BLM 103- Algoritmalar ve Programlama I
2019-2020 Güz Dönemi

Yapay Zeka ile Kanser Tespiti

Proje Teslim Raporu
6 Ocak 2020

EKREM BÜYÜKKAYA,MUSTAFA BERK TAŞKIN

PROJENIN AMACI

- Bu proje yapay zeka uygulamalarında kullanılan bir algoritma olan kNN (k Nearest Neighborhood, En Yakın k Komşu) algoritması gerçeklenecektir. Gerçeklenen algoritma ile, UC Irvine Üniversitesi'nin sağlamış olduğu Göğüs Kanseri verileri işlenecektir. Hasta olup olmadığı belli olmayan bir kişinin verileri sisteme beslenerek, hastalık tahmini yapılacaktır.

KULLANILAN ARAÇLAR

Microsoft Visual Studio

TASARIM

- **Çok boyutlu öklit uzaklığı hesaplamalarında zorlandık. Mevcut tasarımdaki en büyük eksik test ve eğitim veri setlerinin satır sayısını sabit bir değişken olarak tutmamız. Her tahmin için eğitim veri seti tümüyle taranıp uzaklıklardan ve sonuçlardan oluşan bir diziye çevriliyor. Daha sonra uzaklıklar ve sonuçlardan oluşan bu yeni "komşu" seti sıralanıyor ve en yakın komşular elde ediliyor. Bu sıralanmış dizinin ilk N değerinde kanserli mı yoksa kanserli olmayan mı daha fazla tespit ettikten sonra bu sonucu döndürüyoruz.**

```c
#include <stdio.h>
#include "veri.h"
#include "tahmin.h"

void girisMesajiniGoster() {
    printf("Yapay Zeka ile Kanser Tespiti\n");
    printf("k Nearest Neighborhood\n");
    printf("----------------------------\n");
}

int kullaniciTercihiniAl() {
    int secim;

    printf("Tercihinizi girin:\n");
    printf("1) Test verisetini hesapla\n");
    printf("2) Parametre girişi ile tahminde bulun\n");

    // Kullanıcının seçimini al
    scanf_s("%d", &secim);

    return secim;
}

char* okunabilirSonuc(int sonuc) {
    return sonuc == 2 ? "Hasta değil" : "Hasta";
}

void testSonuclari() {
    printf("Test sonuçları:\n");
```

```c
    printf("--------------\n");
    struct TestSeti testSeti = testSetiniYukle();
    int basariliTahmin = 0;
    int falsePositive = 0;
    int falseNegative = 0;

    for (int i = 0; i < TEST_VERI_SAYI; i++) {
        int asilSonuc = testSeti.y[i];
        int tahminEdilenSonuc = tahmin(testSeti.X[i]);

        printf("Asıl sonuç: %-18s|Tahmin: %-20s\n",
            okunabilirSonuc(asilSonuc),
            okunabilirSonuc(tahminEdilenSonuc));

        if (asilSonuc == tahminEdilenSonuc) {
            basariliTahmin++;
        }
        else if (asilSonuc == 2 && tahminEdilenSonuc == 4) {
            falsePositive++;
        }
        else if (asilSonuc == 4 && tahminEdilenSonuc == 2) {
            falseNegative++;
        }
    }

    printf("\n\n");
    printf("Özet Sonuçlar\n");
    printf("---------------------\n");
    printf("Test sayısı:    %d\n", TEST_VERI_SAYI);
    printf("Doğru tahmin:   %d\n", basariliTahmin);
```

```
60      printf("Yanlış tahmin:  %d\n", TEST_VERI_SAYI - basariliTahmin);
61      printf("Accuracy:       %.2f%%\n", ((float)basariliTahmin / (float)TEST_VERI_SAYI) * 100);
62      printf("False positive  %d\n", falsePositive);
63      printf("False negative  %d\n", falseNegative);
64  }
65
66  void kullaniciGirisiyleTahmin() {
67      int clumpThickness;
68      int uniformityOfCellSize;
69      int uniformityOfCellShape;
70      int marginalAdhesion;
71      int singleEpithelialCellSize;
