شود، بحث می‌کند. سپس، در بخش ۴ به برخی از مدل‌سازی‌های گرافیکی با استفاده از نرم‌افزار R و ویژگی‌های اصلی برخی از این بسته‌ها پرداخته می‌شود.

[۴] encoding uncertain expert knowledge in expert systems

۲

تئوری شبکه‌های بیزی و مدل‌های گرافیکی

شبکه‌های بیزی از نظریه احتمال و نظریه گراف برای جستجو در یک فضای حالت و تصمیم‌گیری در شرایط عدم قطعیت استفاده می‌کنند. از نظریه احتمال برای پیدا کردن سریع‌تر وضعیت‌های هدف استفاده می‌کنند. بنابراین، مفاهیم نظری اصلی که برای پیاده‌سازی یک شبکه بیزی نیاز داریم، در حوزه نظریه احتمال (به ویژه قضیه بیز) و نظریه گراف قرار دارند.

۱.۲ قضیه بیز و استنباط بیزی

۱.۱.۲ قضیه بیز

قضیه بیز توسط پیشکسوتی به نام توماس بیز (ریاضیدان) توسعه یافته است. این قضیه یک روش محاسبه توزیع احتمال شرطی است که با معلوم بودن مجموعه‌ای از متغیرهای تعاملی عمل می‌کند. قضیه بیز به صورت زیر بیان می‌شود:

$$P(H|E,c) = \frac{P(H|c) \times P(E|H,c)}{P(E|c)} \qquad (۱.۲)$$

که در آن H فرضیه، E شواهد معلوم و c اطلاعات پس زمینه است. به طور کلی، $P(A|B)$ به معنای "احتمال A با معلوم بودن B است"، که در آن A متغیر وابسته و B متغیر مستقل است.

قضیه بیز امکان محاسبه احتمال فرض H با معلوم بودن شواهد و اطلاعات پس زمینه را می‌دهد. به عبارت دیگر، با معلوم بودن اطلاعات پس زمینه c و شواهد E ، می‌توانیم احتمال فرضیه H را با استفاده از احتمال شرطی H به شرط c ،احتمال شرطی E با شرط دانستن H و c و احتمال E به شرط دانستن c را با استفاده از رابطه ۱.۲ محاسبه کنیم. در اینجا، عبارت $P(H|E,c)$ احتمال H با معلوم بودن احتمال E و c است و به آن احتمال پسین [۱] می‌گویند. عبارت $P(H|c)$ به احتمال فرضیه با معلوم بودن اطلاعات پس زمینه بدون در نظر گرفتن شواهدی که داریم اشاره دارد و به آن احتمال پیشین [۲] می‌گویند. عبارت$P(E|H,c)$ احتمال شواهد با فرض درست بودن فرضیه و اطلاعات زمینه را می‌دهد. این عبارت به عنوان درستنمایی [۳] شناخته شده است. عبارت$P(E|c)$ احتمال شواهد با معلوم بودن پس زمینه است. این یک عامل مقیاس‌بندی برای به دست آوردن احتمال فرضیه به شرط شواهد E و اطلاعات پس زمینه c است. (نیدرمایر، ۲۰۰۸).

فرمول اغلب به فرم زیر ساده می‌شود

$$P(H|E) = \frac{P(H) \times P(E|H)}{P(E)} \qquad (۲.۲)$$

این یک جایگزین قابل قبول است زیرا اطلاعات پس‌زمینه اغلب تغییر نمی‌کند و بنابراین می‌توانیم آن را در طول تحلیل خود ثابت فرض کنیم (وانگ و واسیلوا، ۲۰۰۵).

و قضیه بیز با استفاده از رابطه (۳.۲) ساده تر می شود.

$$P(x|y) = \frac{P(x,y)}{P(y)} \qquad (۳.۲)$$

و به فرم زیر بازنویسی می‌شود

$$P(H|E) = \frac{P(H,E)}{P(E)} \qquad (۴.۲)$$

یکی از مزایای بزرگ قضیه بیز این است که می‌توان احتمال وقوع رویداد A با معلوم بودن رویداد B را محاسبه کرد، زمانی که می‌توانید به راحتی احتمال رویداد B با معلوم بودن رویداد A را محاسبه

[۱] posterior probability
[۲] prior probability
[۳] Likelihood

کنید. به عبارت دیگر، می‌توان با استفاده از معادله (۲.۲) ، احتمال $P(A|B)$ را از $P(B|A)$ استنتاج کنیم.

برای درک مزیت این کاربرد، به مثال زیر توجه کنید. در هنگام تشخیص اینکه آیا یک شخص نیازمند ترمیم دندان خود است یا خیر، تعدادی علائم از جمله اینکه آیا شخص دچار درد دندان است یا خیر را در نظر می‌گیریم. با در نظر گرفتن یک جمعیت از بیماران نیازمند ترمیم دندان، ممکن است محاسبه احتمال (ترمیم | دندان‌درد)P به راحتی صورت بگیرد. با این حال، سوال مفید و جالب‌تر این است که احتمال (دندان‌درد | ترمیم)P چقدر است؟ این احتمال می‌تواند با استفاده از معادله (۲.۲) محاسبه شود.

اغلب اوقات، ممکن است مقادیر تعدادی از متغیرها را بدانیم. به عبارت دیگر، بیش از یک متغیر معلوم باشد. در چنین حالتی، می‌توان از قاعده زنجیره‌ای استفاده کنیم. قاعده زنجیره‌ای، کاربرد مهمی از قضیه بیز است و در معادله (۵.۲) بیان می‌شود. (نیدرمایر، ۲۰۰۸).

$$P(X_1, ..., X_n|c) = \prod_{i=1}^{n} P(X_i|X_1, ..., X_{i-1}, c) \qquad (۵.۲)$$

در یک مجموعه‌ی متغیرها، اگر اثر برخی متغیرها را کنترل کنیم امکان دارد برخی از متغیرها از یکدیگر مجزا یا مستقل باشند . اگر $P(A, B) = P(A)P(B)$ باشد، متغیرهای A و B مستقل شرطی هستند. به طور کلی، گفته می‌شود که متغیرهای A و B به شرط متغیر C مستقل هستند اگر

$$P(A, B|C) = P(A|C)P(B|C) \qquad (۶.۲)$$

از طرف دیگر A و B مستقل هستند اگر $P(A|B, C) = P(A|C)$. (۲۰۱۷، فنتون)

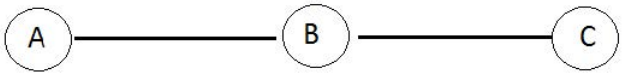
موضوع استقلال شرطی زمانی که درباره‌ی متغیرهایی صحبت می‌کنیم که بر متغیر فرضیه تأثیر می‌گذارند بسیار مهم است. استدلال درمورد روابط استقلال شرطی باعث ساده شدن بسیاری از محاسبات هنگام استفاده از شبکه بیزی می‌شود.

۲.۲ مدل‌های گرافیکی

یک گراف از مجموعه‌ای از گره‌ها و یال‌هایی که جفتی از آن‌ها را به هم وصل می‌کند ساخته می‌شود. هر یال نشان دهنده رابطه‌ای بین دو گره است. یال‌ها می‌توانند جهت‌دار یا بی‌جهت باشند، به طوری که یال جهت‌دار از گره والد N_p به گره فرزند N_c می‌رود. یال بی‌جهت می‌تواند به سادگی به عنوان یک حالت خاص از یال‌های جهت‌دار در نظر گرفته شود که در هر دو جهت بین دو گره حرکت می‌کند.

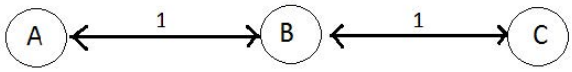
هر یال نیز اطلاعاتی درباره تبدیل از گره والد به گره فرزند رمزگذاری می‌کند و این اطلاعات در فرایند عبور از گراف معمولاً مفید است. در صورتی که این اطلاعات موجود نباشد، فرض می‌شود وزن تمام یال‌ها یکسان یا وزن واحدی داشته باشند. (استفنسون، ۲۰۰۰)

مثالی را از یک گراف ساده، بی‌جهت و بی‌وزن در شکل ۱.۲ در نظر بگیرید. در اینجا اتصالاتی بین سه گره A، B و C نشان داده شده است به طوری که گره‌های A و C از طریق گره B به طور غیرمستقیم به هم وصل هستند. آن‌ها به شرط گره B از یکدیگر مستقل هستند. شکل ۱.۲ بالا



شکل ۱.۲: گراف بدون وزن و بدون جهت

می‌تواند به صورت کلی به شکل ۲.۲ نشان داده شود که جهت‌ها و وزن‌ها را نشان می‌دهد. گراف بی‌جهت می‌تواند با یال‌هایی که در هر دو جهت حرکت می‌کنند و وزن واحدی دارند جایگزین شود. در یک گراف، گره‌های u و v به هم متصل هستند اگر مسیری از یکی به دیگری وجود داشته باشد.



