Drzewo, wyrażenie regularne

Mateusz Czarnewicz-Cyfra
Małgorzata Grzanka

Oryginalny opis tematu

Drzewo decyzyjne w zadaniu klasyfikacji miejsc rozcięcia w sekwencji DNA. Należy użyć wyrażenia regularnego w testach. Wyrażenia regularne powinny być wczytane z pliku tekstowego. Dopasowanie wyrażenia sprawdzamy na rozważanej pozycji w sekwencji, np. jeśli rozważamy atrybut numer 3, a wyrażenie to "A.T", to w sekwencji "GGAGT" nastąpi dopasowanie, a w sekwencji "AGTGG" już nie. Więcej informacji o specyfice problemu znaleźć można w: https://staff.elka.pw.edu.pl/~rbiedrzy/UMA/opisDNA.html. Dane do pobrania- donory: https://staff.elka.pw.edu.pl/~rbiedrzy/UMA/spliceDTrainKIS.dat, akceptory: https://staff.elka.pw.edu.pl/~rbiedrzy/UMA/spliceATrainKIS.dat. Przed rozpoczęciem realizacji projektu proszę zapoznać się z zawartością: https://staff.elka.pw.edu.pl/~rbiedrzy/UMA/index.html.

Interpretacja tematu

Celem zadania jest stworzenie drzewa decyzyjnego, którego zadaniem będzie stwierdzenie, czy dana sekwencja genów jest miejscem rozcięcia, czy nie (***zadanie klasyfikacji)***. Drzewo podczas treningu i inferencji będzie wykorzystywać wyrażenia regularne, których przygotowanie także jest częścią zadania.

Opis algorytmu

Planowana implementacja będzie składała się z dwóch elementów - tworzenie wyrażeń regularnych i budowa drzewa decyzyjnego.

Tworzenie wyrażeń regularnych

Naszym pomysłem stworzenia wyrażeń regularnych jest Byte-Pair Encoding (BPE) tokenizer. Algorytm działałby następująco (dokładnie tak jak typowy BPE):

1. Najpierw rozbijamy wszystkie sekwencje genów na pojedyncze znaki i liczymy ich częstotliwość występowania w danych.
2. W każdej iteracji liczymy, które pary występują najczęściej obok siebie.
3. Parę, która występuje najczęściej, łączymy i zapisujemy w słowniku. Wszystkie wystąpienia tej pary w danych także łączymy.
4. Dopóki zadana wielkość słownika nie zostanie osiągnięta, wracamy do punktu 2.

Algorytm ten stosowany jest jedynie do przykładów z klasy pozytywnej. Dzięki temu powstaje lista najczęściej występujących połączeń genów w donorach/akceptorach.

Nie jest to idealne rozwiązanie, ponieważ zakłada całkowite dopasowanie do wzorca. Możliwym jest, że czasami sekwencja może różnić się jedynie jednym genem w sekwencji i dalej powinna być tak samo klasyfikowana. Do osiągnięcia lepszych wyników można spróbować dodać do tego algorytmu uogólnienie (w odpowiednich sekwencjach postawić “kropkę” symbolizującą dowolny gen).

Parametry tokenizer’a:

- vocab_size - rozmiar słownika wyrażeń regularnych (domyślnie 20)

Budowa drzewa decyzyjnego

Przygotowanie danych dla drzewa decyzyjnego:

Stworzone przez tokenizer wyrażenia regularne przygotowujemy w następujący sposób:

1. **Ekstrakcja cech** - aby odpowiednio wprowadzić dane do modelu, proponujemy następującą konwencję. Dla każdej sekwencji DNA tworzymy wektor cech binarnych, gdzie każda cecha odpowiada wystąpieniu danego wyrażenia regularnego na określonej pozycji względem miejsca rozcięcia. Cecha ma postać `pat{i}@{offset}`, gdzie “i” to indeks wyrażenia regularnego w zbiorze wyrażeń regularnych, a “offset” to numer genu w sekwencji.
2. **Selekcja cech** - ze względu na potencjalnie duży rozmiar takiej struktury, planujemy wykorzystać wzajemną informację (mutual information) do wyboru najbardziej istotnych cech. Pozwoli to na redukcję wymiarowości i usunięcie cech nieistotnych dla klasyfikacji.

