Inteligencja obliczeniowa i jej zastosowania

Laboratorum cz. IV, nr 3-4

Autorzy:
Joanna Piątek, nr indeksu: 199966
Agnieszka Wątrucka, nr indeksu: 200016
Grupa: Środa, 15:15

Data oddania: 12.06.2016

Prowadzący: prof. dr hab. inż. Rafał Zdunek

1 Zadanie 2

W zadaniu nr 2 celem było wyznaczenie estymowanych faktorów na podstawie syntetycznie wygenerowanych obserwacji. Należało je wykonać za pomocą wybranego algorytmu NTF.

1.1 Implementacja

Na poniższym listingu został przedstawiony początkowy fragment implementacji rozwiązania zadania. Pierwszą czynnością do wykonania jest przypisanie zadanych danych do zmiennych oraz wygenerowanie faktorów $U^{(1)}$, $U^{(2)}$ i $U^{(3)}$ (odpowiednio zmienne: *U1w*, *U2w* i *U3w*). Następnie, na podstawie tychże faktorów oraz macierzy jednostkowej I o wymiarach *JxJxJ*, generowane są trójwymiarowe, syntetyczne obserwacje Y. W tym celu została wykorzystana funkcja *ntimes*, mnożąca ze sobą kolejno po dwie podane macierze. Y to tensor o wymiarach 10x20x30 - jest to złożenie dłuższego wymiaru każdego ze stworzonych faktorów.

```
% Dane
I1 = 10;
I2 = 20;
I3 = 30;
J = 5;

% Inicjalizacja faktorow
U1w = max(0, randn(I1, J));
U2w = max(0, randn(I2, J));
U3w = max(0, randn(I3, J));

% Syntetyczne obserwacje Y
I = zeros(J,J,J);
for i = 1 : J
        I(i,i,i) = 1;
end

Y = ntimes(ntimes(ntimes(I, U1w, 1, 2), U2w, 1, 2), U3w, 1, 2);

%% Algorytm ALS
% Inicjalizacja faktorow
U1 = randn(I1, J);
U2 = randn(I2, J);
U3 = randn(I3, J);

% Matrycyzacja Y wzgledem poszczegolnych modow
Y1 = reshape(permute(Y,[1,2,3]),size(Y,1),size(Y,2)*size(Y,3));
Y2 = reshape(permute(Y,[2,1,3]),size(Y,2),size(Y,1)*size(Y,3));
Y3 = reshape(permute(Y,[3,1,2]),size(Y,3),size(Y,1)*size(Y,2));

% Obliczenie wersji 2-D tensora Y
Y_2d = reshape(Y,size(Y,1),size(Y,2)*size(Y,3));
```

Kolejnym krokiem jest dekompozycja CP z wykorzystaniem algorytmu ALS. Inicjalizowane są estymowane faktory (zmienne *U1*, *U2* i *U3*). Algorytm wymaga matrycyzacji tensora *Y* względem poszczególnych modów, co udało się uzyskać używając funkcji reshape i permute. Oznacza to, że macierze wyjściowe *Y1*, *Y2* i *Y3* są dwuwymiarowe - pierwszy z wymiarów jest równy kolejnemu wymiarowi oryginalnego *Y*, a drugi - iloczynem pozostałych wymiarów. Ostatnia czynność to obliczenie dwuwymiarowej wersji tensora *Y*, potrzebnej do późniejszego wyznaczenia błędu residualnego (funkcja *norm* przyjmuje jako argumenty jedynie dwuwymiarowe macierze).

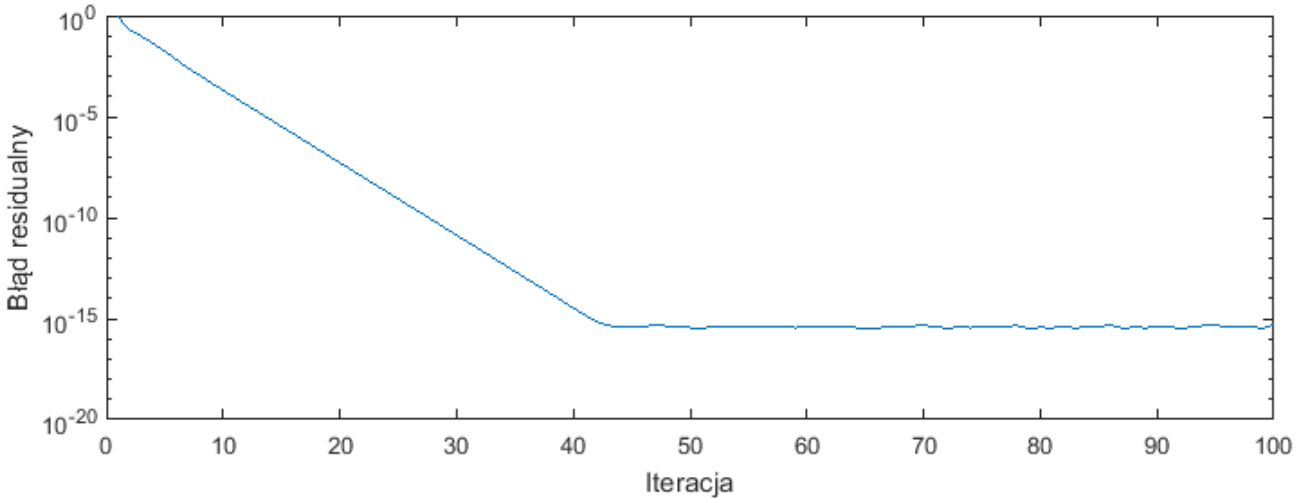
```
MaxIter = 200;
for k = 1: MaxIter

        % Update naprzemienny faktorow i normalizacja
        A1 = max(0,(kr(U3,U2))');
        U1 = max(0, (Y1*A1')/(A1*A1'));
        U1 = U1*(diag(1./sum(U1, 1)));

        A2 = max(0,(kr(U3, U1))');
        U2 = max(0, (Y2*A2')/(A2*A2'));
        U2 = U2*(diag(1./sum(U2, 1)));

        A3 = max(0,(kr(U2,U1))');
        U3 = max(0, (Y3*A3')/(A3*A3'));

        % Wyliczanie tensora Yk na dla k-tej iteracji
        % na podstawie aktualnych U1, U2 i U3
        Yk = ntimes(ntimes(ntimes(I,U1,1,2),U2,1,2),U3,1,2);
        Yk_2d = reshape(Yk,size(Yk,1),size(Yk,2)*size(Yk,3));

        % Blad residualny
        res(k) = norm(Y_2d - Yk_2d, 'fro')/norm(Y_2d, 'fro');
end

% Rysowanie wykresu bledu
semilogy(res)

% Jakosc estymacji
mse = immse(Y, Yk);
```