شکل ۲.۲: گراف وزنی جهت‌دار

در مثال‌های بالا، گره‌های A و C به دلیل وجود یک مسیر از یکی به دیگری (حتی اگر به صورت

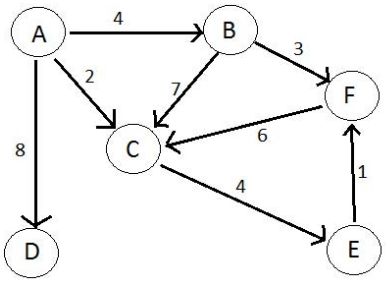
مستقیم نباشد) به هم متصل هستند. در شکل ۳.۲، گره‌های A و B به دلیل عدم وجود مسیری بین آن‌ها از هم جدا هستند. در زمینه شبکه‌های بیز، مسیرها به درک روابط بین گره‌ها کمک می‌کنند.

شکل ۳.۲: گره‌های قطع شده

گراف‌ها هنگامی جذاب می‌شوند که به گراف‌های جهت‌دار با وزن‌های متفاوت نگاه کنیم. شکل ۴.۲ نمونه‌ای از یک گراف با یال‌های جهت‌دار مختلف هر کدام با وزن مرتبط نشان می‌دهد. به عنوان مثال، حرکت از گرهA به گرهB هزینه ۴ واحد دارد.

اگر برای یک گره حداقل از یک یال راهی وجود داشته باشد که به خودش برگردد، گفته می‌شود آن گراف دور دارد یا مدور است. شکل ۴.۲ یک دور دارد زیرا گره‌های E،C وF یک دور را تشکیل می‌دهند.

گره‌ها در یک گراف هر کدام شامل درجات ورودی و خروجی هستند که به ترتیب تعداد گره‌های وارد شده به گره و تعداد گره‌های خارج شده از گره را نشان می‌دهند. به عنوان مثال، گره B در شکل ۴.۲ دارای درجه ورودی ۱ و درجه خروجی ۲ است. در زمینه شبکه‌های بیزی، درجه ورودی یک گره نشان می‌دهد که چند متغیر بر روی آن تأثیر می‌گذارد و درجه خروجی نشان می‌دهد که این گره بر روی چند متغیر تأثیر می‌گذارد.



شکل ۴.۲: گراف جهت‌دار با وزن‌های مختلف

۳

شبکه‌های بیزی

۱.۳ آشنایی

احتمال شرطی مفهومی بسیار مفید است. کاربردهای زیادی از دنیای واقعی را به راحتی میتوان با استفاده از احتمال شرطی بررسی کرد. به عبارت دیگر انسان‌ها به طور ذهنی درست همین طور شرایط را تحلیل می‌کنند. برای مثال یک پزشک معالج علایم بیمار را می‌شنود و محتمل‌ترین بیماری را بر اساس علائم تشخیص می‌دهد. یک سیاست‌گذار، محتمل‌ترین نتایج سیاستی که مد نظر اوست را تحلیل می‌کند.

یک مدل گرافیکی برای برقراری ارتباط بین بسیاری از این متغیرها (به عنوان مثال، دمای بدن، قرمزی گلو، سابقه بیماری و سابقه بیماری خانوادگی و غیره) با پیامدها و متغیرهای پاسخ احتمالی (به عنوان مثال، آلرژی، سل، دیابت و غیره) بسیار مفید است.

با این حال، در نگاه به نمودارهایی با عوامل زیاد، شاید بررسی تمامی نتایج امکان‌پذیر نباشد. یک سیستم نظارتی برای بیماران شامل ۳۷ مولفه مختلف است. در ساده‌ترین سیستم که در آن فرض شده است هر سیستم در وضعیت هشدار یا عدم هشدار است، می‌توان 2^{37} نتیجه ممکن را داشت. بررسی تمام این نتایج به روش‌های الگوریتم‌های جستجوی ساده بدون استفاده از روش‌های هوشمند، بسیار پیچیده است. با این حال، می‌توان با استفاده از الگوریتم‌های هوشمند جستجو،

راه‌حل‌های نزدیک به واقعیت را به صورت تقریبی بدست آورد و نتایج احتمالی را به کمک شبکه بیز تعیین کرد (این مثال از نیدرماير (۲۰۰۸) تغییر یافته است). به عبارتی دیگر، با استفاده از رویدادهای مشاهده شده و اثرات پیش‌بینی آن‌ها، به دست آورد. در چنین شرایطی، شبکه بیز بسیار مفید است.

۲.۳ تعریف

یک شبکه بیز، یک گراف جهت‌دار و غیرمدور است که در آن هر گره دارای اطلاعات احتمالی کمّی مرتبط با آن است. هر گره با یک متغیر تصادفی و یال‌ها بین آن گره و گره‌های دیگر مرتبط است. متغیرها ممکن است گسسته یا پیوسته باشند (اگرچه متغیرهای پیوسته اغلب گسسته‌سازی می‌شوند). یالی که از گره X به گره Y می‌رود به معنای این است که X پدر Y است و یک رابطه شرطی بین آنها را نشان می‌دهد. هر گره A_i دارای یک توزیع احتمال شرطی مرتبط است که توسط رابطه (۱.۳) نمایش داده می‌شود.

$$P(A_i|parents(A_i)) \qquad (۱.۳)$$

مجموعه‌ای از گره‌ها و یال‌ها ـ توپولوژی شبکه ـ رابطه‌های (عدم) وابستگی شرطی را نشان می‌دهد. هر رابطه از گره X به گره Y نشان دهنده تأثیر متغیر X بر متغیر Y است. به عبارت دیگر، مقداری که متغیر X روی توزیع احتمال متغیر Y تأثیر می‌گذارد را نشان می‌دهد. زمانی که یک گراف با چنین وابستگی‌هایی اسخراج می‌شود، توزیع‌های احتمال توام برای مجموعه‌ای از گره‌های وابسته برای استفاده از شبکه بیزی مشخص می‌شود. در این شرایط، قضیه بیز برای محاسبه احتمال‌های رویدادهای مختلف استفاده می‌شود.

اگر یک گره در گراف دارای درجه ورودی صفر باشد یا والدینی نداشته باشد (هیچ یالی به سمت آن نیاید)، تنها یک توزیع احتمالی برای خود دارد. اگر یک گره X دارای n والد باشد، برای هر والد Y_i یک توزیع احتمال شرطی وجود دارد. به این معنی که برای هر والد Y_i، یک جدول توزیع احتمالی با $P(X|Y_i)$ نمایش داده می‌شود.

شبکه بیز، ترکیبی از توپولوژی (گراف) و احتمالات شرطی متغیرها (گره ها) است. همه اینها با هم به بررسی تأثیر متغیرهای مختلف بر یکدیگر می‌پردازد.

تعیین احتمال‌ها در برخی موارد می‌تواند به سادگی با اختصاص دادن آن‌ها از طریق جداول توزیع احتمال توام است. با این حال، برای شبکه‌های بیزی جامع، این احتمالات براساس یادگیری و با جمع‌آوری داده‌های بیشتر امکان‌پذیر است. یادگیری باعث بهبود دانش از طریق ترکیب دانش قبلی با داده‌ها، می‌شود. (هکرمن، ۲۰۰۸)

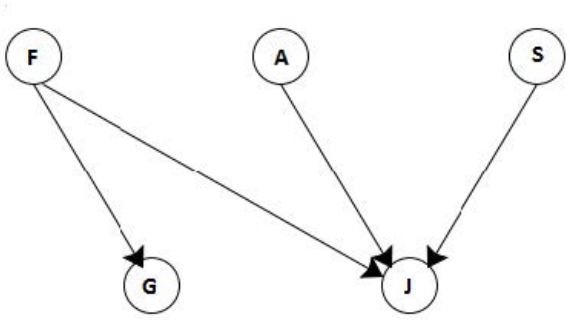
۳.۳ استنباط در یک شبکه بیز

استنباط، فرآیند محاسبه احتمال‌های رویدادهای نامشخص در یک شبکه بیزی با معلوم بودن داده‌های رویدادهای شناخته شده است. استنباط‌ها در صورت تعیین محتمل‌ترین مقدار متغیرها و سپس اخذ نتیجه‌گیری از این مقادیر، بنیادی است. (استفنسون، ۲۰۰۰)

هنگامی که برخی از پیشامدهای خاص معلوم باشند، می‌توان در مورد سایر رویدادها بیشتر از آنچه که هیچ پیشامدی شناخته شده نباشد، اطلاع کسب کنیم. بنابراین، می‌توان از این اطلاعات برای تجدیدنظر در دانش خود درباره چگونگی احتمال وقوع سایر رویدادها استفاده کنیم. این فرایند استنتاج به دنبال دستیابی دقیق به پالایش دانش براساس اطلاعات شناخته شده است.

