**Markiewicz Mikołaj**

**Opasiak Krzysztof**

**Algorytmy Heurystyczne**

**Projekt C5**

**Treść zadania:**

C5. Selekcja modeli regresji: przeszukiwanie przestrzeni podzbiorów zespołu modeli regresji utworzonego przez zastosowanie dostępnego w R algorytmu regresji do kopii zbioru danych zaburzonych przez próbkowanie przykładów i/lub atrybutów z funkcją oceny opartą na

jakości predykcji uzyskiwanych przez uśrednianie modeli z ocenianego podzbioru.

**Interpretacja treści zadania:**

Danymi wejściowymi właściwego algorytmu jest zbiór modeli wygenerowanych przez zastosowanie funkcji *lm/rpart* do kopii zbioru danych zaburzonych poprzez próbkowanie przykładów. Przedmiotem projektu jest zaprojektowanie i implementacja w języku R algorytmu heurystycznego, który z danego zbioru modeli wybierze możliwie mały podzbiór, których uśredniona predykcja dla niezaburzonej kopii zbioru danych będzie charakteryzowała się możliwie małym błędem średniokwadratowym.

**Opis proponowanego algorytmu:**

Zadanie jest dobrze zdefiniowane jako wybór podzbioru ze zbioru, dlatego użyto algorytmu zmodyfikowanego algorytmu genetycznego. Ponieważ dodatkowe warunki zakładają iż podzbiór jest możliwie mały wprowadzono modyfikację do klasycznego algorytmu tworząc algorytm genetyczny z iteracyjnym zwiększaniem maksymalnej liczności podzbiorów. W przypadku zadanego problemu funkcja celu podlega minimalizacji gdyż jest ona zdefiniowana jako błąd średniokwadratowy pomiędzy danymi wejściowymi, a tym co generuje zbiór modeli odpowiadających danemu osobnikowi. Algorytm ma następujący przebieg:

1. Inicjalizacja danych początkowych:

Na podstawie danych użytkownika generowana jest pierwsza populacja jako wszystkie możliwe podzbiory o maksymalnej liczności zadanej przez użytkownika. Ponieważ szukamy minimalnego podzbioru zatem parametr ten powinien również być niewielki (1-4) co spowoduje iż pierwsza populacja nie będzie zbyt liczna ale zapewni ona różnorodną pulę genów początkowych. Po wygenerowaniu znajdowany i zapamiętywany jest najlepszy osobnik.

1. Selekcja

Z bieżącej populacji zostają wybrane osobniki do krzyżowania i mutacji. Wykonywana jest tzw. miękka selekcja czyli każdy osobnik może zostać wybrany. Wybór odbywa się z zwracaniem wiec osobnik może zostać wybrany więcej niż jeden raz

1. Krzyżowanie

Wybrane przez selekcje osobniki są krzyżowane (każdy z sąsiadem z prawej wraz z zapętleniem). Krzyżowanie odbywa się poprzez wykonanie operacji OR na wektorach wartości boolowskich reprezentujących podzbiory. Jeśli liczność podzbioru reprezentowana jest zbyt duża to z podzbioru wyrzucane są nadmiarowe wartości z jednego i z drugiego w proporcjach odpowiadających stosunkom funkcji celu.

1. Mutacja

Osobniki powstałe na skutek krzyżowania poddawane są mutacji. Ze zbioru nowych osobników losowane są te, które zostaną poddane mutacji. Mutacja osobnika odbywa się poprzez wylosowanie genu do zmutowania, po czym jego wartość jest negowana. Na końcu fazy mutacji następuje sprawdzenie czy zmutowany osobnik, nie przekroczył maksymalnej liczności podzbioru. Jeśli tak to następuje wylosowanie genu (spośród mających wartość TRUE) i jego zmiana na FALSE.

1. Dodanie nowych osobników do populacji

Nowe osobniki dodawane są do poprzedniej populacji, a następnie wybierana jest nowa populacja o zadanej maksymalnej liczności. Jeśli liczność jest zbyt duża to eliminowane są osobniki najsłabiej przystosowane.

1. Wyznaczenie nowego najlepszego osobnika

W populacji wyszukiwany jest nowy najlepszy osobnik. Jeśli:

1. Osobnik najlepszy jest inny niż osobnik najlepszy w poprzednim przebiegu to zapamiętaj go, wyzeruj licznik i przejdź do punktu 2.
2. Osobnik najlepszy jest taki sam to zwiększ licznik, jeśli jest odpowiednio duży, przejdź do punktu 7, jeśli nie do punktu 2.
3. Rozszerzenie populacji.

Z populacji wybierane są osobniki do rozszerzenia. Rozszerzenie odbywa się poprzez zaliczenie dodatkowego elementu do podzbioru reprezentowanego przez wybranego osobnika. Rozszerzone osobniki dodawane są do populacji, w której pierwsza selekcja odbywać się może na zbiorze większym niż standardowy aby zwiększyć szanse nowych osobników. Ponadto zwiększany jest maksymalny rozmiar podzbioru i wyliczana różnica pomiędzy osobnikiem najlepszym spośród podzbiorów o rozmiarze maksymalnie *n-1* i najlepszym spośród podzbiorów o rozmiarze *n*. Jeśli różnica pomiędzy nimi jest odpowiednio mała to zwiększany jest licznik, gdy licznik przekroczy zadana wartość następuje koniec algorytmu.

