الگوریتم CLIP4

حسن شجاعی مند
دانشجوی کارشناسی ارشد نرم افزار
دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف
shojaee@ce.sharif.edu

خلاصه: در این مقاله، می‌خواهیم الگوریتم CLIP4 را مورد بررسی قرار دهیم. این الگوریتم از نوع الگوریتم‌های یادگیری ماشین استقرایی می‌باشد، که برای انجام عمل دسته‌بندی[1] مورد استفاده قرار می‌گیرد. الگوریتم ابتدا داده‌ها را به زیرمجموعه‌هایی با استفاده از یک ساختار درختی تقسیم‌بندی می‌کند و سپس قواعد تولید[2] را تنها با استفاده از زیرمجموعه‌هایی که در گره‌های برگ قرار دارند، ایجاد می‌کند. ویژگی منحصر به‌فرد الگوریتم، امکان تولید قواعدی می‌باشد که از نامساوی استفاده می‌کنند. کارایی الگوریتم با استفاده از یکسری منبع داده محک‌زنی[3]، با بقیه الگوریتم‌های یادگیری ماشین مقایسه خواهد شد.

۱- مقدمه

در این مقاله می‌خواهیم الگوریتم CLIP4 که در [1] مطرح شده است را مورد بررسی قرار دهیم. این الگوریتم یکی از الگوریتم‌های یادگیری‌ماشین به حساب می‌آید که از روش هدایت‌شده[4]، استفاده می‌کند. این تنها الگوریتمی‌است که توانایی تولید قواعد نامساوی را دارا می‌باشد.

اولین الگوریتم CLIP[5] در سال ۱۹۹۵ با عنوان CLIP2 در [2] ارائه شد. این الگوریتم بعدا در CLIP3 بهبود یافت[3]. الگوریتم CLIP4، تغییراتی در دو الگوریتم قبلی بوجود آورده است که باعث دقت،کارایی و انعطاف‌پذیری بیشتری می‌شود. این الگوریتم یک الگوریتم ترکیبی است که ایده‌های موجود در خانواده الگوریتم‌های یادگیری ماشین را با هم ترکیب کرده است: الگوریتم‌های مبتنی بر قاعده و الگوریتم‌های درخت تصمیم. تفاوت اصلی بین این الگوریتم و الگوریتم‌های موجود در این دو خانواده، استفاده زیاد آن از مسئله پوشش مجموعه[6] می‌باشد. الگوریتم دیگری که به صورت ترکیبی از الگوریتم‌های مبتنی بر قاعده و الگوریتم‌های درخت تصمیم استفاده می‌کند، الگوریتم CN2 [5] می‌باشد که از یک معیار انتروپی شبیه به آنچه در الگوریتم‌های درخت تصمیم استفاده می‌شود، استفاده می‌کند.

چند مورد در [1] به عنوان فاکتورهای مهمی‌که می‌توانند در یک سیستم یادگیری‌ماشین هدایت شده مطرح باشند، عنوان شده است:

---

[1] classification
[2] Production rule
[3] benchmarking
[4] supervised
[5] cover learning using integer linear programming
[6] set covering

- دسته‌بندی دقیق[7]: قواعد تولید شده، باید توانایی دسته‌بندی درست رکوردهایی که در آینده مطرح خواهند شد را داشته باشد و باید بتواند در حضور نویز در داده‌ها کار خود را بدرستی انجام دهد.
- قواعد ساده: قواعد باید تا حد ممکن مختصر باشند. چرا که در اینصورت قابل‌فهم‌تر خواهند بود.
- تولید کارای قواعد: الگوریتم باید توانایی کار با یک منبع داده بزرگ را نیز داشته باشد.
- انعطاف‌پذیری: الگوریتم باید حوزه وسیعی از کاربردها را شامل شود. بتواند انواع داده‌های مختلف، تعداد فیلدهای زیاد و فیلدهای با داده‌های خطادار را تحت پوشش قرار دهد.

در ادامه ، ابتدا الگوریتم CLIP4 را با جزئیات و بر اساس شبه‌کد ارائه شده برای آن بررسی خواهیم کرد و سپس مثالی از کاربرد آنرا خواهیم دید و در ضمن مثال جزئیات کارکرد قسمتهای مختلف الگوریتم را توضیح خواهیم داد. سپس ویژگی‌های الگوریتم را بررسی کرده و در پایان نتایج مقایسه کارایی این الگوریتم با الگوریتم‌های موجود با استفاده از یکسری منبع داده محک‌زنی، ارائه خواهد شد.

۲- الگوریتم CLIP4

توصیف مفاهیم یادگیری، به صورت قواعد if-then یا درخت‌های تصمیم، از مطرح‌ترین روش‌های یادگیری‌ماشین می‌باشند. هر الگوریتم یادگیری ماشین هدایت‌شده قواعد دسته‌بندی را با استفاده از یک مجموعه‌داده اولیه بدست می‌آورد. این مجموعه‌داده به دو گروه مثبت و منفی تقسیم‌بندی می‌شوند. قواعد تولید باید گروه مثبت را توصیف کند و هیچ‌یک از موارد موجود در گروه منفی را شامل نشود. بسیاری از عملیاتی که توسط CLIP4 انجام می‌شود، از مسئلهSC[8] [4] استفاده می‌کند، به همین دلیل ابتدا توضیحی کلی در خصوص این مسئله خواهیم داد.

