Drzewa decyzyjne

Oparte na drzewie algorytmy uczące są szeroką i popularną rodziną powiązanych ze sobą i pozbawionych parametrów nadzorowanych metod klasyfikowania i regresji. Podstawą tych klasyfikatorów jest tak zwane drzewo decyzyjne. Połączono w nim serie reguł decyzyjnych, np. „jeśli płeć to mężczyzna, to …". Nazwa grupy algorytmów nie jest przypadkowa. Otóż pełny graf przypomina odwrócone drzewo z **korzeniem** (ang. *root node*, wysokość 0 – węzeł na samej górze grafu), gdzie każdy kolejny **węzeł decyzyjny** to właściwie reguła decyzyjna i przejście do następnego węzła. Gałąź bez reguły decyzyjnej na końcu nosi nazwę **liścia**.

Największą **zaletą** drzew decyzyjnych jest ich łatwość wytłumaczenia nietechnicznym osobom, ale nie tylko:

- niektórzy ludzie uważają, że drzewa decyzyjne bliżej odzwierciedlają nasz sposób, wnioskowania niż inne metody regresji i klasyfikacji,
- drzewa mogą być łatwo wyświetlane i prezentowane graficznie,
- drzewa łatwo radzą sobie z predykatorami jakościowymi i nie potrzebują tworzenia fikcyjnych zmiennych (ang. *dummy variable*).

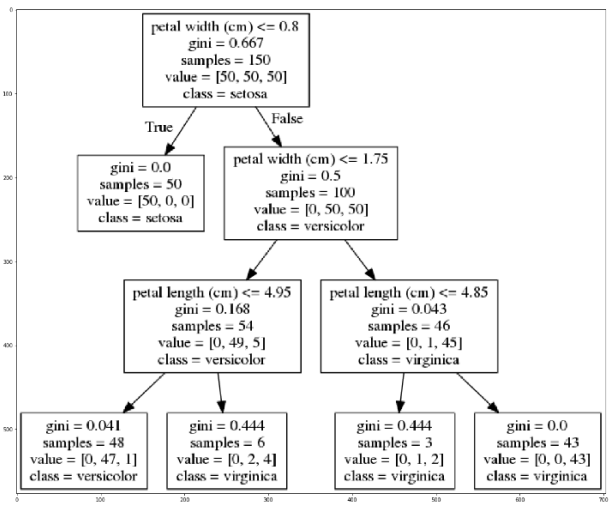
Drzewa decyzyjne posiadają niestety swoje **wady**:

- stosunkowo niska dokładność predykcji w porównaniu do innych, popularnych algorytmów regresji i klasyfikacji,
- drzewa bywają bardzo nieodporne, innymi słowy mała zmiana w zbiorze danych może spowodować znaczne zmiany w ostatecznej estymacji.

Właśnie z powyższych powodów pojedyncze drzewa, poprzez ich agregację, są z reguły wykorzystywane w takich algorytmach jak *bagging*, *boosting* czy *random forests*. Efektywność rozwiązań korzystających z dobrodziejstw drzew decyzyjnych należy do najlepszych w grupie algorytmów uczenia maszynowego!

Przykładowa realizacja klasyfikacji za pomocą modułu Scikit-Learn

Pełny kod znaleźć można w pliku *tree_sklearn.ipynb*. Wykorzystano w nim zbiór danych Iris. Składa się on z 150 obiektów przyporządkowanych do trzech klas/odmian gatunków Irysów (kosaćców): Setosa, Versicolour oraz Virginica . Każdy z nich posiada po cztery parametry: długość i szerokość listka kielicha kwiatowego (ang. *sepal*) oraz długość i szerokość płatka kwiatu (ang. *petal*). W przykładzie, do budowania drzewa decyzyjnego, wykorzystano klasę *DecisionTreeClassifier* z modułu *sklearn.tree* biblioteki *Scikit-Learn*. Wynik działania skryptu *tree_sklearn.ipynb* zaprezentowano poniżej:



Jak wnioskować na podstawie takiego drzewa? Załóżmy, że znaleźliśmy irysa i chcemy go sklasyfikować za pomocą powyższych reguł decyzyjnych. Zaczynając od korzenia, zadajemy serię pytań: czy szerokość płatka jest <= 0.8 cm? Jeśli tak to przechodzimy do lewego węzła potomnego. W tym przypadku to liść, gdyż nie posiada kolejnej reguły decyzyjnej. Uzyskujemy w ten sposób odpowiedź: irys którego znaleźliśmy to odmiana Setosa. Gdyby długość była > 0.8 cm to przechodzimy do prawego węzła i znajdujemy kolejną regułę. Odpowiadamy na pytanie: czy szerokość płatka jest <= 1.75 cm. Węzły potomne nie są liśćmi, dlatego musimy odpowiedzieć na kolejne pytanie w zależności od udzielonej przed chwilę odpowiedzi. Algorytm nie jest specjalnie skomplikowany.