**Rezultaty działania algorytmu:**

W celu przeprowadzenia testów wybrano następujące parametry dla algorytmu:

Liczba modeli – 50

Rodzaj regresji – rpart

Rozmiar populacji – 150

Liczba wybieranych osobników do krzyżowania i mutacji – 50

Rozszerzalność populacji (nExtend) – 75

Algorytm został uruchomiony 100 razy dla połowy zbiorów danych *BostonHousing* oraz *Satellite* bez kolumny z danymi tekstowymi. Następnie modele sprawdzono na drugiej połowie danych tzw. danych testowych. Rezultaty przedstawiono w poniższych tabelach:

|  |  |  |
| --- | --- | --- |
|  | *BostonHousing* | *Satellite* |
| Ilość uruchomień, gdzie model utworzony przez algorytm był lepszy od najlepszego pojedynczego modelu | 88 | 100 |
| Średnia różnica błędu średniokwadratowego pomiędzy najlepszym pojedynczym, a modelem utworzony przez algorytm | 2.3491146 | 7.218573 |
| Ilość uruchomień, gdzie model utworzony przez algorytm był lepszy od modelu będącego uśrednieniem wszystkich modeli | 72 | 100 |
| Średnia różnica predykcji pomiędzy modelem będącym uśrednieniem wszystkich modeli, a utworzony przez algorytm | 0.3125132 | 2.287202 |
| Średni rozmiar modelu utworzonego przez algorytm | 7.9600000 | 7.860000 |

Podstawowym celem tych testów była odpowiedz na pytanie czy zastosowanie opracowanego algorytmu ma sens. Odpowiedz brzmi jednoznacznie, tak ma ponieważ model wygenerowany przez algorytm w blisko 100% (88 i 100 w zależności od danych) przypadków daje lepsze rezultaty niż model pojedynczy i są one zazwyczaj duże lepsze niż dla modelu pojedynczego. Natomiast porównując z modelem pełnym czyli uśrednieniem wszystkich modeli uzyskujemy w większości przypadków nie zbyt duża ale jednak poprawę, a ponadto uzyskujemy znaczną redukcję liczby modeli składowych (od 50 z modelu pełnego do około 8) co pozwala zaoszczędzić na obliczeniach funkcji *predict* – wykonujemy zaledwie 1/5 obliczeń! Natomiast strata związana z nie używaniem modelu pełnego występująca w części przypadków jest relatywnie mała z korzyściami płynącymi z zmniejszenia liczby modeli składowych.

Ponadto wykonano testy, które prezentują tendencje panujące w algorytmie. Wszystkie poniższe testy odbywały się na zbiorze *BostonHousing*, a dla każdych danych algorytm uruchamiany był 20-krotnie.

1. Iteracyjne zwiększanie w przedziale 1-20 parametru, który mówi jak długo (przez ile liczności zbioru) musi się utrzymać najlepszy obiekt, aby zakończyć działanie algorytmu.

|  |  |  |  |
| --- | --- | --- | --- |
| parametr | Najlepszy pojedynczy - nasz | Wszystkie - nasz | Średnia liczność podzbioru |
| 1 | -0.5107951 | -2.6085268 | 2 |
| 2 | 1.8537395 | -0.3130339 | 8 |
| 3 | 1.8708653 | 0.1248744 | 8.45 |
| 4 | 2.1368529 | 0.1923683 | 7.95 |
| 5 | 2.7046993 | 0.5169757 | 7.7 |
| 6 | 2.3298202 | 0.2433755 | 8.8 |
| 7 | 3.0387135 | 0.4680231 | 8.9 |
| 8 | 2.5371775 | 0.3765038 | 9.6 |
| 9 | 2.9848851 | 0.3064405 | 9.4 |
| 10 | 2.4648203 | 0.4018494 | 8.55 |
| 11 | 1.9426875 | 0.001706697 | 8 |
| 12 | 3.383621 | 0.754975 | 10.55 |
| 13 | 2.0155839 | -0.2678041 | 8.1 |
| 14 | 3.2890808 | 0.6427327 | 7.95 |
| 15 | 3.0891697 | 0.3287259 | 8.65 |
| 16 | 2.3814741 | 0.7184193 | 8.45 |
| 17 | 3.1375605 | 0.6075746 | 8.75 |
| 18 | 3.0221907 | 0.3991161 | 8.1 |
| 19 | 2.4828369 | 0.6326475 | 9.3 |
| 20 | 2.563598 | 0.1571548 | 9.85 |

Zależność różnicy błędów średniokwadratowych dla najlepszego z wszystkich modeli i naszego

Zależność różnicy błędów średniokwadratowych dla modelu będącego uśrednieniem wszystkich modeli i naszego

