Selekcja cech

Wstęp

Podczas budowy modelu służącego predykcji w systemach inteligencji obliczeniowej często spotykamy się z problemem, w którym nie znamy znaczenia ani istotności poszczególnych zmiennych składających się na zbiór danych którymi dysponujemy. Innym częstym przypadkiem jest tzw. występowanie cech(zmiennych) redundantnych. Zmienne takie w pewnym uproszczeniu mogą być interpretowane jako cechy które niosą tą samą informację co inne zmienne. Mogą to być więc zmienne wyrażone w różnych jednostkach np. wzrost wyrażony w centymetrach, a druga zmienna wyrażona w metrach, jeszcze innym przykładem tego problemu może być np. w przypadku klasyfikacji płci zmienna opisująca rozmiar buta oraz druga opisująca rozmiar stopy wyznaczony w „cm". Usunięcie takich cech nie wpływa negatywnie na sam proces uczenia, a jednocześnie redukuje przestrzeń zmiennych, co dla niektórych algorytmów może być szczególnie istotne. Wiąże się to z tzw. *przekleństwem wymiarowości*. Problem ten występuje np. w większości modeli statystycznych bazujących na wielowymiarowej estymacji prawdopodobieństwa, gdyż wówczas złożoność obliczeniowa i pamięciowa jest rzędu m^n – gdzie m to liczba wartości/przedziałów zmiennych, natomiast *n* to liczba zmiennych. Dla przykładu złożoność dla 4 przedziałów zmiennej dyskretnej i 10 cech = 1.048.576, natomiast już dla 12 cech złożoność ta jest równa = 16.777.216

Reasumując za „dobry" zbiór cech to taki, który zawiera kombinację zmiennych pozwalającą na możliwie najlepszy błąd klasyfikacji.

Metody selekcji cech można podzielić na:

- Ze względu na charakter problemu
  - Nadzorowane
  - Nienadzorowame
- Ze względu na relację z innymi algorytmami nadrzędnymi
  - Filtry
  - Wrappery (opakowane)
  - Frapery – kombinacja filtrów i Wrapperów
  - Metody wbudowane

W matlabie

Celem zajęć będzie implementacja Wrappera oraz frappera z trzema różnymi algorytmami przeszukiwania: w przód, w tył oraz metoda rankingowa, gdzie jako ranking wykorzystuje się współczynnik korelacji

Ćw 1

Implementacja algorytmu selekcji w przód wg. Algorytmu:

```
Require: f
  f′ ← ∅
  n ← numof(f)
  repeat
    chk ← true
    for i = 1...n do
      tacc ←oceń(f′ ∪ f_i)
      if tacc > acc then
        acc ← tacc
        j ← i
        chk ← false
      end if
    end for
    if not chk then
      f′ ← f′ ∪ f_i
      f ← f/f_i
    end if
    n ← numof(f)
  until chk ∨ n = 0
  return f′
```

Kod w matlabie uzupełnij kod:

```
%--------
chk = true;

%1) wczytaj dane
%2) podział danych na trening (dataTrain) oraz test (dataTest)

idx = [];
subDataTrain.X = [];
subDataTest.X  = [];
dokladnosc = -inf;
tmpDataTrain.Y = dataTrain.Y;
tmpDataTest.Y = dataTest.Y;
subDataTrain.Y = dataTrain.Y;
subDataTest.Y = dataTest.Y;

while (chk && (size(dataTrain.X,2)>1))
    chk = false;
    for i = 1 : size(dataTrain.X,2)
        % - Wybranie podzbioru cech
        tmpDataTrain.X = [subDataTrain.X dataTrain.X(:,i)];
        tmpDataTest.X  = [subDataTest.X dataTest.X(:,i)];
        % - Ocena jakości cech
        %3) uczenie modelu liniowego na danych tmpDataTrain
        %4) testowanie modelu liniowego na danych tmpDataTest
        % - Zapis największej dokładności
        if tmp_dokladnosc > dokladnosc
            dokladnosc = tmp_dokladnosc;
            best_i = i;
            chk = true;
        end;
    end;
    % - Wybranie podzbioru cech i wyjście jeśli nie ma poprawy
    if (~chk), return; end;
    idx = [idx best_i];
    % - Rozszerzenie podzbioru
    subDataTrain.X = [subDataTrain.X dataTrain.X(:,best_i)];
    subDataTest.X  = [subDataTest.X dataTest.X(:,best_i)];
```

```
    % - Usunięcie dodanych cech z podzbioru
    dataTrain.X(:,best_i) = [];
    dataTest.X(:,best_i)  = [];
end;
```