Na podstawie tak przygotowanych danych, budujemy drzewo decyzyjne. Będzie to implementacja całkowicie zgodna ze standardową implementacją drzew decyzyjnych.

Drzewo tworzone jest rekurencyjnie, zaczynając od pierwszego węzła (root). Dla każdego węzła wykonujemy następującą serię akcji:

1. **Kryterium podziału** - wybieramy podział danych należących do węzła, który daje największy zysk informacji. Do liczenia zysku informacji, będziemy używać współczynnika entropii. Kryterium podziału to obecność określonego wyrażenia regularnego na określonej pozycji w próbce (wartość 1 w macierzy pat{i}@{offset}).
2. **Podział danych** - dzielimy zbiór danych na elementy spełniające i niespełniające danej cechy (czyli czy wyrażenie regularne dopasowało się na danej pozycji).
3. **Tworzenie liści** - dla każdego z otrzymanych podziałów, jeżeli osiągnięto zadaną głębokość drzewa (max_depth) lub dane zawierają elementy tylko jednej klasy (lub spełniają warunek min_samples_leaf), tworzymy liść z klasą dominującą. W przeciwnym wypadku powtarzamy od punktu 3.

Parametry drzewa decyzyjnego

Planujemy uwzględnić takie parametry:

- max_depth - maksymalna głębokość drzewa (domyślnie 6)

- min_samples_leaf - minimalna liczba próbek w liściu (domyślnie 10)

Planowane eksperymenty

Jakość wyników może zależeć od założonych przez nas atrybutów drzewa (maksymalna głębokość, minimalna liczba próbek w liściu) oraz samych danych wejściowych. Zatem, planujemy wykonać następujące eksperymenty:

- Różne głębokości drzewa decyzyjnego (zbyt duża będzie prowadzić prawdopodobnie do większego przeuczenia, a zbyt mała - do niedouczenia)
- Różne wartości minimalnych liczb próbek w liściu (analogicznie)
- Różne wielkości słownika tokenizer'a oraz ewentualne zastosowanie/niezastosowanie generalizacji wyrażeń regularnych.
- Wykorzystanie i niewykorzystanie selekcji za pomocą mutual information

Dla każdego z planowanych eksperymentów kryterium porównawczym będą różne miary jakości.

Miary jakości

Stosowanymi miarami jakości będą:

- ACCURANCY (dokładność)
- CONFUSION MATRIX (macierz pomyłek)
- PRECISION (precyzja)
- RECALL (czułość)
- SPECIFICITY (specyficzność)
- F1-SCORE
- Analiza przeuczenia - porównywana będzie dokładność na zbiorze treningowym i testowym.

Porównamy także naszą implementację z gotową implementacją drzewa decyzyjnego w bibliotece *scikit-learn.*

Sposób weryfikacji poprawności implementacji

Będziemy stosować:

- testy jednostkowe dla pojedynczych funkcji: poprawność działania BPE (na małych sekwencjach), dopasowanie regexów (na krótkich przykładach), poprawność obliczeń cech,
- porównanie predykcji drzewa z implementacją scikit-learn (gdy cechy są identyczne),
- ręczne przejście przez drzewo na przykładach z małego zbioru testowego.

Opis zbioru danych

Model będzie trenowany i testowany na dwóch udostępnionych zbiorach danych. Każdy z nich zawiera przykłady klasyfikacji jednego z dwóch rodzajów miejsc rozcięcia sekwencji kodującej białko:

- donory (miejsca rozcięcia typu donor), 10513 wartości https://staff.elka.pw.edu.pl/~rbiedrzy/UMA/spliceDTrainKIS.dat
- akceptory (miejsca rozcięcia typu akceptor), 11577 wartości https://staff.elka.pw.edu.pl/~rbiedrzy/UMA/spliceATrainKIS.dat

Struktura danych

W pierwszej linii każdego pliku napisano, na której pozycji (licząc litery od lewej strony) we fragmentach sekwencji jest granica pomiędzy intronem a eksonem. Dalej w pliku występują parami:

- Linia określająca czy jest to przykład pozytywny (1) czy negatywny (0)
- Sam przykład czyli sekwencja DNA

Podział danych

Zostanie zastosowana walidacja krzyżowa. Dane zostaną podzielone na zbiór treningowy (80%) i testowy (20%) zachowując stosunek przypadków pozytywnych do negatywnych.