۲-۱- مسئله SC و کاربرد آن در الگوریتم CLIP4

چند تا از عملیات کلیدی که در الگوریتم CLIP4 انجام می‌شود، از مسئله SC استفاده می‌کند. این مسئله یک نسخه ساده‌تر از مدل برنامه‌نویسی خطی می‌باشد. مدل برنامه‌نویسی خطی، برای بهینه کردن یک تابع با درنظر گرفتن مجموعه‌ای از محدودیت‌ها بکار می‌رود. یکسری ساده‌سازی را اگر در مدل برنامه‌نویسی خطی انجام دهیم به مسئله SC تبدیل خواهد شد. تمام ضرایب تابع هدف و توابع استفاده شده در قسمت محدودیت‌ها مقدار ۱ را دارا می‌باشند، متغیرها از نوع دودویی هستند و تمام توابع محدودیت، بزرگتر مساوی ۱ هستند. مسئله SC یک مسئله NP-Hard است و بنابراین باید از راه‌حل‌های تقریبی برای آن استفاده کرد. شکل۱ مثالی از یک مسئله SC به صورت یک مدل برنامه‌نویسی خطی را نشان می‌دهد. در الگوریتم CLIP4 از فرم ماتریسی برای نمایش مسئله SC استفاده می‌شود. ماتریسی با عنوان BIN ساخته می‌شود که همانند ماتریس استفاده شده یک ماتریس دودویی بوده و ضرایب توابع محدودیت را مشخص می‌کند. نحوه ساخت آنرا در ادامه بررسی خواهیم کرد. تعداد ستون‌های این ماتریس در حقیقت، مشخص کننده صفات[9] موجود برای رکوردها و به ازای هر نمونه یا رکورد، حاوی یک سطر می‌باشد. در ادامه خواهیم دید که با استفاده از یک نمونه منفی و مقایسه آن با مقادیر موجود در صفات نمونه‌های مثبت منبع داده، اگر مقدار مساوی داشته باشند در سطر و ستون مربوط به آن در این ماتریس ۰ و در غیر اینصورت مقدار ۱ قرار داده می‌شود. به این ترتیب حل این مدل، حداقل ستون‌ها یا صفاتی را برای مجموعه نمونه‌های مثبت موجود، مشخص می‌کند که باعث تمایز بین این نمونه‌ها و نمونه منفی در نظر گرفته می‌شوند. جواب یک بردار است که ستون‌های انتخاب شده حاوی مقدار ۱ می‌باشند.

---
[7] accurate classification
[8] set covering
[9] attribute

در CLIP4 بدلیل استفاده زیاد از این مسئله نیاز به یک‌راه‌حل سریع برای آن وجود دارد. در CLIP3 از یک الگوریتم حریصانه[10] ساده برای حل مسئله SC استفاده می‌شود. در CLIP4 الگوریتمی‌برای حل این مسئله ارائه شده است که ستون‌ها را بر اساس کمینه‌کردن تعداد ۱ هایی که دارا می‌باشند انتخاب می‌کند. این الگوریتم باعث افزایش دقت محاسبات و در نتیجه تولید قواعد مناسب‌تری می‌شود. شبه‌کد الگوریتم استفاده شده و مقایسه آن با الگوریتم استفاده شده در CLIP3 در [۱] آمده است.

Minimize :

$$x_1 + x_2 + x_3 + x_4 + x_5 = Z$$

Subject to :

$$x_1 + x_3 + x_4 \geq 1$$
$$x_2 + x_3 + x_5 \geq 1$$
$$x_3 + x_4 + x_5 \geq 1$$

Minimize :

$$x_1 + x_2 + x_3 + x_4 + x_5 = Z$$

Subject to :

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq 1$$

$$\text{Solution}: Z = 2, \text{ when } x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$$

شکل۱: یک مسئله SC و راه‌حل آن با استفاده از دو فرم استاندارد و ماتریسی

۲-۲- جزئیات الگوریتم CLIP4

شبه کد الگوریتم CLIP4 در شکل۲ نمایش داده شده است. قبل از شرح مراحل مختلف الگوریتم نمادهای استفاده شده در شبه‌کد و الگوریتم را در ابتدا تعریف می‌کنیم. مجموعه نمونه‌های موجود برای یادگیری، را S می‌نامیم. مجموعه نمونه‌های مثبت را S_P و مجموعه نمونه‌های منفی را با S_N نمایش خواهیم داد. نمونه‌های مثبت کلاسی را مشخص می‌کنند که قواعد بر اساس آنها تولید خواهد شد و بقیه را نمونه‌های منفی می‌باشند. یک نمونه e از مجموعه‌ای از انتخاب‌گرها به صورت $s_j=[a_j=v_j]$ تشکیل شده است. الگوریتم CLIP4 قواعدی را به صورت زیر تولید خواهد کرد:

$IF(s_1 \wedge ... \wedge s_m) THEN\ class = class_i$ که در آن $s_i=(a_i \neq v_j)$ می‌باشد.

مجموعه‌های S_P و S_N به صورت ماتریس‌هایی مشخص می‌شوند. ماتریس نمونه‌های مثبت با POS نمایش داده شده است و N_{POS} تعداد نمونه‌های مثبت موجود یا به عبارت دیگر تعداد سطرهای ماتریس می‌باشد. در خصوص نمونه‌های منفی از NEG و برای نمایش تعداد آن از N_{NEG} استفاده شده است. همانگونه که قبلا اشاره شد، این الگوریتم از ماتریس‌هایی با عنوان BIN نیز استفاده می‌کند که دارای K ستون می‌باشد، که هر ستونی مشخص‌کننده یک صفت از نمونه‌ها می‌باشد. این ماتریس توسط الگوریتم CLIP4 مقداردهی شده و به صورت یک مسئله SC حل می‌شود. این الگوریتم در ۳ مرحله کار خود را انجام می‌دهد که در ادامه آنها را بررسی می‌کنیم.

مرحله اول:

داده‌های موجود در مجموعه نمونه‌های مثبت به زیرمجموعه‌هایی با داده‌های مشابه در یک ساختار درخت تصمیم تبدیل می‌شود. هر گره از درخت زیرمجموعه‌ای از داده‌ها را نشان می‌دهد. هر سطح درخت با استفاده از یک نمونه منفی ساخته می‌شود، به این صورت که از آن استفاده می‌شود تا انتخاب‌گری که می‌تواند بین این نمونه منفی و مجموعه نمونه‌های مثبت تمایز قائل شود را پیدا کنیم. در حین رشد درخت، از هرس‌کردن استفاده می‌کنیم تا داده‌های خطادار حذف شوند، از رشد زیاد درخت جلوگیری کنیم و زمان اجرا را کاهش دهیم.