Ćw 2

Zastanów się i zmodyfikuj powyższą funkcję tak aby realizowała selekcję w tył wg. Zależności:

Require: f
  $\mathrm{f}' \leftarrow f$
  $n \leftarrow numof(\mathrm{f})$
  repeat
    $chk \leftarrow$ **true**
    for $i = 1 \ldots n$ **do**
      $tacc \leftarrow$oceń(f'/f_i)
      if $tacc > acc$ **then**
        $acc \leftarrow tacc$
        $j \leftarrow i$
        $chk \leftarrow$ **false**
      end if
    end for
    if not chk **then**
      $\mathrm{f}' \leftarrow \mathrm{f}'/f_i$
      $\mathrm{f} \leftarrow \mathrm{f}/f_i$
    end if
    $n \leftarrow numof(\mathrm{f})$
  until $chk \vee n = 0$
  return f′

Ćw 3

Zaimplementuj algorytm selekcji w bazującej na rankingu wg zależności:

Require: f {wejściowy zbiór cech}
Require: $J(\cdot)$ {funkcja rankingowa}
  for $i = 1 \ldots n$ **do**
    $a_i \leftarrow J(f_i)$ {oceń i-tą cechę}
  end for
  $\mathrm{f} \leftarrow$ sort(f, a) {sortuj cechy wg. istotności}
  $acc \leftarrow 0$
  $\mathrm{f}^a \leftarrow \emptyset$
  for $j = 1 \ldots n$ **do**
    $\mathrm{f}^a = \mathrm{f}^a \cup f_i$ {dodaj do podzbioru f^a nową cechę}
    $tacc \leftarrow$oceń(f^a) {oceń podzbiór cech f^a }
    if $tacc > acc$ **then**
      $acc \leftarrow tacc$ {jeżeli nowy podzbiór jest lepszy od poprzedniego }
      $\mathrm{f}' \leftarrow \mathrm{f}^a$ {zapamiętaj wynik aktualny podzbiór}
    end if
  end for
  return f′

W matlabie:

```
clc;
clear;

%rezerwacja pamięci
ranking = zeros(1,size(dataTrain.X,2));
dokladnosc = -inf;
```

```
for i=1:size(dataTrain.X,2)
    tmp = abs(corr(dataTrain.X(:,i),dataTrain.Y));
    ranking(i) = tmp;
end;

[ranking,idx] = sort(ranking,'descend');

for i=1:size(dataTrain.X,2)
    tmpDataTrain.X = dataTrain.X(:,idx(1:i));
    tmpDataTest.X = dataTest.X(:,idx(1:i));
    %naucz model liniowy na danych tmpDataTrain
    %przetestuj model liniowy na danych tmpDataTest
    %wyznacz tmp_dokladność i wybierz przypadek w którym dokładność jest
max
    if tmp_dokladnosc > dokladnosc
        best_i = i
        dokladnosc = tmp_dokladnosc
    end;
end;
```

Do zrobienia

Zaimplementuj wszystkie powyższe algorytmy

W sprawozdaniu umieść kod selekcji w tył

Dokonaj analizy algorytmów dla danych WBC, pima, ionosphere, dla każdego algorytmu wyznacz maksymalną dokładność, oraz najlepszy podzbiór cech. Wypisz najlepsze rozwiązania (listę najlepszych cech) wraz z uzyskanymi dokładnościami. Wyniki zapisz w tabelce.

Czym charakteryzują się poszczególne algorytmy (który zwykle wybiera najmniej zmiennych, a który zwykle najwięcej, który z algorytmów prowadzi zwykle do najlepszej dokładności).