شبکه بیز در شکل ۱.۳ را در نظر بگیرید. گره‌های شکل معرف متغیرهایی هستند که به بررسی این موضوعات می‌پردازند که آیا تراکنش تقلبی بوده است (F) ، آیا در ۲۴ ساعت گذشته خرید بنزین (G) انجام شده است، آیا در ۲۴ ساعت گذشته خرید جواهر (J) انجام شده است، جنسیت فرد (S) و سن فرد (A) . تعدادی وابستگی‌های شرطی در گراف دیده می‌شود. همچنین برخی استقلال‌های شرطی مشاهده می‌شود. به عنوان مثال، به شرط معلوم بودن J ، F و A مستقل هستند و به شرط معلوم بودن F ، G و J مستقل هستند .

سوال جالب در این شرایط این است که با توجه به سایر اطلاعات احتمال تقلب چقدر است؟ به عبارت دیگر، مشاهده چهار متغیر دیگر ساده‌تر از مشاهده این است که آیا یک تراکنش تقلبی است یا خیر . علاوه بر این، می‌توان از داده‌های گذشته در مورد تراکنش‌های تقلبی که برای این



شکل ۱.۳: نمونه شبکه بیزی اقتباس شده از هکرمن (۲۰۰۸)

متغیرها ثبت شده‌اند، استفاده کنیم تا برآوردی در مورد اینکه تراکنش تقلبی است یا نه بدهیم. از فرمول کلی که از رابطه (۳.۲) به دست آمده، برای به دست آوردن یک نگاه واقع‌گرایانه‌تر به احتمال رویداد f هنگامی که نتایج g ، s ، a و j را می‌دانیم، استفاده می‌کنیم. (استفنسون، ۲۰۰۰)

$$P(f|j,g,s,a) = \frac{P(j,g,s,a,f)}{P(j,g,s,a)} \qquad (۲.۳)$$

با در اختیار داشتن این دانش، می‌توان با استفاده از اطلاعات استقلال‌های شرطی موجود و قرار دادن آن‌ها در معادله (۶.۲) و سپس با استفاده از قانون زنجیره‌ای معادله (۵.۲) ، معادله‌ای که معادله (۲.۳) را ساده‌تر می‌کند، به دست آورد.

$$P(f|j,g,s,a) = \frac{P(j|s,a,f) * P(g|f) * P(f)}{\sum_{f_i} P(j|s,a,f_i) * P(g|f_i) * P(f_i)} \qquad (۳.۳)$$

این مقادیر جداگانه بسیار ساده‌تر از معادله (۲.۳) محاسبه می‌شود. بنابراین، ما می‌توانیم با استفاده از این روش، احتمال دقیق‌تری از تقلب به دست آوریم که مبتنی بر دانش مشاهده‌پذیر است.

۴.۳ سودمندی شبکه‌های بیزی

حال که شبکه‌های بیز را شناختیم، به بررسی مفید بودن آن‌ها می‌پردازیم. بخش بعد به بررسی مزایای شبکه‌های بیز نسبت به راه‌حل‌های دیگر در فضای حالت اختصاص دارد. اگرچه محاسبه اطلاعات اضافی هزینه‌بر است، اما سود حاصل از افزایش سرعت، ارزش انجام محاسبات اضافی را دارد.

۱.۴.۳ مناسب برای مجموعه داده‌های کوچک و ناقص

در شبکه‌های بیز، مفهوم "داده خیلی کم" [۱] وجود ندارد. در حالی که داده‌های بیشتر بهتر است، شبکه‌های بیز با داده‌های موجود کار می‌کنند و نتایج نسبتاً دقیقی ارائه می‌دهند. علاوه بر این، با هر بار تکرار، بیشتر یاد می‌گیرد و مدل خود را بهبود می‌بخشد تا دفعه بعد نتایج بهتری ارائه دهد. شبکه‌های بیز در واقع مدل‌های ریاضی هستند که با استفاده از مفاهیم گراف نمایش داده می‌شوند تا تحلیل، پیاده‌سازی و درک آن آسان‌تر شود. (یوزیتالو، ۲۰۰۷)

احتمال‌های شرطی با استفاده از تکنیک‌های مختلف برآورد می‌شوند و برای ارائه احتمال‌های نسبتاً دقیق به رویدادهای مختلف استفاده می‌شوند. تنها چیزی که لازم است، شناخت مدل است. از آنجا که مدل می‌تواند با حجم داده و وزن داده‌های قدیمی در مقابل داده‌های جدید، انعطاف پذیر باشد، می‌توان آن را به اندازه لازم انعطاف پذیر کرد.

به عنوان مثال، می‌توان یک مدل بسیار انعطاف‌پذیر داشت که وزن داده‌های جدید را بیشتر از داده‌های گذشته در نظر بگیرد. از طرف دیگر، ممکن است وزن داده‌های گذشته را بیشتر از داده‌های جدید در نظر بگیریم. هدف از این کار ایجاد یک برآورد اولیه و سپس اصلاح آن با داده‌های بیشتر است. با این حال، در هر دو مورد، می‌توان با مجموعه داده‌های کوچک و ناقص نتایج دقیقی تولید کرد. (یوزیتالو، ۲۰۰۷)

[۱] too little data

به گفته یوزیتالو (۲۰۰۷) شبکه‌های بیز در مدل‌سازی تغییرات محیطی مفید هستند. داده‌های محیطی اغلب ناقص و ناکافی هستند. به عنوان مثال، داده‌های ناقص ممکن است از رویدادهای خاص یا بازه‌های زمانی خاصی بی‌خبر باشند. در این صورت، شبکه‌های بیزی می‌توانند با استفاده از این داده‌های ناقص، با استدلال ریاضی و دقت منطقی، نتایج معناداری تولید کنند.

۲.۴.۳ امکان یادگیری ساختاری

استفاده جالب از شبکه‌های بیز هنگام بحث در مورد چگونگی یادگیری ساختاری مدل جدا از احتمالاتی که آنها رمزگذاری می‌کنند، است. در نسخه‌های ساده شبکه‌های بیز، به ایجاد و ساخت ساختار کمک می‌کنند و این ساختار ثابت می‌ماند. در حالی که این شبکه‌ها می‌توانند توزیع‌های احتمال شرطی را اصلاح کنند، اما وابستگی‌ها یا استقلال‌های جدید از داده‌ها را نمی‌توانند ایجاد کنند.

با این حال، شبکه‌های بیز را می‌توان به اندازه‌ای انعطاف‌پذیر ساخت که به آن‌ها اجازه دهد ساختار گراف را همانطور که از داده‌ها یاد می‌گیرند، تغییر دهند. به دلیل اینکه محاسبه نسخه بهینه برای برای پیاده‌سای در شبکه‌های بزرگ بسیار دشوار است، معمولاً الگوریتم‌ها به جای محاسبه نسخه بهینه، به دنبال تقریب چنین ساختارهایی هستند. (یوزیتالو، ۲۰۰۷)

یوزیتالو (۲۰۰۷) مدعی است که دو رویکرد اصلی برای یادگیری ساختاری در این نوع مدلسازی وجود دارد ـ رویکرد بیزی و رویکرد مبتنی بر رضایت‌مندی. رویکرد بیزی از کاربر/کارشناس می‌خواهد که ابتدا یک مدل را با دانش خود همراه با اعتماد کاربر به مدل وارد کند. سپس الگوریتم با استفاده از داده‌ها برای پیدا کردن بهترین مدل برازش استفاده می‌کند. از طرف دیگر، رویکرد مبتنی بر رضایت‌مندی از محدودیت نیازی به دانش تخصصی کاربر ندارد. وابستگی‌ها و استقلال‌های شرطی را بین جفت متغیرها جستجو می‌کند و با استفاده از دانشی که ایجاد می‌کند ساختار را می‌سازد.