[10] greedy algorithm

```
N0=1; Create POS0,1 consisting of entire SP; Create NEG consisting of entire SN // Initialize
for negi, i=1 to NNEG do {                                          // PHASE I STARTS
  for j=1 to Ni-1 do {                                              //for each POSi-1,j matrix
    for k=1 to K do {                                               //create new BINj matrix
      for l=1 to number of POSi-1,j rows do {
        if posl^i-1,j[k] = negi[k] then binl^j[k] = 0;
        if posl^i-1,j[k] ≠ negi[k] then binl^j[k] = 1;
        if posl^i-1,j[k] = '*' then binl^j[k] = 0;                  // missing value encountered
        if negi[k] = '*' then binl^j[k] = 0; }}                     // missing value encountered
    SOLj = SolveSCProblem(BINj);}
  PruneMatrices(POSi-1,j, SOLj , j=1,…,Ni-1);
  ApplyGeneticOperators(POSi-1,j , SOLj , j=1,…,Ni-1);
  Ni=1;                                                             //counter for POSi+1 matrices
  for j=1 to Ni-1 do {                                              //for each POSi,j matrix
    if POSi-1,j was not pruned or redundant then {
      for k=1 to K do {                                             //through entire solution vector
        if SOLj[k]=1 then {                                         //then create new POSi,Ni matrix
          for l=1 to number of POSi-1,j rows do {
            if posl^i-1,j[k]≠negi[k] then add posl^i-1,j to POSi,Ni matrix; }}
    Ni=Ni+1; }}}
  for j=1 to Ni do {          //for each POSi matrix check if it large enough to be not considered as a noise
    if number of rows of POSi,j < NoiseThreshold then { remove POSi,j from the tree; Ni=Ni-1;}}
  EliminateRedundantMatrices(POSi,j j=1,…,Ni); }
Create BIN matrix that consist of N_N_NEG columns and NPOS rows, and fill with zeros   // PHASE II STARTS
for i=1 to N_N_NEG do {                                             // for all tree leaves
  for j=1 to NPOS do {
    for k=1 to number of rows of POS_N_NEG,i do { if posj^0,0 = pos_k^N_NEG,i then binj[i]=1;}}}
SOL = SolveSCProblem (BIN);                                         // select best leaf node subsets
for i=1 to N_N_NEG do {                                             // through entire solution vector
  create BINi=NEG;
  if SOL[i]=1 then {                                                // back-project POSNneg,i matrix
    for j=1 to K do {
      for k=1 to NNEG do {
        if negk[j]='*' then bink^i[j]=0; else {
          for l=1 to number of rows of POS_N_NEG,i do { if pos_l^N_NEG,i[j]=negk[j] then bink^i[j]=0; }}}}}}
  for j=1 to K do {                                                 // convert BINi values to binary
    for k=1 to NNEG do { if bink^i[j]≠0 then bink^i[j]=0;}}
  SOLi = SolveSCProblem (BINi);
  for j=1 to K do {                                                 // start generation of i-th rule
    if SOLi[j]=1 then {                                             // add selectors to the rule
      for k=1 to NNEG do { if bink^i[j]=1 then add "aj≠negk[j]" selector to the Rulei; }}}}
                                                                    // PHASE III STARTS
best#covered=0; previous_best#covered=0;                            // holds # ex. covered by the rule
while best#covered≥0 do {                                           //until best rules are accepted
  for i=1 to N_N_NEG do {                                           // through all generated rules
    coversi=NPOS;                                                   // # examples covered by Rulei
    for j=1 to NPOS do {                                            // for all examples in POS
      for k=1 to K do {
        for l=1 to number of selectors in Rulei do {
          if al=k and vk=posj^0,1[k] then { coversi=coversi –1; j=j+1; }}}}   // example not covered by Rulei
    if best#covered<coversi then best#covered=coversi; best_rule=Rulei; }
  NPOS = NPOS – best#covered;
  if NPOS<StopThreshold or NPOS=0 then STOP;
  POS0,1 = POS0,1 – examples covered by best_rule;
  if (best#covered<previous_ best#covered/2 and best#covered<NPOS/2) then best#covered=-1;   // multiple rules
  previous_best#covered_examples=best#covered_examples; }
```

شکل۲: شبه‌کد الگوریتم CLIP4

خطوط ۲-۲۳ شبه‌کد این مرحله الگوریتم را نشان می‌دهد. درختی که گره‌هایش نمایش دهنده نمونه‌های مثبت می‌باشد با استفاده از یک روش بالا به پایین توسعه داده می‌شود. در سطح i ام درخت Ni گره خواهیم داشت که هر کدام با استفاده از ماتریسی با عنوان $POS_{i,j}(j=1,\ldots N_i)$ نمایش داده می‌شوند. این زیرمجموعه‌ها با استفاده از N_{i-1} زیرمجموعه سطح بالاتر و یک نمونه منفی neg_i تولید می‌شوند. هر زیرمجموعه با استفاده از neg_i به صورت ماتریسی از صفر و یک با عنوان BIN تبدیل می‌شود. که بعدا به صورت یک مسئله SC مدل شده و حل خواهد شد. با استفاده از حل ارائه شده برای مسئله SC، زیرشاخه‌های گره فعلی شکل می‌گیرند و ماتریس‌های POS مربوط به آنها مقداردهی می‌شود. به این ترتیب، از سطح اول تا برگ‌ها با استفاده از نمونه‌های منفی موجود، سطوح مختلف شکل گرفته و مجموعه نمونه‌های مثبت اولیه به زیرمجموعه‌هایی شکسته می‌شوند.