72      int bareNuclei;
73      int blandChromatin;
74      int normalNucleoli;
75      int mitoses;
76
77      printf("Clump thickness:\n");
78      scanf_s("%d", &clumpThickness);
79      printf("Uniformity of Cell Size:\n");
80      scanf_s("%d", &uniformityOfCellSize);
81      printf("Uniformity of Cell Shape:\n");
82      scanf_s("%d", &uniformityOfCellShape);
83      printf("Marginal Adhesion:\n");
84      scanf_s("%d", &marginalAdhesion);
85      printf("Single Epithelial Cell Size:\n");
86      scanf_s("%d", &singleEpithelialCellSize);
87      printf("Bare Nuclei:\n");
88      scanf_s("%d", &bareNuclei);
89      printf("Bland Chromatin:\n");
```

```c
    scanf_s("%d", &blandChromatin);
    printf("Normal Nucleoli:\n");
    scanf_s("%d", &normalNucleoli);
    printf("Mitoses:\n");
    scanf_s("%d", &mitoses);

    int X[PARAMETRE_SAYI] = { clumpThickness, uniformityOfCellSize, uniformityOfCellShape, marginalAdhesion, singleEpithelialCellSize, bareNuclei, blandChromatin, normalNucleoli, mitoses };

    printf("Sonuç: %s\n", okunabilirSonuc(tahmin(X)));
}

int main() {
    int secim;

    girisMesajiniGoster();

    while (1) {
        // Kullanıcının tercihini sor
        secim = kullaniciTercihiniAl();

        // Geçerli bir giriş olmadıkça karşılama mesajı göster
        if (secim == 1) {
            testSonuclari();
            break;
        }
        else if (secim == 2) {
            kullaniciGirisiyleTahmin();
            break;
        }
    }

    return 0;
}
```

```c
#include "tahmin.h"
#include "veri.h"

float oklid_uzaklik(int a[], int b[]) {
    int total = 0, diff;
    for (int i = 0; i < PARAMETRE_SAYI; i++) {
        diff = b[i] - a[i];
        total += diff * diff;
    }

    return (float)sqrt(total);
}

int karsilastir(const void* komsu1, const void* komsu2) {
    struct Komsu* k1 = (struct Komsu*)komsu1;
    struct Komsu* k2 = (struct Komsu*)komsu2;

    return (k1->oklid_uzaklik > k2->oklid_uzaklik) - (k1->oklid_uzaklik < k2->oklid_uzaklik);
}

int tahmin(int X_test[PARAMETRE_SAYI]) {
    struct EgitimSeti set = egitimSetiniYukle();
    struct Komsu komsular[EGITIM_VERI_SAYI];

    // Uzaklıkları hesapla
    for (int i = 0; i < EGITIM_VERI_SAYI; i++) {
        komsular[i].oklid_uzaklik = oklid_uzaklik(X_test, set.X[i]);
        komsular[i].sonuc = set.y[i];
    }
```

```
30
31      // Komşuları uzaklığa göre sırala (küçükten büyüğe)
32      qsort(komsular, EGITIM_VERI_SAYI, sizeof(struct Komsu), karsilastir);
33
34      // En yakındaki N komşuyu al ve kaç hasta
35      int hasta = 0;
36      int hasta_degil = 0;
37
38      for (int i = 0; i < N; i++) {
39          if (komsular[i].sonuc == 4) {
40              hasta++;
41          }
42          else {
43              hasta_degil++;
44          }
45      }
46
47      // En yakın N komşuda hastalar hasta olmayanlardan fazla mı?
48      if (hasta > hasta_degil) {
49          return 4;
50      }
51
52      return 2;
53  }
```

ÖKLİD UZAKLIĞI

Bu algoritma beş adımdan oluşur.

Öncelikle K değeri belirlenir.

Diğer nesnelerden hedef nesneye olan öklit uzaklıkları hesaplanır.

Uzaklıklar sıralanır ve en minimum uzaklığa bağlı olarak en yakın komşular bulunur.

En yakın komşu kategorileri toplanır.