۳.۴.۳ ترکیب منابع مختلف دانش

مزیت بزرگ شبکه‌های بیزی این است که امکان ترکیب دانش قبلی با داده‌های جدید را می‌دهد. یعنی می‌تواند دانش قبلی را با اطلاعات جدید به روز کند. یکی از مزایای این کار این است که امکان ترکیب داده‌های منابع مختلف با هم را می‌دهد. دانش قبلی حاصل از یک منبع را می‌توان با داده‌های منبع جدید ترکیب کرد تا استنتاج‌های جدیدی به‌دست آورد که می‌تواند در منبع قبلی وجود نداشته باشد. (یوزیتالو، ۲۰۰۷)

از آنجا که مدل‌ها در شبکه‌های بیزی داده‌ها را از منابع مختلف به طور مساوی وزن‌دهی می‌کنند، این مدل‌ها، داده‌ها را در حالی که از درجه‌های مختلفی از دقت برخوردار هستند، با هم ترکیب می‌کنند. علاوه بر این، مدل‌های شبکه‌های بیزی با ترکیب دانش کیفی با داده‌های کمی، محاسبات را آسان‌تر می‌کنند تا یک نتیجه دقیق ریاضی ایجاد کنند. (یوزیتالو، ۲۰۰۷)

در هوش مصنوعی، شبکه‌های بیزی به روش‌های مختلف و در ارتباط با تکنیک‌های دیگر استفاده می‌شوند. به عنوان مثال، می‌توان آن را با مسائل تصمیم‌گیری زنجیرهٔ مارکوف، روش‌های مونت کارلو، و سایر روش‌های گرداوری اطلاعات بدون نیاز به داده‌های گسترده و جامع استفاده کرد. (یوزیتالو، ۲۰۰۷)

۴.۴.۳ درمان صریح عدم قطعیت و پشتیبانی از تحلیل تصمیم

کدگذاری عدم قطعیت در شبکه بیزی بسیار آسان است. به عنوان مثال، فرض کنید مدلی برای پیش‌بینی تصمیمات صنعتی تهیه شده است. ممکن است تصمیم گیرنده در هنگام تصمیم‌گیری اهداف کوتاه یا بلندمدت در ذهن داشته باشد و این تصمیمات می‌تواند با «عقلانی‌ترین» تصمیم مطابق مدل ما مطابقت نداشته باشد. در این شرایط، می‌توان متغیری را برای درنظر گرفتن درجه‌ای از تصادفی بودن در تصمیم‌گیری اضافه کرد. (یوزیتالو، ۲۰۰۷)

راه دیگری برای بررسی این موضوع این است که وقتی از قطعیت نتایج اطلاع نداریم، می‌توانیم درجه‌ای از تصادفی بودن را که نمی‌توان آن را پیش‌بینی کرد، کدگذاری کنیم. برای مثال، شرایط پیش‌بینی نشده و متغیرهای غیرمنتظره را می‌توان با پذیرش اینکه ممکن است مقداری انحراف از

مدل داشته باشد که ما آن را در نظر نگرفته‌ایم، در نظر گرفت. این به ویژه برای مطالعه سیستم‌های کلان مانند محیط، آب و هوا، اقتصاد و غیره مفید است.

۵.۴.۳ پاسخ‌های سریع

به محض اجرای مدل، می‌توان با استفاده از جداول توزیع احتمال شرطی به سرعت نتایجی را دریافت کرد. با استفاده از مقادیر موجود در جداول و فرمول‌های مختلف مانند قانون زنجیره‌ای و روابط استقلال شرطی، می‌توان نتایج موردنظر را ارائه داد. بنابراین، نیازی به داشتن کلاسترهای بزرگی از کامپیوترها یا اجرای اسکریپت‌های بسیار تخصصی برای دریافت نتایج نیست. این روش در درک نتایج یک شبیه‌سازی و به اشتراک گذاشتن نتایج این با دیگران مناسب است. در واقع بسیار مشابه اجرا کردن یک الگوریتم جست‌وجوی عمق اول DFS [۲] (در جایی که می‌توان از مزیت‌های DFS بهره برد) در مقابل اجرا کردن جست‌وجوی سطحی (جست‌وجوی سطح اول) BFS [۳] است. (یوزیتالو، ۲۰۰۷)

[۲] Depth-first search

[۳] Breadth First Search

۴

آشنایی با برخی بسته‌های R در مورد گراف‌ها

همانطور که قبلاً اشاره شد یک گراف به صورت یک جفت $G = (V, E)$ تعریف می‌شود که در آن V مجموعه‌ای از رئوس یا گره‌ها و E مجموعه‌ای از یال‌ها است. هر یال با یک جفت گره که نقاط پایانی نام دارد، مرتبط است. یال‌ها به طور کلی جهت‌دار (سودار)، بدون جهت (بی‌سو) یا دو جهته هستند. گراف‌ها معمولاً با نمایش گره‌ها توسط دایره‌ها یا نقاط و یال‌ها توسط خطوط، فلش (پیکان) یا فلش‌های دوطرفه دیدارسازی می‌شوند. از نماد $\alpha - \beta, \alpha \longrightarrow \beta$ و $\alpha \longleftrightarrow \beta$ ، برای نشان دادن یال‌های بین α و β استفاده می‌شود. گراف‌هایی که در این پروژه به آن‌ها می‌پردازیم دارای مجموعه‌ای متناهی گره V است و در بیشتر موارد به این دلیل که نه چرخه و نه یال‌های متعدد دارند، گراف‌هایی ساده هستند. به دو رأس α و β مجاور گفته می‌شود و به صورت $\alpha \sim \beta$ نمایش داده می‌شود، اگر یک یال بین α و β در G وجود داشته باشد، یعنی اگر یکی از $\alpha - \beta, \alpha \longrightarrow \beta$ یا $\alpha \longleftrightarrow \beta$ برقرار باشد.

گراف‌ها کاربردهای زیادی دارند و تعدادی بسته در R برای کار با گراف‌ها وجود دارد که در ادامه به معرفی برخی از آن‌ها پرداخته می‌شود. بسته $graph$ یکی از بسته‌های بسیار مفید است که گراف‌ها را به عنوان شئ‌هایی که به اصطلاح $graphNEL$ نامیده می‌شوند (گراف‌ها را به عنوان لیستی از گره و یال ارائه می‌دهند) نمایش می‌دهد. در نتیجه این بسته امکان دسترسی به طیف وسیعی از عملگرهای نظری گراف (در بسته $graph$) ، امکان پیاده‌سازی کارآمد الگوریتم‌های

گراف استاندارد (در بسته RBGL) ، را فراهم می‌کند و اجازه می‌دهد تا گراف‌ها به راحتی در طرح‌بندی‌های مختلف نمایش داده شوند (با استفاده از بسته $Rgraphviz$ از $BioConductor$) . در کاربردهای آماری به دو نوع گراف خاص علاقمند هستیم: گراف‌های بدون جهت و گراف‌های غیر مدور جهت‌دار [1].

بسته $igraph$ بسته مفید دیگری است . مشابه بسته $graph$ ، بسته $igraph$ از گراف‌های بدون جهت و جهت‌دار پشتیبانی می‌کند و گراف‌های مختلفی را پیاده‌سازی می‌کند. توابع موجود در بسته‌ها امکان نمایش مختلفی از گراف‌ها را فراهم می‌کند. نمایش داخلی گراف‌ها در بسته $igraph$ متفاوت از نمایش در بسته $graph$ است.

بسته $gRbase$ با پیاده‌سازی برخی الگوریتم‌های مفید در مدل‌سازی گرافیکی، $graph$ و $igraph$ را تکمیل می‌کند. $gRbase$ همچنین دو تابع ، $ug()$ و $dag()$ را برای ایجاد آسان گراف‌های بدون جهت و DAG ، به عنوان اشیاء $graphNEL$ (پیش‌فرض) ، اشیاء $igraph$ یا ماتریس‌های مجاورت ارائه می‌دهد.

بخش‌های اول این فصل برخی از مفیدترین توابع موجود در هنگام کار با مدل‌های گرافیکی را شرح می‌دهد. این توابع به شکل‌های مختلف از بسته‌های graphgRbase و $RBGL$ می‌آیند. اما معمولاً نیازی به دانستن اینکه کدام یک باشد، نیست. برای استفاده توابع و رسم گراف‌ها کافی است $gRbase$ و $Rgraphviz$ را بارگذاری کنیم، زیرا $gRbase$ به طور خودکار سایر بسته‌ها را بارگیری می‌کند.