مرحله دوم:

دو معیار بکار برده می‌شود تا مجموعه‌ای از بهترین زیرمجموعه‌های نهایی(برگ‌های درخت) را انتخاب کنیم. یکی اینکه مجموعه‌های بزرگتر ترجیح داده می‌شوند، زیرا قواعد تولید شده از آنها عمومیت بیشتری خواهد داشت. دوم اینکه به یک معیار کامل‌بودن[11] نیاز داریم. به این منظور یک عملیات برگشتی برای هر زیرمجموعه موجود در برگ‌ها که حاوی نمونه‌های مثبت می‌باشد با استفاده از تمام نمونه منفی موجود انجام می‌شود، ماتریس بدست آمده به یک ماتریس دودویی تبدیل شده و به صورت یک مسئله SC حل می‌شود. از حل آن برای تولید یک قاعده استفاده می‌شود. اینکار برای تمام زیرمجموعه‌های موجود انجام می‌شود. خطوط ۲۴-۴۱ الگوریتم مربوط به این مرحله می‌باشند. تعداد برگ‌های موجود در درخت $N_{N_{NEG}}$ می‌باشد. از یک ماتریس دودویی BIN استفاده می‌شود تا یک مجموعه کمینه از برگ‌ها انتخاب شود که کلیه نمونه‌های مثبت را پوشش دهد. عملیات برگشتی برای تولید قاعده، بر روی زیرمجموعه‌های انتخاب شده انجام خواهد شد.

مرحله سوم:

مجموعه قواعد بهینه از میان قواعد موجود انتخاب می‌شود. قواعدی که نمونه‌های مثبت بیشتری را پوشش دهد انتخاب می‌شوند. اگر نتوان از این معیار استفاده کرد، از میان قواعد درگیر در تصمیم‌گیری قاعده‌ای انتخاب می‌شود که طول کمتری داشته باشد، یعنی قاعده‌ای که انتخاب‌گرهای کمتری داشته باشد. خطوط ۴۲-۵۵ الگوریتم شکل۲ مربوط به این مرحله می‌باشد. برای یافتن تعداد نمونه‌هایی که یک قاعده پوشش می‌دهد به صورت برعکس عمل می‌شود. یعنی تعداد نمونه‌هایی را که پوشش نمی‌دهد و سپس آنرا از تعداد کل نمونه‌های موجود کم می‌کنیم. اینکار به این علت انجام می‌شود که انتخاب‌گرهای موجود از نوع نامساوی هستند. متغیر $cover_i$ تعداد نمونه‌های مثبت پوشش داده شده توسط قاعده i ام را نشان می‌دهد. بعد از اینکه یک قاعده مورد قبول قرار گرفت، نمونه‌های متناظر آن از ماتریس POS حذف می‌شوند.

در این الگوریتم، از تعدادی از معیارهای آستانه‌ای[12]، برای هرس‌کردن، حذف نمونه‌های خطا دار و توقف الگوریتم استفاده می‌شود. این معیارهای آستانه‌ای استفاده شده را در بخش۵، بررسی خواهیم کرد.

نشان داده شده است که پیچیدگی زمانی این الگوریتم $O(k^3n^2)$ می‌باشد، که در آن k تعداد صفات نمونه‌ها و n تعداد نمونه‌های موجود می‌باشد.

[11] completeness
[12] threshold

۳- مثالی از کاربرد الگوریتم CLIP4

در این مثال، ۸ نمونه داریم که در جدول۱ نشان داده شده‌اند. ۳ نمونه، مقادیر نامشخص دارند که با * مشخص شده‌اند. کلاس۱، نمونه‌های مثبت و کلاس۲، نمونه‌های منفی می‌باشد.

Ex.#	F1	F2	F3	F4	Class
1	1	2	3	*	1
2	1	3	1	2	1
3	*	3	2	5	1
4	3	3	2	2	1
5	1	1	1	3	1
6	3	1	2	5	2
7	1	2	2	4	2
8	2	1	*	3	2

جدول۱ : نمونه‌های مثبت و منفی بکار برده شده در مثال

بنا به الگوریتم فوق، در ابتدا ماتریس POS برای نمونه‌های مثبت و ماتریس NEG برای نمونه‌های منفی، به صورت زیر تشکیل می‌شود:

$$NEG \begin{bmatrix} 3 & 1 & 2 & 5 \\ 1 & 2 & 2 & 4 \\ 2 & 1 & * & 3 \end{bmatrix} \quad POS \begin{bmatrix} 1 & 2 & 3 & * \\ 1 & 3 & 1 & 2 \\ * & 3 & 2 & 5 \\ 3 & 3 & 2 & 2 \\ 1 & 1 & 1 & 3 \end{bmatrix}$$

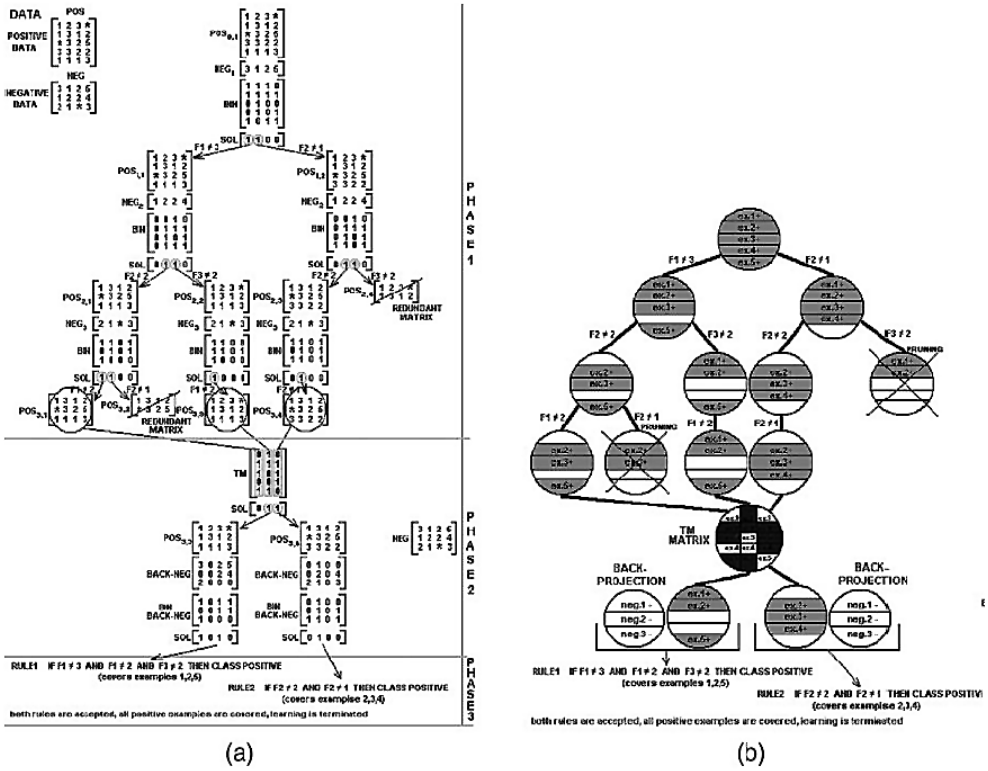
حال برای شروع الگوریتم ماتریس $POS_{0,1}$ با مقدار ماتریس POS مقداردهی می‌شود. اولین سطر ماتریس NEG، یعنی نمونه منفی را در نظر می‌گیریم: $NEG_1 = [3,1,2,5]$. حال ماتریس BIN به صورتیکه در الگوریتم آمده است با مقایسه مقادیر موجود در نمونه منفی و مقادیر موجود در ماتریس $POS_{0,1}$ به صورت زیر مقداردهی می‌شود:

$$BIN \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

این ماتریس به صورت یک مسئله SC، حل می‌شود. جواب عبارتست از: SOL=[1,1,0,0]. بنابراین با توجه به مقادیر نمونه منفی استفاده شده و SOL بدست آمده، ماتریس $POS_{0,1}$ با استفاده از دو قاعده F1≠3 و F2≠1 به دو ماتریس $POS_{1,2}$ و $POS_{1,1}$ شکسته می‌شود، که به عنوان فرزندان گره فعلی در درخت در نظر گرفته می‌شوند. بقیه مراحل به همین صورت ادامه پیدا می‌کند تا به ازای تمام نمونه‌های منفی سطح در درخت ایجاد شود و برگ‌های درخت شکل بگیرد. تولید ماتریس‌های مربوطه در شکل۳ دیده می‌شود. همانگونه که در شکل دیده می‌شود، برخی از ماتریس‌ها که تعداد نمونه کمی‌را شامل شده‌اند، حذف می‌شوند. پس از رسیدن به برگ‌ها، مرحله اول الگوریتم به پایان می‌رسد. در مرحله دوم، از میان برگ‌ها، یک مجموعه کمینه انتخاب می‌شود که بتواند کل مجموعه نمونه‌های مثبت را بپوشاند. یک ماتریس BIN دیگر که در اینجا نام آنرا TM گذاشته است، بر اساس رکوردهایی که ماتریس‌های برگ، یعنی $POS_{3,1}$ ، $POS_{3,2}$ و $POS_{3,3}$ از ماتریس POS که حاوی کلیه نمونه‌های مثبت می‌باشد، پوشش می‌دهند، تشکیل می‌شود و سپس

به صورت یک مسئله SC حل می‌شود و نتیجه آن به صورت SOL=[0,1,1] بدست می‌آید که نشان می‌دهد، دو مجموعه $POS_{3,2}$ و $POS_{3,3}$ برای پوشش مجموعه POS کافی می‌باشند. سپس ماتریس BACK_NEG برای هر کدام از این دو ماتریس ساخته می‌شود(مراحل می‌توانید در شکل۳ دنبال کنید). برای ساخت این ماتریس ابتدا تمام سلول‌های آن با ماتریس NEG مقداردهی می‌شود، سپس به ازای تمام ستون‌ها چک می‌شود که اگر مقداری که در یک سلول قرار دارد، در یکی از رکوردهای ماتریس متناظر POS در همان ستون وجود دارد، مقدار آن به صفر تغییر می‌کند. همچنین اگر مقدار موجود در ماتریس BACK_NEG برابر * یعنی مقدار آن نامشخص باشد، مقدار آن به صفر تغییر می‌کند. در انتها ماتریس BIN متناظر با جایگذاری مقادیر غیرصفر به ۱، تولید می‌شود. هر کدام از دو ماتریس BIN تولید شده به صورت یک مسئله SC حل می‌شود. به ازای هر حل، یک قاعده تولید می‌شود. قواعد تولید شده به صورت زیر می‌باشد:

RULE 1: If F1≠3 and F1≠2 and F3≠2 then class 1
RULE 2: If F2≠2 and F2≠1 then class 1



شکل۳: مثالی از تولید قواعد توسط الگوریتم CLIP4.(a) نمایش سطح پایین اجرا (b) نمایش سطح بالا

۴- ویژگی‌های الگوریتم CLIP4

الگوریتم CLIP4، یکسری از ویژگی‌های الگوریتم CLIP3 را حفظ کرده و ویژگی‌های جدیدی به آنها افزوده است. ویژگی‌های کلی این الگوریتم را در ادامه بررسی می‌کنیم.

۴-۱- رسیدگی به داده‌های نامشخص[۱۳]

داده‌ها یا مقادیر نامشخص، داده‌هایی هستند که در رکوردهای موجود در بانک‌اطلاعاتی وارد نشده‌اند. الگوریتم CLIP4، رکوردهایی را که مقادیر نامشخص دارند، را حذف نمی‌کند و همچنین آنها را با مقادیر پیش‌فرض یا محاسبه‌شده مقداردهی نمی‌کند. این الگوریتم از مقادیر موجود برای بقیه صفات رکورد استفاده کرده و از مقادیر نامشخص صرف‌نظر می‌کند. بنابراین در این الگوریتم کاربر نیازی به انجام پردازش بر روی داده‌های نامشخص موجود در منبع‌داده قبل از اعمال الگوریتم نخواهد داشت.