Wysokość tego drzewa wynosi 3. Atrybut *sample* węzła zlicza liczbę wyznaczonych do niego próbek uczących. Z kolei *value* mówi jak wiele przykładów uczących z każdej klasy przynależy do danego węzła. Atrybut *gini* stanowi **miarę zanieczyszczenia** (ang. *impurity*). Węzeł jest „czysty" (*gini* = 0), jeśli wszystkie znajdujące się w nim próbki uczące należą do tej samej klasy. **Wskaźnik Giniego** (G_i) można wyliczyć z równania:

$$G_i = 1 - \sum_{k=1}^{n} \left(p_{i,k}\right)^2$$

gdzie: $p_{i,k}$ - współczynnik występowania klas *k* wśród próbek uczących w *i*-tym węźle.
Inną miarą zanieczyszczenia jest **entropia**:

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \, log\left(p_{i,k}\right)$$

Samo pojęcie entropii wywodzi się z termodynamiki, gdzie służy do opisu miary nieuporządkowania cząsteczek. Wartość entropii wynosi 0, gdy cząsteczki są nieruchome i uporządkowane w przestrzeni. Koncepcja tej miary wkradła się również do świata informatyki, m.in. do uczenia maszynowego, jak widać, ale nie tylko. Entropia jest niezwykle istotna chociażby w przypadku generatorów liczb pseudolosowych. Który z powyższych współczynników zanieczyszczenia wybrać do budowy drzewa? Otóż w większości przypadków użytkownik za ich pomocą uzyska podobne drzewa. Jednak wskaźnik Giniego obliczany jest nieznacznie szybciej, co może stanowić wskazówkę przy wyborze jednej z dwóch miar.

Mamy tutaj gotową realizujące zadanie klasyfikacji za pomocą drzew decyzyjnych. Rozwiązania „*out of the box*" są jednak mało interesujące. Zadać można jednak pytanie: jak wybrać korzeń drzewa i tworzyć jego kolejne poziomy oraz rozgałęzienia? Na tę potrzebę opracowano algorytmy tworzenia drzew, m.in. **ID3** (ang. *Iterative Dichotomiser 3*) oraz **CART** (ang. *Classification And Regression Tree*). Moduł *Scikit-Learn* korzysta z algorytmu CART do uczenia drzew decyzyjnych, inaczej ich wzrostu, dlatego też on zostanie tutaj przedstawiony. Na wstępie należy zaznaczyć, iż CART jest algorytmem zachłannym (ang. *greedy algorithm*), a to znaczy, że poszukuje zawsze optimum (minimum lub maksimum) lokalnego funkcji, które nie koniecznie musi być choćby blisko optimum globalnego. Nie mniej, algorytmy zachłanne często dają dobre wyniki, nie muszą być one jednak optymalne. Niestety, poszukiwanie optymalnego drzewa jest klasyfikowane jako problem NP-zupełny. Stąd też musi wystarczyć rozwiązanie dobre, które niekoniecznie jest najlepsze.

Jak zatem działa CART? Zastosowany tutaj mechanizm jest całkiem prosty: algorytm rozdziela najpierw dane uczące na dwa podzbiory przy użyciu pojedynczej cechy *k* i progu t_k (u nas szerokość płatka <= 0.8 cm). Jak dobrać wartości *k* i t_k? Wyszukiwana jest para parametrów (k, t_k) generująca najczystsze podzbiory, ważone pod względem rozmiaru. **Funkcja kosztu**, jaką CART stara się minimalizować, opisana jest wzorem:

$$J(k, t_k) = \frac{m_{lewy}}{m} G_{lewy} + \frac{m_{prawy}}{m} G_{prawy}$$

gdzie: $G_{lewy/prawy}$ - współczynnik Giniego lewego/prawego podzbioru, $m_{lewy/prawy}$ – liczba próbek w lewym/prawym podzbiorze.

Dane rozdzielone na dwa podzbiory poddawane są kolejnym podziałom i tak aż do osiągnięcia określnego warunku zakończenia pracy algorytmu. Co może być takim warunkiem zakończenia pracy algorytmu? Na przykład: maksymalna wysokość drzewa lub brak możliwości określenia podziału, który zmniejszyłby zanieczyszczenie. Zastosować można także inne warunki zatrzymania budowy drzewa. Zachęca się czytelnika do przejrzenia literatury przedmiotu.

Przykładowa, „ręczna" realizacja działania pierwszego etapu algorytmu CART, czyli wyboru korzenia, przedstawiona została w pliku *cart_algo.ipynb*. Zwrócić należy uwagę na wybraną regułę decyzyjną oraz uzyskane w ten sposób rozgałęzienie. Doprecyzowania wymaga jedynie jeden podpunkt finalnych wyników rozdziału: wartość *Final score* reprezentuje tutaj końcową wartość funkcji kosztu $J(k, t_k)$. Pozostaje wartości powinny być zrozumiałe po lekturze powyższego opracowania.

Wykorzystano materiały:

- Albon Ch. (2019) „Uczenie maszynowe w Pythonie. Receptury", Helion
- James G. i inni (2015) „An Introduction to Statistical Learning with Application in R", Springer
- Geron A. (2018) „Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow", Helion
- Autor przykładów kodu: Piotr Gnyś

Autor: Paweł Przestrzelski