Na listingu widocznym powyżej przedstawiona jest główna pętla działania programu. W każdym przebiegu pętli wyznaczane są wartości faktorów *U1*, *U2* i *U3*. Wykonywana jest także ich normalizacja. Obliczana jest także wartość *Yk*, czyli wartości tensora *Y* dla faktorów wyznaczonych w danej iteracji. Następnie z *Yk* powstaje macierz dwuwymiarowa *Yk_2d*. Na podstawie *Yk_2d* i *Y_2d* wyliczany jest błąd residualny dla aktualnie wyznaczonej wartości *Yk*.

Podsumowanie wyników obliczeń polega na narysowaniu wykreu błędu residualnego w kolejnych iteracjach oraz wyliczenie błędu średniokwadratowego.

1.2 Wyniki

Na wykresie 1 można zaobserwować zmiany wartości błędu residualnego w kolejnych iteracjach algorytmu. W omawianym przypadku stabilizuje się ona między 40 a 50 przebiegiem pętli. Dalsze iteracje nie wpływają znacząco na wynik obliczeń.



Rysunek 1: Unormowany błąd residualny w danej iteracji dla funkcji iteracji naprzemiennych

Powyższe obserwacje potwierdza także otrzymana z obliczeń wartość błędu średniokwadratowego. Po 50 iteracjach wynosi on 2.3831e−31, natomiast po 100 przebiegach pętli - 2.1673e−31. Wynik został więc ustabilizowany.

Algorytm HALS jest najbardziej złożony z trzech wykorzystanych w zadaniu algorytmów. Dlatego w jego głównej pętli są tworzone zmienne pomocnicze, które swoimi nazwami przypominają oznaczenia w formule matematycznej algorytmu. Wygląda ona w następujący sposób:

$$\boldsymbol{a}_j \leftarrow \left[\boldsymbol{a}_j + \frac{[\boldsymbol{Y}\boldsymbol{X}^T]_{*j} - \boldsymbol{A}[\boldsymbol{X}\boldsymbol{X}^T]_{*j}}{[\boldsymbol{X}\boldsymbol{X}^T]_{jj}}\right]_+, \quad \underline{\boldsymbol{x}}_j \leftarrow \left[\underline{\boldsymbol{x}}_j + \frac{[\boldsymbol{A}^T\boldsymbol{Y}]_{j,*} - [\boldsymbol{A}^T\boldsymbol{A}]_{j,*}\boldsymbol{X}}{[\boldsymbol{A}^T\boldsymbol{A}]_{jj}}\right]_+.$$

Jak widać na poniższym listingu, implementacja HALS zawiera dwie pętle. Dzieje się tak dlatego, że podczas jednego przebiegu pętli wewnętrznej zostaje zmieniona tylko jedna kolumna w macierzy A i jeden wiersz w macierzy X. Żeby całe macierze przeszły ten proces, pętla od 1 do J musi wykonać się w całości. Z tego powodu normalizacja zostaje wykonana dopiero po pełnym przebiegu pętli wewnętrznej.

```
%% Algorytm HALS
%% Inicjalizacja
A_hals = rand(size(Y,1), J);
X_hals = rand(J, size(Y,2));

%% Update
MaxIter = 400;
res_hals = zeros(1,MaxIter);

for k = 1: MaxIter
        % Inicjalizacja zmiennych pomocniczych
        XXT = X_hals*X_hals';
        YXT = Y*X_hals';
        ATA = A_hals'*A_hals;
        ATY = A_hals'*Y;

        for j = 1 : J
        % Update A
        nextA = A_hals(:,j)
                + (YXT(:,j) - A_hals*XXT(:,j))./max(eps,XXT(j,j));
        A_hals(:, j) = max(0, nextA);

        % Update X
        nextX = X_hals(j,:)
                + (ATY(j,:) - ATA(j,:)*X_hals)./max(eps,ATA(j,j));
        X_hals(j, :) = max(0, nextX);
        end

        % Normalizacja
        A_hals = A_hals*(diag(1./max(eps, sum(A_hals, 1))));

        % Blad residualny
        res_hals(k) = norm(Y - A_hals*X_hals, 'fro')/norm(Y, 'fro');
end
```