۴-۲- رسیدگی به صفات اسمی[۱۴] و تولید قاعده برای منابع‌داده با چند کلاس

الگوریتم CLIP4 داده‌های اسمی‌را در یک مرحله پیش‌پردازش به صورت عددی کدگذاری می‌کند. الگوریتم وابسته به شمای کدگذاری استفاده شده نیست، زیرا در هنگام تولید قواعد از معیاری که مرتبط با محاسبه فاصله باشد، استفاده نمی‌کند. همانگونه که دیدیم، این الگوریتم از مسئله SC استفاده می‌کند، که وابسته به مقادیر واقعی صفات نمی‌باشد. برای مواردی که چندکلاس داریم، الگوریتم در مراحل مختلف و در هر مرحله با در نظر گرفتن یکی از کلاس‌ها به عنوان کلاس مثبت، قواعد مربوط به آنرا تولید می‌کند.

۴-۳- معیارهای آستانه‌ای

در طول الگوریتم از استفاده شده است که برای هرس‌کردن و حذف نمونه‌های خطادار از آن استفاده می‌کند. آستانه‌های استفاده شده به صورت زیر می‌باشد:

آستانه خطا[۱۵](NT) مشخص می‌کند که چه گره‌هایی باید حذف شوند. این آستانه حداقل داده‌های موجود در یک زیرمجموعه را تعیین می‌کند. گره‌ای که زیرمجموعه مربوطه آن دارای تعداد نمونه‌های کمتر از این آستانه باشد، حذف خواهد شد.

آستانه هرس[۱۶](PT) نیز همانند آستانه خطا برای حذف گره‌ها از درخت مورد استفاده قرار می‌گیرد. الگوریتم از یک تابع کمی‌برای اندازه‌گیری میزان شایستگی[۱۷] یک گره استفاده می‌کند. سپس تعداد PT گره به ترتیب میزان شایستگی آنها انتخاب می‌شوند.

آستانه توقف[۱۸](ST) زمان توقف الگوریتم را مشخص می‌کند. الگوریتم وقتی متوقف می‌شود که تعداد نمونه مثبت کمتر از ST پوشش نداده شده توسط قواعد باقی بماند. همانگونه که قبلا گفته شد، وقتی قاعده‌ای انتخاب شد، نمونه‌های پوشش داده شده توسط آن قاعده از ماتریس POS که حاوی نمونه‌های مثبت موجود می‌باشد، حذف می‌شوند.

---

[13] missing values
[14] nominal
[15] noise threshold
[16] pruning threshold
[17] fitness
[18] stop threshold

۵- نتایج تجربی

الگوریتم CLIP4 آنگونه که در [1] آمده است، بر روی چند مسئله محک‌زنی تست شده است. تست‌های انجام شده برای مقایسه این الگوریتم با الگوریتم‌های موجود و همچنین تست‌هایی برای مشخص کردن ویژگی‌های خاص الگوریتم می‌باشد.

۵-۱- منابع داده

CLIP4 بر روی ۲۹ منبع داده تست شده است. مشخصات این منابع داده را به صورت زیر می‌توان دسته بندی کرد:

- اندازه منابع داده یادگیری[19]: بین ۱۵۰ تا ۳۰ هزار نمونه
- اندازه منابع داده تست: بین ۴۵ تا ۵۶۵ هزار نمونه
- تعداد صفات: بین ۴ تا ۱۵۵۸
- تعداد کلاس‌ها: بین ۲ تا ۱۰

منابع داده از طریق مخزن منابع اطلاعاتی موجود در [6] و [7] بدست آمده است. اطلاعات کلی این منابع داده استفاده شده که در جدول۲، که با حروف اختصار نمایش داده شده است، در [1] آورده شده است.

۵-۲- مقایسه با سایر الگوریتم‌ها

نرخ خطا، تعداد قواعد، تعداد انتخاب‌گرها و زمان اجرای CPU برای الگوریتم CLIP4 با استفاده از ۱۶ منبع داده در جدول۲، دیده می‌شود. نتایج بدست آمده از الگوریتم CLIP4 با نتایج بدست آمده در [8] مقایسه شده است. در [8] کارایی ۳۳ الگوریتم یادگیری ماشین با یکدیگر مقایسه شده‌اند. در جدول۲، تنها مقادیر بیشینه و کمینه برای موارد تستی مختلف در کل ۳۳ الگوریتم تست شده آورده شده است. همانگونه که دیده می‌شود، میانگین نرخ خطا برای الگوریتم CLIP4 عدد ۲۵,۱ درصد بدست آمده است. کمترین میزان میانگین نرخ خطا مربوط به الگوریتم POLYCLASS [9] می‌باشد که مقدار ۱۹,۵ درصد را بدست آورده است. در [8] معیاری با عنوان اهمیت آماری برای نرخ‌های خطا در نظر گرفته شده است و نشان داده شده است که تفاوت میانگین نرخ خطای دو الگوریتم در یک سطح ۱۰ درصد در صورتیکه از ۵,۹ تجاوز کند، اهمیت دارد. بنابراین الگوریتم CLIP4 نیز در گروه مشابه همان الگوریتم قرار می‌گیرد و به این صورت یکی از الگوریتم‌های یادگیری ماشین خوب به حساب می‌آید.

زمان اجرای CPU در الگوریتم CLIP4 برای ۱۶ منبع داده، در جدول۲، دیده می‌شود. میانگین زمان اجرای الگوریتم، ۵,۸ دقیقه می‌باشد. نتایج [8] نشان می‌دهد که میانگین زمان اجرا برای ۳۳ الگوریتم تست شده بین ۶,۹ ثانیه و ۴۶,۸ ساعت بوده است،. که آنها را به دو دسته کمتر از ۱۰ دقیقه (۲۲ الگوریتم) و بیشتر از ۱۰ دقیقه (۱۱ الگوریتم) تقسیم‌بندی کرده است. این الگوریتم در دسته اول قرار می‌گیرد.