به عنوان اهداف آماری، از گراف‌ها برای نشان دادن مدل‌ها استفاده می‌شود ، به گونه‌ای که گره‌ها معرف متغیرهای مدل (و گاهی اوقات پارامترهای مدل) هستند و ساختار مستقل مدل را می‌توان مستقیماً از گراف‌ها آموخت. بر این اساس، بخشی از این فصل به شرح مختصری از مفهوم کلیدی استقلال شرطی اختصاص دارد و چگونگی پیوند آن با گراف‌ها را توضیح می‌دهد.

در این فصل نمودارها به عنوان اشیاء $graphNEL$ بیان می‌شوند. در صورت نیاز به سایر روش‌های نمایش گراف‌ها (به عنوان مثال ماتریس مجاورت)، توابع دیگری می‌تواند روی این اشیاء اعمال شود. تبدیل بین شئ $graphNEL$ ، شئ $graph$ و ماتریس‌های مجاورت با استفاده

[1] DAGs

از تابع $as()$ به صورت زیر امکان‌پذیر است. به عنوان مثال ، اگر gNg یک شئ $graphNEL$ باشد، بنابراین دستورهای زیر نسخه‌هایی از همان گراف را ایجاد می‌کند که به عنوان یک شئ $igraph$ و یا به عنوان یک ماتریس مجاورت نمایش داده شوند.

```
> ig <- as(gNg, "igraph")
> ag <- as(gNg, "matrix")
```

۱.۴ رسم نمودارهای بدون جهت در R

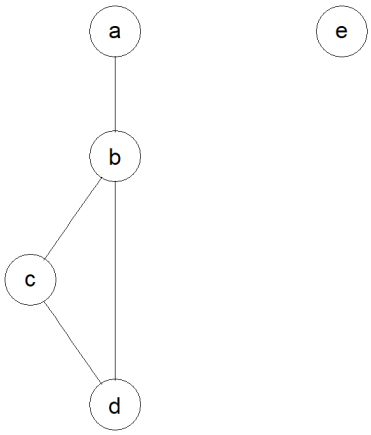
یک گراف بدون جهت می‌تواند با استفاده از تابع $ug()$ ایجاد شود. گراف با استفاده از فهرستی از فرمول‌ها ، یک فرمول واحد یا فهرستی از بردارها مشخص می‌شود. بنابراین فرم‌های زیر معادل هستند:

```
> library(gRbase)
> ug0 <- ug(~a:b, ~b:c:d, ~e)
> ug0 <- ug(~a:b+b:c:d+e)
> ug0 <- ug(~a*b+b*c*d+e)
> ug0 <- ug(c("a","b"),c("b","c","d"),"e")
> ug0

A graphNEL graph with undirected edges
Number of Nodes = 5
Number of Edges = 4
```

گراف‌های graphNEL با دستور $plot()$ به صورت زیر نمایش داده می‌شوند:

```
> library(Rgraphviz)
> plot(ug0)
```



به طور پیش فرض تابع $ug()$ یک شئ $graphNEL$ را برمی‌گرداند، اما گزینه‌های $result =$ "$igraph$" یا "$matrix$" $result =$ آن را به بازگرداندن $igraph$ یا ماتریس مجاورت هدایت می‌کنند. به عنوان مثال :

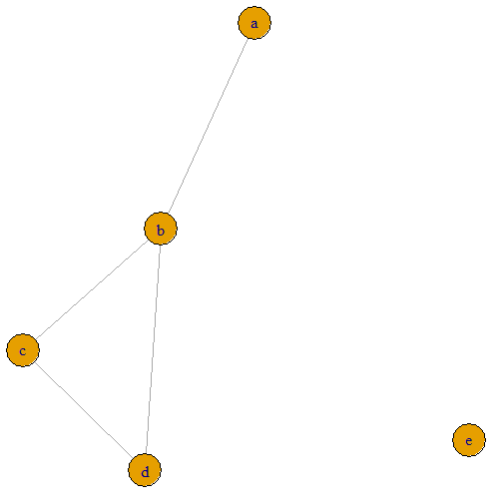
```
> ug0i <- ug(~a:b+b:c:d+e, result="igraph")
> ug0i

Vertices: 5
Edges: 4
Directed: FALSE
Edges:
[0] 'a' -- 'b'
```

```
[1] 'b' -- 'c'
[2] 'b' -- 'd'
[3] 'c' -- 'd'
```

یک روش $plot()$ برای شئ $igraph$ در بسته $igraph$ وجود دارد. همچنین امکانات مختلفی برای کنترل چیدمان گراف‌ها وجود دارد. به عنوان مثال، می‌توان از یک الگوریتم چیدمانی به نام $layout.spring$ به صورت زیر استفاده کنیم:

```
> plot(ug0i, layout=layout.spring)
```



یال‌ها را می‌توان با استفاده از توابع $addEdge()$ و $removeEdge()$ ، به ترتیب اضافه و حذف کرد:

```
> ug0a <- addEdge("a","c", ug0)
> ug0a <- removeEdge("c","d", ug0)
```

گره‌ها و یال‌های گراف را می‌توان با توابع $nodes()$ و $edges()$ بازیابی کرد.

```
> nodes(ug0)
> edges(ug0)
```

بنابراین دستور $edges()$ برای هر گره، گره‌های مجاور را می‌دهد. تابع $edgeList()$ فهرستی از جفت‌های (نامرتب) را برمی‌گرداند (برای فشرده‌سازی خروجی، از دستور $str()$ استفاده شود).

```
> str(edgeList(ug0))

List of 4
$ : chr [1:2] "b" "a"
$ : chr [1:2] "c" "b"
$ : chr [1:2] "d" "b"
$ : chr [1:2] "d" "c"
```

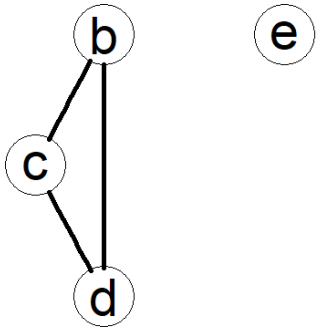
تابع $maxClique()$ دسته‌های (maximal) را از یک گراف برمی‌گرداند:

```
> maxClique(ug0)

$maxCliques
$maxCliques[[1]]
[1] "b" "c" "d"
$maxCliques[[2]]
 [1] "b" "a"
$maxCliques[[3]]
[1] "e"
```

با استفاده از تابع $subGraph()$ می‌توان زیرگراف‌های یک گراف را استخراج کرد. برای مثال:

```
> ug1 <- subGraph(c("b","c","d","e"), ug0)
> plot(ug1)
```



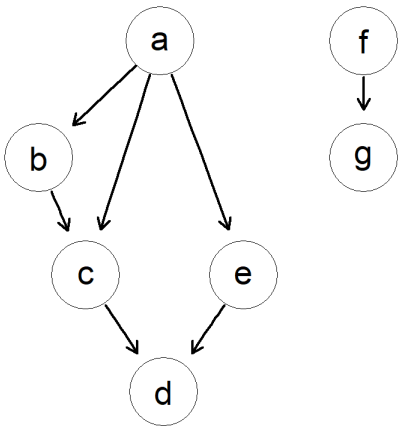
۲.۴ رسم نمودارهای غیر مدور جهت‌دار در R

یک گراف غیرمدور جهت‌دار می‌تواند با استفاده از تابع $dag()$ ایجاد شود. گراف با استفاده از فهرستی از فرمول‌ها، یک فرمول واحد یا فهرستی از بردارها مشخص می‌شود. بنابراین فرم‌های زیر معادل هستند:

```
> dag0 <- dag(~a, ~b*a, ~c*a*b, ~d*c*e, ~e*a, ~g*f)
> dag0 <- dag(~a + b*a + c*a*b + d*c*e + e*a + g*f)
> dag0 <- dag(~a + b|a + c|a*b + d|c*e + e|a + g|f)
> dag0 <- dag("a", c("b","a"), c("c","a","b"), c("d","c","e"),
+ c("e","a"),c("g","f"))
> dag0
```

تابع $dag()$ به صورت پیش‌فرض یک شئ $graphNEL$ را برمی‌گرداند، اما گزینه‌های $result =$ "$igraph$" یا $result =$ "$matrix$" آن را به بازگشت $igraph$ یا ماتریس مجاورت هدایت می‌کنند. DAG با $plot()$ نمایش داده می‌شود:

```
> plot(dag0)
```



گره‌ها و یال‌های یک DAG را می‌توان با دستورهای $nodes()$ و $edges()$ بازیابی کرد.

```
> nodes(dag0)
> str(edges(dag0))
```

