Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za računarstvo i informatiku

Zadaća №
Naziv zadaće

Predmet

Ime i prezime: /
Broj indexa: /
Grupa: /
Datum: /

Teorijski uvod

Algoritam koji smo odabrali za ovu temu je RBF algoritam interpolacije(Radial Basis Function Interpolation). Ovaj algoritam se zasniva na ideji da svaka poznata tačka na osnovu koje vršimo interpolaciju jednako utiče na vrijednost funkcije u svim tačkama koje su jednako udaljene od te tačke na način opisan nekom pretpostavljenom funkcijom. Formalno rečeno

$$f(\mathbf{x}) \approx \sum_{i=1}^{N} w_i \phi(|\mathbf{x} - \mathbf{x}_i|)$$

gdje u ovoj formuli f predstavlja funkciju koju želimo interpolirati, N označava broj poznatih tačaka na osnovu kojih vršimo interpolaciju, dok $\mathbf{x}_i$ su upravo te tačke. Označimo sa y_i vrijednosti $f(\mathbf{x}_i)$. Težinski koeficijenti w_i predstavljaju koliko intenzivno na vrijednost funkcije u proizvoljnoj tački utiče udaljenost od tačke $\mathbf{x}_i$, dok ϕ predstavlja pretpostavljenu radijalnu funkciju koja određuje na koji način udaljenost djeluje na tu istu vrijednost.

Za funkciju ϕ imamo mnogo opcija mada se najčešće koristi takozvana multikvadratna funkcija koja glasi $\phi(r) = \sqrt{r^2 + r_0^2}$, gdje r_0 predstavlja takozvani faktor skaliranja koji se tipično uzima da je veći od prosječnih udaljenosti, a manje od najvećih udaljenosti tačaka $\mathbf{x}_i$. Još neke funkcije koje se mogu koristi su inverzna multikvadratna: $\phi(r) = \frac{1}{\sqrt{(r^2+r_0^2)}}$, gausovska: $\phi(r) = e^{-\frac{r^2}{2r_0^2}}$, thin-plate spline: $\phi(r) = r^2 \log(\frac{r}{r_0})$.

Prirodno se još postavlja pitanje kako odrediti težinske koeficijente nakon što sto pretpostavili radijalnu funkciju. Uvrštavanjem vrijednosti $\mathbf{x}_n$ u gore navedenu formulu znak $\approx$ možemo da zamijenimo znakom jednakosti jer se radi o interpolaciji, te nam to daje

$$f(\mathbf{x}_n) = \sum_{i=1}^{N} w_i \phi(|\mathbf{x}_n - \mathbf{x}_i|) \qquad , n = \overline{1, N}$$

a ovo je ništa drugo nego sistem od N linearnih jednačina sa N nepoznatih(w_i za $i = \overline{1, N}$). Rješavanjem ovog sistema metodama za rješavanje sistema linearnih jednačina(na primjer Gaussova eliminacija) dobijamo tražene težinske koeficiente. Interpolacija se onda vrši po prvoj navedenoj formuli

Implementacija algoritma

Opis implementacije

Algoritam je pisan u Scilabu i sastoji se od sljedečih funkcija. Za funkciju ϕ smo uzeli multikvadratnu funkciju.

- radial_distance:
  Služi za računanje udaljenosti dvije tačke n-dimenzijaonalnom prostoru koristeći standardnu formulu $\sqrt{\sum_{i=1}^{N}(a_i - b_i)^2}$.

- multiquadratic:
  Računa vrijednost multikvadratne funkcije za date parametre r i r_0.

- radial_matrix:
  Stvara trivijalnu matricu R tako da vrijedi da je $R_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|$ pozivajući radial_distance.

- multiquadratic_matrix:
  Prima radijalnu matricu i primjenjuje multikvadratnu funkciju na svaki član, sa vrijednošću r_0 postavljenom na geometrijsku sredinu srednjeg i najvećeg elementa radijalne matrice, time ustvari računajući vrijednosti koeficijenata uz w_i sistema koji se treba riješiti.

- point_distance_vector:
  Prima tačku/e u kojoj/im želimo vršiti interpolaciju, kao i tečke na osnovu kojih smo vršili interpolaciju i vrijednost koeficienta r_0 korištenog da se izračuna multikvadratna matricai vraća horizontalni vektor, odnosno matricu u slučaju da smo vršili interpolaciju u više tačaka, koji predstavlja izračunate sve odgovarajuće vrijednosti funkcije ϕ.

- RBF_weights:
  Prima tačke u kojima vršimo interpolaciju, i vrijednosti funkcije u tim tačkama, te riješava sistem preko kojeg određujemo vrijednosti težinskih koeficijeneata koristeći gore nevedene pomoćne funkcije da formira sistem.