میانگین تعداد قواعد تولید شده توسط CLIP4، ۱۶,۸ می‌باشد. [8] میانگین تعداد برگ‌های درخت تولید شده توسط ۲۱ الگوریتم درخت تصمیم تست شده(که برابر تعداد قواعد تولید شده می‌باشد)، برابر ۱۷,۸ می‌باشد. همانگونه که دیده می‌شود، الگوریتم CLIP4، از لحاظ میانگین قواعد تولید شده توسط آن میزان کمتری را دارا می‌باشد. به طور کلی، ۱۰ الگوریتم از ۲۱ الگوریتم‌های تست شده در [8]، میانگین قواعد تولید شده توسط آنها کمتر از الگوریتم CLIP4 می‌باشد.

میانگین تعداد انتخاب‌گرهای تولید شده توسط الگوریتم CLIP4 برابر ۱۶ منبع داده برابر ۵۸۹,۵ می‌باشد که به ازای هر قاعده تعداد ۳۵ انتخاب‌گر بدست می‌آید.

[19] training dataset

منبع‌داده	نرخ‌های خطای گزارش شده برای ۳۳ الگوریتم یادگیری ماشین		نرخ خطای CLIP4	میانگین گزارش‌شده برای تعداد قواعد تولید شده توسط ۲۱ الگوریتم	تعداد قواعد تولید شده توسط CLIP4	زمان CPU گزارش شده برای ۳۳ الگوریتم یادگیری ماشین		زمان CPU الگوریتم CLIP4 (ثانیه)	تعداد انتخاب‌گرهای قواعد CLIP4
	کمینه	بیشینه				کمینه(s)	بیشینه(h)		
bcw	3	9	5	7	4.2	4	2.7	5.1	121.6
cmc	43	60	53	15	8	12	23.9	46	60.7
dna	5	38	9	13	8	2	475.2	662.8	90
hea	14	34	28	6	11.6	4	3.3	2.2	192.3
bos	22	31	29	11	10.5	9	5.5	35.8	133.5
led	27	82	29	24	41	1	12.4	166.4	189
bld	28	43	37	10	9.7	5	1.5	6.6	272.4
pid	22	31	29	7	4	7	2.5	5.8	64.1
sat	10	40	20	63	61	8	73.2	3696.7	3199
seg	2	52	14	39	39.2	28	75.6	614.6	1169.9
smo	30	45	32	2	18	1	3.8	90.5	242
thy	1	89	1	12	4	3	16.1	164.2	119
veh	15	49	44	38	21.3	14	14.1	45.3	380.7
vot	4	6	6	2	9.7	2	25.2	4.4	51.7
wav	15	48	25	16	9	4	4.3	43.1	85
tae	33	69	40	20	9.3	6	10.2	0.7	273.2
میانگین	۱۷,۱	۴۵,۴	۲۵,۱	۱۷,۸	۱۶,۸	۶,۹	۴۶,۸	۵,۸ دقیقه	۵۹۸,۵

جدول۲: نتایج مقایسه با سایر الگوریتم‌ها

۵-۳- تست قابلیت انعطاف[20]

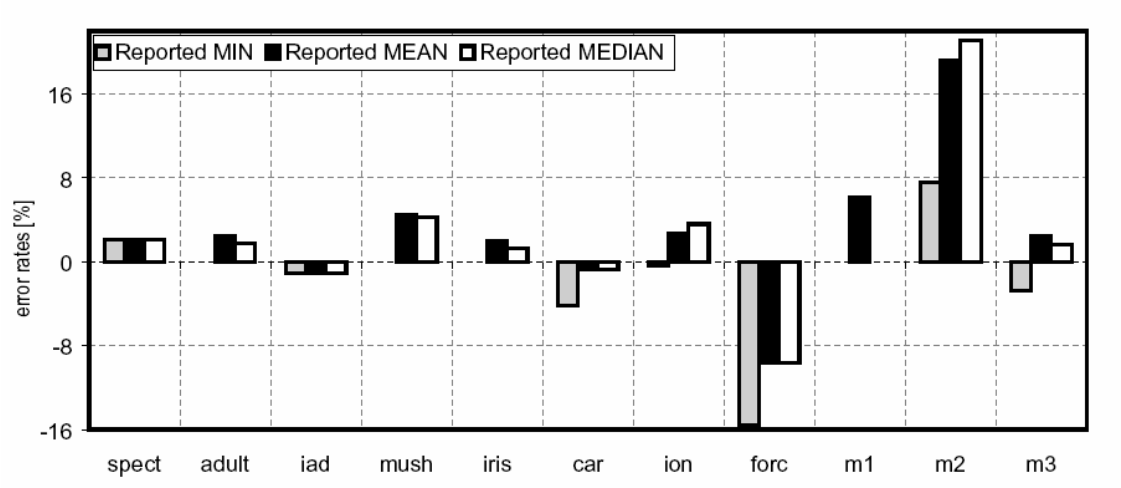
تست‌های انجام شده، شامل اندازه‌گیری نرخ‌خطا، تعداد قواعد تولید شده، تعداد انتخاب‌گرها و زمان اجرای CPU برای ۱۱ منبع داده می‌باشد. تست‌ها نشان می‌دهند که بسته به نوع معیارهای آستانه‌ای انتخاب شده، نتایج متفاوتی خواهیم داشت. جدول مربوط به نتایج در [1] آمده است. نتایج نشان می‌دهد که در منبع داده adult، نرخ‌خطا ۳,۱ درصد افزایش خواهد یافت، اگر بخواهیم که تعداد قواعد تولید شده را ۳,۵ برابر کاهش دهیم. و همین افزایش نرخ خطا در ازای تولید تعداد قواعد کمتر برای دیگر منابع داده نیز تکرار شده است.