بنابراین $edges()$ فرزندان هر گره را می‌دهد. متناوباً فهرستی از جفت‌ها (مرتب شده) را می‌توان با $edgeList()$ دریافت کرد.

```
> str(edgeList(dag0))

List of 7
$ : chr [1:2] "a" "b"
```

```
$ : chr [1:2] "a" "c"
$ : chr [1:2] "b" "c"
$ : chr [1:2] "c" "d"
$ : chr [1:2] "e" "d"
$ : chr [1:2] "a" "e"
$ : chr [1:2] "f" "g"
```

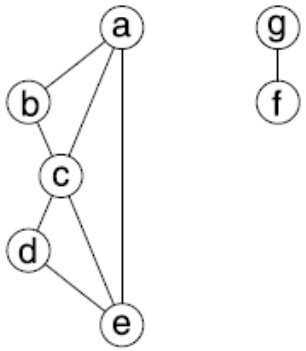
تابع $vpar()$ یک فهرستی را با یک عنصر برای هر گره همراه با والدین آن برمی‌گرداند:

```
> vpardag0 <- vpar(dag0)
> vpardag0$c

[1] "c" "a" "b"
```

تابع $moralize()$ برای یافتن گراف بدون جهت از یک گراف غیر مدور جهت‌دار استفاده می‌شود:

```
> dag0m <- moralize(dag0)
> plot(dag0m)
```



۳.۴ گراف‌های آمیخته

گراف‌های آمیخته، گراف‌هایی با حداقل دو نوع یال، به عنوان مثال جهت‌دار و بدون‌جهت، یا جهت‌دار و دو جهته هستند. گراف‌های آمیخته در بسته‌های $graph$ و $igraph$ به عنوان گراف‌های جهت‌دار با چندین نوع یال نمایش داده می‌شوند. یک راه راحت تعریف آنها (به جای فرمول‌های مدل) استفاده از ماتریس‌های مجاورت است. نحوهٔ ساخت چنین ماتریسی به شرح زیر است.

```
> adjm <- matrix(c(0,1,1,0,1,0,0,1,1,0,0,0,1,1,1,0), nrow=4)
> rownames(adjm) <- colnames(adjm) <- letters[1:4]
> adjm
```

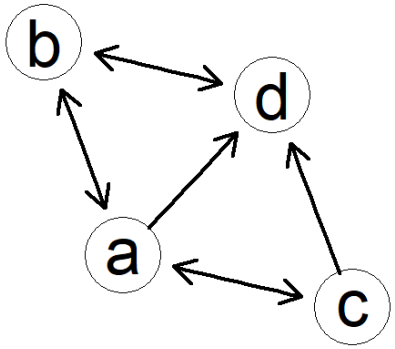
```
  a b c d
a 0 1 1 1
b 1 0 0 1
c 1 0 0 1
d 0 1 0 0
```

می‌توان از این ماتریس برای ایجاد یک شی $graphNEL$ استفاده کرد:

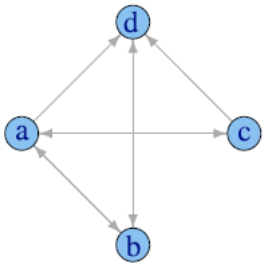
```
> gG <- as(adjm, "graphNEL")
> plot(gG, "neato")
```

توجه داشته باشید که $Rgraphviz$ ورودی‌های متقارن را به عنوان پیکان‌های دوتاج نمایش می‌دهد، بنابراین بین یال‌های دوجهته [۲] و بدون‌جهت [۳] تمایز قائل نمی‌شود. همین مساله در صورت نمایش گراف به عنوان یک شیء $igraph$ نیز صادق است:

[۲] bidirected
[۳] undirected

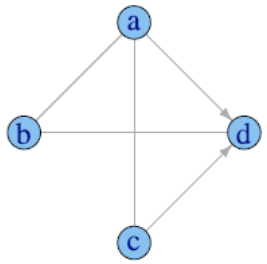



```
> gG1 <- as(adjm, "igraph")
> plot(gG1, layout=layout.spring)
```



با این حال می‌توانیم $igraph$ را وادار کرد که به‌جای یال‌های دوطرفه، یال‌های بدون جهت را نمایش دهد:

```
> E(gG1)$arrow.mode <- c(2,0)[1+is.mutual(gG1)]
> plot(gG1, layout=layout.spring)
```

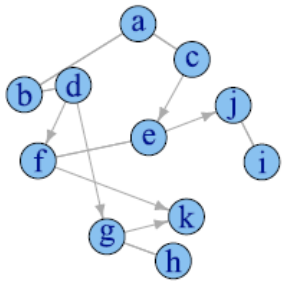


گراف زنجیری، گرافی آمیخته است که هیچ یال دوجهته و هیچ دور نیم‌جهته [۴] ندارد. این گراف‌ها تعمیمی از گراف‌های بدون جهت و گراف‌های جهت‌دار را تشکیل می‌دهند، که در ادامه خواهیم دید. مثال زیر برگرفته از فریدنبرگ (۱۹۹۰) است.

```
> d1 <- matrix(0,11,11)
> d1[1,2] <- d1[2,1] <- d1[1,3] <- d1[3,1] <- d1[2,4] <- d1[4,2] <-
+ d1[5,6] <- d1[6,5] <- 1
> d1[9,10] <- d1[10,9] <- d1[7,8] <- d1[8,7] <- d1[3,5] <-
+ d1[5,10] <- d1[4,6] <- d1[4,7] <- 1
> d1[6,11] <- d1[7,11] <- 1
> rownames(d1) <- colnames(d1) <- letters[1:11]
> cG1 <- as(d1, "igraph")
> E(cG1)$arrow.mode <- c(2,0)[1+is.mutual(cG1)]
> plot(cG1, layout=layout.spring)
```

تابع $is.chaingraph()$ در بسته lcd، مشخص می‌کند که گراف آمیخته، گراف زنجیری است یا خیر. این تابع، یک ماتریس مجاورت را به عنوان ورودی می‌گیرد. به عنوان مثال، گراف زیر، گرافی زنجیری است.

[۴] semi-directed cycle

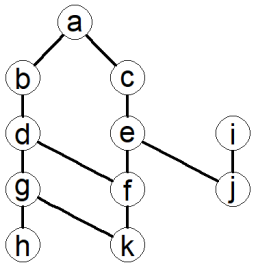



```
> library(lcd)
> is.chaingraph(as(cG1, "matrix"))

$result
[1] TRUE
$vert.order
[1] 1 2 3 4 5 6 7 8 9 10 11
$chain.size
[1] 4 2 2 2 1
```

عملیات moralization برای نمودارهای زنجیری مهم است. مشابه DAG ها، والدین مجرد مولفه‌های زنجیره مشابه با یکدیگر ادغام می‌شوند و سپس جهت‌ها حذف می‌شوند. این عملیات در تابع $moralize()$ در بسته lcd پیاده‌سازی شده است که از نمایش ماتریس مجاورت استفاده می‌کند. به عنوان مثال:

```
> cGm <- as(moralize(as(cG1, "matrix")), "graphNEL")
> plot(cGm)
```



۴.۴ چیدمان گراف در *Rgraphviz*

با وجود اینکه نحوه نمایش گراف‌ها در صفحه یا صفحه‌نمایش بر ویژگی‌های ریاضی یا آماری آن‌ها تأثیری ندارد، اما در عمل، نمایش آن‌ها به نحوی که ساختار آن‌ها به وضوح آشکار شود، مفید است. بسته Rgraphviz چندین روش برای تنظیم خودکار چیدمان گراف پیاده‌سازی می‌کند. این روش‌ها به طور خلاصه در ادامه بیان می‌شود.

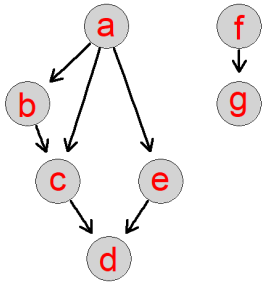
- روش dot که پیش فرض است، برای ترسیم DAG یا سلسله مراتب‌ها مانند ارگانوگرام‌ها [۵] یا فیلوژنی‌ها [۶] در نظر گرفته شده است.
- روش twopi برای گراف‌های متصل مناسب است: این روش یک چیدمان مدور با یک گره قرار گرفته در مرکز و دیگر گره‌ها در چندین حلقه مرکزی دور مرکز ایجاد می‌کند.
- روش circo همچنین یک چیدمان مدور ایجاد می‌کند.
- روش neato برای گراف‌های بدون جهت مناسب است: یک الگوریتم تکراری مختصات گره‌ها را به گونه‌ای تعیین می‌کند که فاصله هندسی بین زوج گره‌ها مسیر آن‌ها در گراف را تقریباً تعیین می‌کند.
- به طور مشابه، روش fdp بر اساس یک الگوریتم تکراری به مبتنی بر فروچترمن و رینگولد(۱۹۹۱) است که در آن گره‌های مجاور به هم جذب و گره‌های غیر مجاور دفع می‌شوند.