Zależność wynikowego rozmiaru modelu od wartości parametru wejściowego

Test przeprowadzony w tym podpunkcie pokazuje prawidłowość algorytmu tj. właściwy sposób rozszerzania populacji. Parametr zwiększany w tym podpunkcie pośrednio kontroluje maksymalny rozmiar podzbioru. Ponieważ algorytm bazuje na założeniu iż dodanie nowego modelu do podzbioru powinno poprawić jego ranking to parametr ten informuje nas o ile większe podzbiory od tego, który jest aktualnie wybrany jako najlepszy jesteśmy w stanie uwzględnić w poszukiwaniu optymalnego. Parametr ten odpowiada jest liczbowym wyrażeniem wagi rozmiaru zbioru i błędu średniokwadratowego.

Zaprezentowane wykresy świadczą o zgodności implementacji algorytmu z semantyka tego parametru i pokazują nam pewne istotne tendencje. Jak łatwo zauważyć, dla małych wartości średni rozmiar podzbioru jest stosunkowo mały np. dla *n=1* rozmiar wynosi 2, jednak błąd średniokwadratowy dla tak małego zbioru bardzo często okazuje się być większy dla modelu wybranego przez nasz algorytm niż dla uśrednienia wszystkich modeli. Odpowiada to sytuacji, gdy zależy nam na znalezieniu minimalnego podzbioru, mogąc sobie pozwolić na większy błąd. Natomiast gdy ten parametr ma wartość znacznie większą np. 10, to algorytm może przeszukiwać podzbiory o liczności o 10 większej od bieżącego najlepszego. Odpowiada to sytuacji gdy zależy nam aby wybrany podzbiór był niewielki, ale znacznie bardziej zależy nam jednak na minimalizacji błędu średniokwadratowego.

Ponadto z przedstawionych wykresów można zauważyć bardzo ciekawe zjawisko oscylacji wokół pewnej liczności zbioru. Oznacza to iż w przeważającej większości przypadków optymalny podzbiór modeli jest mniejszy niż zbiór złożony z wszystkich modeli.

1. Drugim ciekawym testem ukazującym własności prezentowanego algorytmu jest iteracyjne zwiększanie populacji, odpowiednie dane przedstawiono w tabeli:

|  |  |  |  |
| --- | --- | --- | --- |
| parametr | najlepszy pojedynczy - nasz | Wszystkie - nasz | Średnia liczność podzbioru |
| 10 | 2.36558049 | 0.02694862 | 10.45 |
| 20 | 1.6103754 | 0.2830431 | 10.45 |
| 30 | 3.2709751 | 0.4499502 | 8.55 |
| 40 | 2.64013329 | 0.02015605 | 8.55 |
| 50 | 2.1548992 | 0.3187503 | 9.05 |
| 60 | 3.5082213 | 0.7328796 | 8.3 |
| 70 | 2.3385462 | 0.2332656 | 7.8 |
| 80 | 2.7688408 | 0.3783047 | 8 |
| 90 | 2.0523154 | 0.1504085 | 8.95 |
| 100 | 3.0278252 | 0.5552842 | 8.85 |
| 110 | 2.6574932 | 0.5026854 | 9.25 |
| 120 | 2.289582 | 1.010476 | 8.4 |
| 130 | 2.977185 | 0.3875311 | 8.3 |
| 140 | 2.6157505 | 0.2734939 | 8.45 |
| 150 | 2.11831555 | 0.00209384 | 7.65 |

Zależność liczności podzbioru modeli od rozmiaru populacji używanej w algorytmie

Prezentowany wykres jest bardzo ciekawy ponieważ ukazuje nam iż w prezentowanym algorytmie, jak i w każdym algorytmie genetycznym rozmiar populacji odgrywa istotną rolę. W przypadku standardowego algorytmu genetycznego zbyt mała populacja powoduje w przypadku ograniczonej liczby iteracji, znaczne *'ulokalnienie'* przeszukiwań, a więc w znacznym stopniu uzależnia algorytm od pierwszej generacji czyli od początkowej puli genów, a przez to algorytm staje się dużo bardziej losowy.

Ponieważ w przypadku prezentowanego algorytmu, algorytm genetyczny wykonywany jest iteracyjnie to sprawia, że przy małej populacji wybieramy podzbiory o większym rozmiarze, ponieważ nie jesteśmy w stanie przeszukać podzbiorów o mniejszych rozmiarach gdyż pula genów jest zbyt *uboga*. Prowadzi to do degradacji algorytmu do algorytmu przeszukiwania losowego podobszaru przestrzeni przeszukiwań. W łatwy sposób można zauważyć tendencje spadkową na omawianym wykresie. Uzasadnienie takiej tendencji jest oczywiste: Jeśli zwiększamy populację to oznacza, że przeszukujemy więcej osobników z podzbiorów o liczności co najwyżej *n*, dzięki czemu przy dostatecznie dużym rozmiarze populacji możemy uznać selekcje za globalną, co z kolei skutkuje znalezieniem rozwiązania zgodnego z założeniami.