قابلیت انعطاف الگوریتم با بکارگیری آن برای منبع داده adult که تعداد نمونه(رکورد) زیادی دارد و همچنین منبع داده iad که تعداد صفات زیادی دارد، تست شده است. نرخ‌خطا برای منبع‌داده adult مشابه کمترین نرخ گزارش شده می‌باشد، که مربوط به الگوریتم FSS Naïve Bayes می‌باشد. نرخ‌خطا برای منبع داده iad برابر ۳,۹ درصد می‌باشد. بهترین نرخ‌خطای گزارش شده برای این منبع داده، الگوریتم C4.5 می‌باشد که نرخ‌خطای آن ۲,۸ درصد می‌باشد. ولی این در حالی است که تعداد قواعد تولید شده توسط CLIP4 برابر ۶,۷ قاعده می‌باشد، که ۳,۷ برابر کمتر از تعداد قواعد تولید شده توسط C4.5 می‌باشد. همچنین الگوریتم CLIP4 برای این منبع داده از ۳۰ انتخاب‌گر در هر قاعده استفاده می‌کند. لازم به ذکر است که این منبع داده دارای ۱۵۵۸ صفت می‌باشد. بنابراین می‌توان گفت که الگوریتم CLIP4 در

[20] flexibility

مورد منابع داده با تعداد صفات زیاد، به خوبی عمل می‌کند و با تولید تعداد قواعد کمتر، تقریبا نرخ‌خطای زیـادی را نیـز تولید دربرندارد.

شکل۴، تفاوت نرخ‌خطای بدست آمده توسط الگوریتم CLIP4، در مقایسه با نتایج گزارش شده در مقالات برای ۱۱ منبع داده نشان داده شده است. مقادیر مثبت، یعنی CLIP4 بهتر از بقیه الگوریتم‌های گزارش شده عمل کرده است.



شکل۴: مقایسه بین نرخ‌خطاهای گزارش شده و نرخ‌خطای CLIP4(مقادیر مثبت بهبود را نشان می‌دهد)

۶- جمع‌بندی و نتیجه‌گیری

در این مقاله جزئیات الگوریتم CLIP4 را بررسـی کـردیم. ایـن الگـوریتم یکـی از الگـوریتم‌هـای یـادگیری ماشـین استقرایی به حساب می‌آید که برای دسته‌بندی[۲۱] مورد استفاده قرار می‌گیرد. این الگـوریتم از یـک ایـده ترکیبـی از الگوریتم‌های مبتنی بر قاعده و الگوریتم‌های درخت تصمیم، استفاده می‌کند. از ویژگی‌های بارز آن مـی‌تـوان تولیـد قواعدی که حاوی نامساوی هستند و رسیدگی به منابع داده با ابعاد زیاد را ذکر کرد. تولید قواعد نامساوی این امکان را می‌دهد که برای یک مفهوم خاص به تعداد قواعد کمتری نیاز داشته باشیم، و همانگونه که قسمت بررسـی نتـایج آزمایشات انجام شده بر روی این الگوریتم مشاهده شد، در برخی از موارد می توانـد بـا تولیـد تعـداد قاعـده کمتـر، نرخ‌خطای قابل قبولی را نیز به همراه داشته باشد. این الگوریتم علاوه بر دسته‌بندی در موارد دیگری مانند تشخیص مهمترین صفات موجود در مجموعه داده نیز کاربرد دارد. همانگونه که در قسـمت مقدمـه ذکـر شـد، چهـار فـاکتور مهمی که یک الگوریتم یادگیری ماشین که برای دسته‌بندی بکار می‌رود، باید دارا باشد عبارتند از: دقت دسته‌بندی دقیق، قواعد ساده، تولید کارای قواعد و انعطاف‌پذیری می‌باشد. نشان داده شـد کـه اجـرای ایـن الگـوریتم بـر روی منابع‌داده مختلف، در حضور داده‌های خطادار و نامشخص امکان‌پذیر است و نرخ‌خطای بدست آمده از لحاظ آمـاری شبیه به بهترین الگوریتم گزارش شده می‌باشد. همچنین استفاده از هرس‌کردن‌های پی‌درپی در طول اجرا و مسـئله SC، به الگوریتم توانایی تولید یک مجموعه کمینه از قواعد مختصر را می‌دهـد. همچنـین زمـان اجـرای مـورد نیـاز الگوریتم بدلیل استفاده از هرس‌کردن به میزان قابل قبولی کاهش یافته است و همانگونه که دیده شـد، در بـین ۲۲ الگوریتم از ۳۳ الگوریتم موجود قرار گرفت. همچنین با آزمایش‌های انجام شده، مشخص شد که الگوریتم از قابلیـت انعطاف بالایی برخوردار است و می‌تواند رسیدگی به منابع‌داده با نمونه‌های زیاد و همچنین مجموعه صفات زیـاد را، به صورت قابل قبولی انجام دهد.

[21] classification

مراجع:

[1] K.J. Cios, L.A. Kurgan, CLIP4: Hybrid inductive machine learning algorithm that generates inequality rules, Information Sciences 163 (2004) 37–83.

[2] K.J. Cios, N. Liu, An Algorithm which Learns Multiple Covers via Integer Linear Programming. Part I—The CLILP2 Algorithm, Kybernetes, 24:2, pp. 29–50 (The Norbert Wiener Outstanding Paper Award), 1995.

[3] K.J. Cios, D.K. Wedding, N. Liu, CLIP3: Cover learning using integer programming, Kybernetes 26 (4–5) (1997) 513–536.

[4] M. Bellmore, H.D. Ratliff, Set covering and involuntary bases, Management Science 18 (1971)3.

[5] P. Clark, R. Boswell, Rule induction with CN2: Some recent improvements, in: Y. Kodratoff (Ed.), Lecture Notes in Artificial Intelligence, Proceedings of European Working Session on Learning, Springer-Verlag, 1991, pp. 151–163.

[6] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases. Available from <http://www.ics.uci.edu/~mlearn/MLRepository.html>. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[7] P. Vlachos, StatLib Project Repository. Available from <http://lib.stat.cmu.edu>, 2000.

[8] T.-S. Lim, W-Y. Loh, Y-S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine Learning 40 (2000) 203–228.

[9] C. Kooperberg, S. Bose, C.J. Stone, Polychotomous regression, Journal of the American Statistical Association 92 (1997) 117–127.