گراف‌های نمایش داده شده با استفاده از *Rgraphviz* را نیز می‌توان به روش های مختلفی تزئین کرد: به عنوان مثال، در مثال زیر، متن گراف با رنگ قرمز نمایش داده شده و گره‌ها با رنگ

[۵] organograms
[۶] phylogenies

خاکستری روشن پر شده‌اند.

```
> plot(dag0, attrs=list(node = list(fillcolor="lightgrey",
fontcolor="red")))
```

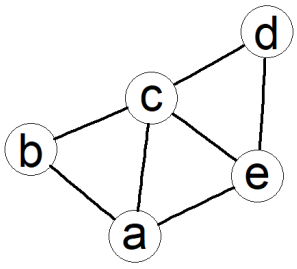


چیدمان گراف را می‌توان مجدداً استفاده کرد و این امر بسیار مفید است. برای این کار از تابع $agopen()$ برای تولید یک شی $Ragraph$ استفاده می‌شود که نمایشی از چیدمانیک گراف است (نه خود گراف به عنوان یک شیء ریاضی). از این قسمت یال موردنظر را حذف می‌کنیم.

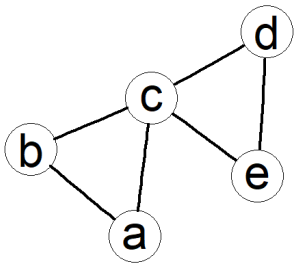
```
> edgeNames(ug3)
```

```
[1] "a~b" "a~c" "a~e" "b~c" "c~d" "c~e" "d~e"
```

```
> ng3 <- agopen(ug3, name="ug3", layoutType="neato")
> ng4 <- ng3
> AgEdge(ng4) <- AgEdge(ng4)[-3]
> plot(ng3)
```



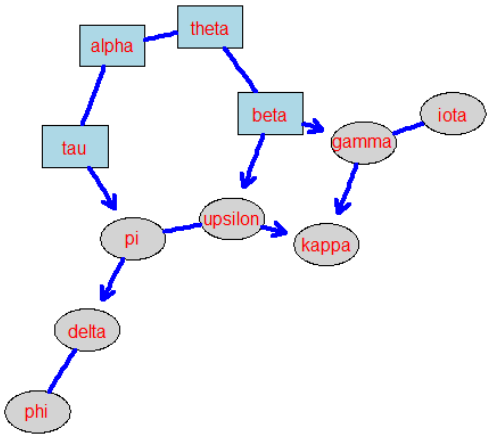

```
> plot(ng4)
```



مثال زیر نشان می‌دهد چگونه ویژگی‌های مختص به هر یال و گره می‌تواند تعیین شود. در اینجا از گراف زنجیره‌ای cG۱ که قبلاً توصیف شد استفاده می‌کنیم.

```
> cG1a <- as(cG1, "graphNEL")
> nodes(cG1a) <- c("alpha","theta","tau","beta","pi","upsilon","gamma",
+ "iota","phi","delta","kappa")
> edges <- buildEdgeList(cG1a)
> for (i in 1:length(edges)) {
+ if (edges[[i]]@attrs$dir=="both") edges[[i]]@attrs$dir <- "none"
+ edges[[i]]@attrs$color <- "blue"
+ }
> nodes <- buildNodeList(cG1a)
```

```
> for (i in 1:length(nodes)) {
+ nodes[[i]]@attrs$fontcolor <- "red"
+ nodes[[i]]@attrs$shape <- "ellipse"
+ nodes[[i]]@attrs$fillcolor <- "lightgrey"
+ if (i <= 4) {
+ nodes[[i]]@attrs$fillcolor <- "lightblue"
+ nodes[[i]]@attrs$shape <- "box"
+ }
+ }
> cG1al <- agopen(cG1a, edges=edges, nodes=nodes, name="cG1a",
layoutType="neato")
> plot(cG1al)
```



۵.۴ بسته $igraph$

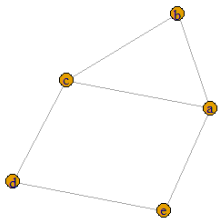
بسته $igraph$ مکمل یا جایگزینی برای بسته $graph$ با ویژگی‌های بسیار جذاب است، همانطور که دیده‌ایم، تبدیل بین اشیاء $graphNEL$ ، اشیاء $igraph$ و ماتریس‌های مجاورت با استفاده از تابع $as()$ بسیار آسان است. به علاوه، می‌توان اشیاء $igraph$ را با استفاده از تابع $graph.formula()$ ایجاد کرد:

```
> ug4 <- graph.formula(a -- b:c, c--b:d, e -- a:d)
> ug4

Vertices: 5
Edges: 6
Directed: FALSE
Edges:
[0] 'a' -- 'b'
[1] 'a' -- 'c'
[2] 'b' -- 'c'
[3] 'c' -- 'd'
[4] 'a' -- 'e'
[5] 'd' -- 'e'

> plot(ug4, layout=layout.graphopt)
```

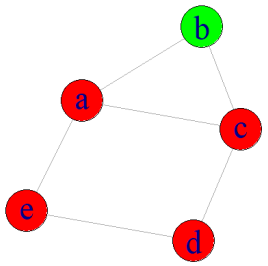
همین گراف ممکن است از ابتدا به صورت زیر ایجاد شود:




```
> ug4.2 <- graph.empty(n=5, directed=FALSE)
> V(ug4.2)$name <- V(ug4.2)$label <- letters[1:5]
> ug4.2 <- add.edges(ug4.2, c(0,1, 0,2, 0,4, 1,2, 2,3, 3,4))
> ug4.2
```

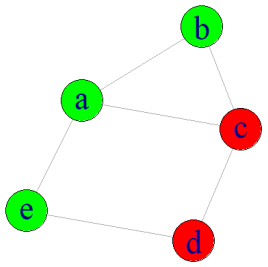
همانند شیء $graphNEL$ ، در $igraph$ گراف‌ها به وسیله‌ی فهرستی از گره‌ها و یال‌ها تعریف می‌شوند. به‌علاوه، آن‌ها ویژگی‌هایی دارند: این ویژگی‌ها متعلق به گره‌ها، یال‌ها یا خود گراف هستند. در مثال زیر یک ویژگی گراف به نام $layout$ و دو ویژگی گره به نام‌های $label$ و $color$ تنظیم می‌شوند. این‌ها هنگامی که گراف نمایش داده می‌شود مورد استفاده قرار می‌گیرند. ویژگی name حاوی برچسب‌های گره است.

```
> ug4$layout <- layout.graphopt(ug4)
> V(ug4)$label <- V(ug4)$name
> V(ug4)$color <- "red"
> V(ug4)[1]$color <- "green"
> V(ug4)$size <- 40
> V(ug4)$label.cex <- 3
> plot(ug4)
```



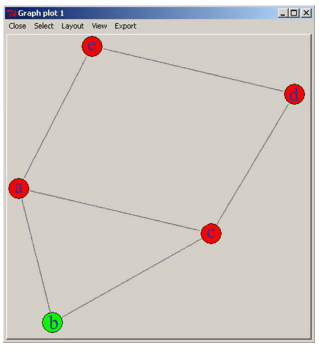
تعریف ویژگی‌های جدید برای اشیاء $igraph$ بسیار آسان است. در مثال زیر، یک ویژگی جدید برای گره‌ها به نام $discrete$ تعریف می‌شود و از این ویژگی برای رنگ‌آمیزی گره‌ها استفاده می‌شود.

```
> ug5 <- set.vertex.attribute(ug4, "discrete", value=c(T,T,F,F,T))
> V(ug5)[discrete]$color <- "green"
> V(ug5)[!discrete]$color <- "red"
> plot(ug5)
```



تابع $tkplot()$ یک امکان نمایش تعاملی گراف را فراهم می‌کند. این تابع باعث ظهور یک پنجره پاپ‌آپ می‌شود که در آن می‌توان گراف را به‌صورت دستی ویرایش کرد. یکی از کاربردهای این امکان ویرایش دستی، ویرایش چیدمان گراف است: مختصات جدید می‌توانند استخراج شده و با استفاده از تابع $plot()$ مجدداً استفاده شوند. به عنوان مثال:

```
> tkplot(ug4)
```