En uygun komşu kategorisi seçilir.

```c
#include "veri.h"

struct EgitimSeti egitimSetiniYukle() {
    struct EgitimSeti set;

    FILE* fp;

    fopen_s(&fp, EGITIM_DOSYA, "r");

    for (int i = 0; i < EGITIM_VERI_SAYI; i++) {
        char str[255];
        fgets(str, 255, fp);

        int _gozlemNumarasi;
        int clumpThickness;
        int uniformityOfCellSize;
        int uniformityOfCellShape;
        int marginalAdhesion;
        int singleEpithelialCellSize;
        int bareNuclei;
        int blandChromatin;
        int normalNucleoli;
        int mitoses;
        int sonuc;

        sscanf_s(str, "%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d", &_gozlemNumarasi, &clumpThickness, &uniformityOfCellSize, &uniformityOfCellShape, &marginalAdhesion, &singleEpithelialCellSize, &bareNuclei, &blandChromatin, &normalNucleoli, &mitoses, &sonuc);

        set.X[i][0] = clumpThickness;
        set.X[i][1] = uniformityOfCellSize;
        set.X[i][2] = uniformityOfCellShape;
        set.X[i][3] = marginalAdhesion;
        set.X[i][4] = singleEpithelialCellSize;
        set.X[i][5] = bareNuclei;
        set.X[i][6] = blandChromatin;
        set.X[i][7] = normalNucleoli;
        set.X[i][8] = mitoses;

        set.y[i] = sonuc;
    }

    fclose(fp);

    return set;
}
```

```c
struct TestSeti testSetiniYukle() {
    struct TestSeti set;

    FILE* fp;

    fopen_s(&fp, TEST_DOSYA, "r");

    for (int i = 0; i < TEST_VERI_SAYI; i++) {
        char str[255];
        fgets(str, 255, fp);

        int _gozlemNumarasi;
        int clumpThickness;
        int uniformityOfCellSize;
        int uniformityOfCellShape;
        int marginalAdhesion;
        int singleEpithelialCellSize;
        int bareNuclei;
        int blandChromatin;
        int normalNucleoli;
        int mitoses;
        int sonuc;

        sscanf_s(str, "%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d", &_gozlemNumarasi, &clumpThickness, &uniformityOfCellSize, &uniformityOfCellShape, &marginalAdhesion, &singleEpithelialCellSize, &bareNuclei, &blandChromatin, &normalNucleoli, &mitoses, &sonuc);

        set.X[i][0] = clumpThickness;
        set.X[i][1] = uniformityOfCellSize;
        set.X[i][2] = uniformityOfCellShape;
        set.X[i][3] = marginalAdhesion;
        set.X[i][4] = singleEpithelialCellSize;
        set.X[i][5] = bareNuclei;
        set.X[i][6] = blandChromatin;
        set.X[i][7] = normalNucleoli;
        set.X[i][8] = mitoses;

        set.y[i] = sonuc;
    }

    fclose(fp);

    return set;
}
```

```h
#ifndef parametreler_h
#define parametreler_h

#define N 5
#define PARAMETRE_SAYI 9

#define EGITIM_VERI_SAYI 600
#define EGITIM_DOSYA "egitim.txt"

#define TEST_VERI_SAYI 83
#define TEST_DOSYA "test.txt"

#endif /* parametreler_h */
```

```c
#ifndef tahmin_h
#define tahmin_h

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "parametreler.h"

struct Komsu {
    float oklid_uzaklik;
    int sonuc;
};

float oklid_uzaklik(int[], int[]);

int karsilastir(const void* komsu1, const void* komsu2);

int tahmin(int X_test[PARAMETRE_SAYI]);

#endif /* tahmin_h */
```

```c
#ifndef egitim_h
#define egitim_h

#include <stdio.h>
#include "parametreler.h"

struct EgitimSeti {
    int X[EGITIM_VERI_SAYI][PARAMETRE_SAYI];
    int y[EGITIM_VERI_SAYI];
};

struct EgitimSeti egitimSetiniYukle(void);

struct TestSeti {
    int X[TEST_VERI_SAYI][PARAMETRE_SAYI];
    int y[TEST_VERI_SAYI];
};

struct TestSeti testSetiniYukle(void);


#endif /* egitim_h */
```

SONUÇLAR

- Öklit uzaklığının nasıl hesaplanacağını ,txt'den veri okumayı ve girilen veriler ile ekrana sonuç yazdırmayı öğredik.