- RBF_evaluate:
  Prima tačku/e u kojim se vrši interpolacija, tačke na osnovu kojih interpoliramo, vektor težinskih koeficijenata kao i r_0 koje smo koristili da ih izračunamo, i vraća interpoliranu vrijednost, odnosno vektor ako aproksimiramo u više tačaka.

- RBF_one_time_evaluation:
  Ova funkcija služi ako hoćemo samo jednom da interpoliramo, te samo joj prosljeđujemo tačke na osnovu kojih interpoliramo, vrijednost funkcije u njima, i tačke u kojima interpoliramo, te ona poziva RBF_weights i RBF_evaluate i vraća tražene vrijednosti

- Test:
  Prima funkciju, broj argumenata, broj tačaka na osnovu kojih vršimo interpolaciju i broj tačaka u kojima interpoliramo. Stvara matrice odredjenih veličina: X_train i X_test i popunjava ig+h sa odgovarajučim brojem nasumičnih vrijednosti izmedju –500 i 500 i računa vrijednosti funkcije u njima, nakon toga vrši se interpolaciju u X_test na osnovu X_train i vračaju se očekivanje(stvarne) i interpolirane vrijednosti.

Source code

```
function d = radial_distance(a,b)
    d = sqrt(sum((a-b).^2));
endfunction
```

```
function fi = multiquadratic(r,r0)
    fi = sqrt(r.^2+r0.^2);
endfunction
```

```
function R = radial_matrix(X)
    [nrows, ncols] = size(X);
    R = zeros(nrows, nrows);
    for i = 1:nrows
        for j = (i+1):nrows
            R(i,j) = radial_distance(X(i,:),X(j,:));
            R(j,i) = R(i,j);
        end
    end
endfunction
```

```
function [M, r0] = multiquadratic_matrix(X)
    r0 = sqrt(mean(X) * max(X));
    M = multiquadratic(X, r0);
endfunction
```

```
function W1 = point_distance_vector(A, X, r0)
    [xrows,xcols] = size(X);
    [arows,acols] = size(A);
    W1 = zeros(arows, xrows);
    for i = 1:arows
        for j = 1:xrows
            W1(i,j) = radial_distance(A(i,:), X(j,:));
        end
    end
    W1 = multiquadratic(W1, r0);
endfunction
```

```
function [W, r0] = RBF_weights(X,Y)
    [W1, r0] = multiquadratic_matrix(radial_matrix(X));
    W = linsolve(W1,-Y);
endfunction
```

```
function Est = RBF_evaluate(A, X, W, r0)
    Est = point_distance_vector(A,X,r0) * W;
endfunction
```

```
function Est = RBF_one_time_evaluation(X, Y, A)
    [W, r0] = RBF_weights(X,Y);
    Est = RBF_evaluate(A, X, W, r0);
endfunction
```

```
function [y_test, y_pred] = Test(f, nargs, train_size,
   test_size)
    X_train = rand(train_size, nargs) * 1000 - 500;
    X_test = rand(test_size, nargs) * 1000 - 500;
    y_train = zeros(train_size, 1);
    y_test = zeros(test_size, 1);
    y_pred = zeros(test_size, 1);
    for i=1:train_size
        y_train(i) = f(X_train(i,:));
    end
    for i=1:test_size
        y_test(i) = f(X_test(i,:));
```

```
    end
    y_pred = RBF_one_time_evaluation(X_train, y_train, X_test)
   ;
endfunction
```

Testiranje

Obavljene su 3 faze testiranja:

Prva faza koristi jednostavna funkciju $f(a, b, c) = a + b + c$ sa 10 poznatih tačaka, i računanjem u 10 nepoznatih tačaka. Srednja apsolutna vrijednost greške je 27.883985, dok je standardna devijacija 27.177035. Ako isti postupak ponovimo za 100 tačaka umijesto 10(i poznatih i traženih) Srednja vrijednost i standardna devijacija glase redom 0.7763442 i 1.0250536.

Druga faza koristi funkciju glasi $f(a, b, c) = a * b + c$ sa 100 poznatih i 1000 traženih tačaka. Srednja vrijednost apsolutne greške i njena standardna devijacija glase 4138.8169 i 3997.0845. Iako su oe vrijednosti velike, apsolutna vrijednost funkcije u prosjeku je oko $6*10^5$ te ovo nije ni 10% relativna greška.

Treča faza koristi funkciju $f(a, b, c, d, e) = a + \sin(b) + \ln(c) + 2107$ sa 100 poznatih i 100 traženih i onda daje srednju vrijednost i standardnu devijaciju apsolutne greške da glase redom 6.8748745 i 5.1381267.