تابع $tkplot()$ یک شناسه پنجره را برمی‌گرداند (در اینجا شماره ۲). در حالی که پنجره پاپ‌آپ باز است، می‌توان با ارسال شناسه پنجره به تابع $tkplot.getcoords()$ ، چیدمان فعلی را به‌دست آورد، به عنوان مثال:

```
> xy <- tkplot.getcoords(2)
> plot(g, layout=xy)
```

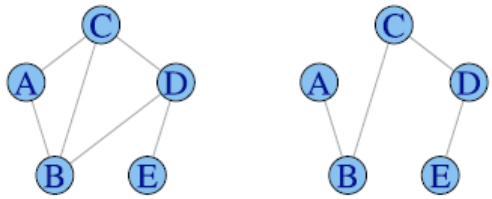
استفاده مجدد از اطلاعاتچیدمان با اشیاء $igraph$ ساده است. هنگامی که طرح توابع بر روی گراف‌ها اعمال می‌شوند، ماتریسی از مختصات (x, y) را بر می‌گردانند:

```
> layout.spring(ug4)

      x             y
1 7.568e-01 0.2459
2 -7.568e-01 0.2459
3 4.677e-01 -0.6438
4 -4.677e-01 -0.6438
5 4.337e-19 0.7958
```

کد زیر نشان‌دهنده‌ی این است که چگونه دو گراف با همان مجموعه‌ی گره‌ها [۷] می‌توانند با استفاده از همان چیدمان [۸] نمایش داده شوند.

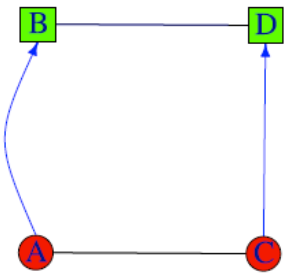
```
> ug5 <- ug(~A*B*C + B*C*D + D*E, result='igraph')
> ug6 <- ug(~A*B+B*C+C*D+D*E, result='igraph')
> ug6$layout <- ug5$layout <- layout.spring(ug5)
> V(ug5)$size <- V(ug6)$size <- 50
> V(ug5)$label.cex <- V(ug6)$label.cex <- 3
> par(mfrow=c(1,2))
> plot(ug5); plot(ug6)
```



[۷] vertex set
[۸] layout

میتوان نمای کلی از ویژگی‌های مورد استفاده در نمودار را با تایپ کردن *?igraph.plotting* بدست آورد. مثال نهایی برای نشان دادن گراف‌های پیچیده‌تر به شرح زیر است.

```
> em1 <- c(0,0,1,0,1,0,0,1,1,0,0,0,0,1,1,0)
> dim(em1) <- c(4,4)
> iG <- graph.adjacency(em1)
> V(iG)$shape <- c("circle","square","circle","square")
> V(iG)$color <- c("red","green")
> V(iG)$label <- c("A", "B", "C", "D")
> E(iG)$arrow.mode <- c(2,0)[1+is.mutual(iG)]
> E(iG)$color <- c("blue", "black")
> E(iG)$curved <- c(T,F,F,F,F,F)
> iG$layout <- layout.graphopt(iG)
> plot(iG)
```



واژه‌نامه فارسی به انگلیسی

آمار بیزی ..... Bayesian statistics
احتمال پسین ..... Posterior probability
احتمال پیشین ..... Prior probability
انعطاف‌پذیر ..... Flexible
مجزّا ..... Disjoint
استقلال شرطی ..... Conditional independence
استنباط ..... Inference
انحراف ..... Deviation
بدون جهت ..... Undirected
بدون وزن ..... Unweighted
بهینه ..... Optimal
تقلب ..... Fraud
توپولوژی ..... Topology
توپولوژی کدبندی شبکه ..... Topology of the network encode
ثابت ..... Constant
ـ جداساز d ..... d-Separation
جدول توزیع احتمال ..... Probability distribution table
جست‌وجوی سطح اول ..... Breadth first search

جست‌وجوی عمق اول Depth first search

جهت‌دار Directed

خوشه Clusters

درست‌نمایی Likelihood

دسته Clique

دو جهته Bidirected

رأس Vertex

زنجیر Chain

زنجیر مارکوف Markov chain

زیرگراف Subgraph

زیرمجموعه Subset

سیستم‌های خبره Expert systems

شبکه‌ی بیزی Bayesian network

شبیه‌سازی Simulation

شرطی Conditional

غیرمدور Acyclic

فرزند،یچه Child

فرضیه Hypothesis

فضای حالت State space

فضای مسئله Problem space

قاعده زنجیره‌ای Chain rule

کمان Arc

گراف Graph

گراف غیرمدور جهت‌دار Directed Acyclic Graphs

گره، رأس Node

ماتریس مجاورت ..... Adjacency matrix

متغیر مستقل ..... Independent variable

متغیر وابسته ..... Dependent variable

مسیر ..... Path

مسیر بدون جهت ..... Undirected path

نظریه احتمال ..... Probability theory

نظریه‌ی گراف ..... Graph theory

نمایی ..... Exponentially

هوش مصنوعی ..... Artificial Intelligence

والدین ..... Parents

یال ..... Edge

کتاب‌نامه

[1] Albert J (2009) Bayesian computation with R, 2nd edn. Springer, New York

[2] Bøttcher SG, Dethlefsen C (2003) deal: A package for learning Bayesian networks. J Stat Softw **8(20)**:1–40. http://www.jstatsoft.org/v08/i20

[3] Cheng, J. and Greiner, R. (1999). Comparing bayesian network classifiers. In Proceeding *UAI'99 Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence.*

[4] Dawid AP (1998) Conditional independence. In: Kotz S, Read CB, Banks DL (eds) Encyclopedia of statistical sciences, update, vol 2. Wiley-Interscience, New York, pp 146–155

[5] Edwards D (2000) Introduction to graphical modelling, 2nd edn. Springer, New York

[6] Fenton, N. Independence and conditional independence. https://www.eecs.qmul.ac.uk/~norman/BBNs/Independence_and_conditional_independence.htm. Accessed: 2017-03-06.

[7] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). *Bayesian network classifiers. In Machine Learning Volume 29.*

[8] Fruchterman T, Reingold EM(1991) Graph drawing by force-directed placement. Softw Pract Exp 21: 1129–1164

[9] Frydenberg M (1990) The chain graph Markov property. Scand J Stat 17: 333–353

[10] Gilks WR, Thomas A, Spiegelhalter DJ (1994) BUGS: a language and program for complex Bayesian modelling. Statistician **43**: 169–178

[11] Green PJ (2005) GRAPPA: R functions for probability propagation. http://www.stats.bris.ac.uk/~peter/Grappa/

[12] Heckerman, D. (2008). *A Tutorial on Learning with Bayesian Networks, pages 33–82. Springer Berlin Heidelberg, Berlin, Heidelberg.*

[13] Koch, K.-R. (2006). *Bayesian Inference with Geodetic Applications, volume 31. Springer.*

[14] Niedermayer, D. (2008). An introduction to bayesian network theory and their contemporary applications. In *Innovations in Bayesian Networks Volume 156.*

[15] Russell, S. J., Norvig, P., and Davis, E. (2009). *Artificial intelligence: a modern approach.* Pearson, 3 edition.

[16] S. Højsgaard et al., Graphical Models with R, Use R!, DOI 10.1007/978-1-4614-2299-0, © Springer Science+Business Media, LLC 2012

[17] Stephenson, T. A. (2000). *An introduction to bayesian network theory and usage*. In IDIAP Research Report.

[18] Uusitalo, L. (2007). Advantages and challenges of bayesian networks in environmental modelling, *Ecological Modelling*,**203 (3-4)**, 312-318.

[19] Wang, Y. and Vassileva, J. (2005). Bayesian network trust model in peer-to-peer networds. In Agents and Peer-to-Peer Computing.

[20] Whittaker J (1990) Graphical models in applied multivariate statistics. Wiley, Chichester

Abstract

Bayesian networks are a means to study data. A Bayesian network gives structure to data by creating a graphical system to model the data. It then develops probability distributions over these variables. It explores variables in the problem space and examines the probability distributions related to those variables. It conducts statistical inference over those probability distributions to draw meaning from them. They are good means to explore a large set of data efficiently to make inferences. There are a number of real world applications that already exist and are being actively researched. This project discusses the theory and applications of Bayesian networks and explores some graphical modeling using R software.

College of Science

School of Mathematics, Statistics, and Computer Science

An Introduction to Graphical Models with R

Mohammadreza Yadollahi

Supervisor: Zahra Rezaei Ghahroodi

A thesis submitted in partial fulfillment of the requirements for the degree of B.Sc. in Statistics

1402