Laporan Tugas Program 3 Kecerdasan Buatan
Ahmad Arsyel Abdul H. / 1301164193
IF-40-06

A. Analisis Masalah

Permasalahan dalam program tugas 3 ini adalah menentukan 200 klasifikasi data dari *test set* dengan 4 label/kelas berbeda (0, 1, 2, dan 3). Diberikan 800 data *train set* dengan 5 atribut *input* (X1, X2, X3, X4, X5) dan 1 *output* (Y) pada file *.csv* dan akan dipilih dengan algoritma tertentu, sehingga label/kelas pada data *test* dapat ditentukan.

B. Metode Penyelesaian

Berdasarkan analisis permasalahan di atas, algoritma *K-Nearest Neighbor* merupakan salah satu solusi yang dapat diterapkan pada model permasalahan klasifikasi berdasarkan data yang telah memiliki target (*supervised learning*).

K-Nearest Neighbor (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap data (*test set*) pada objek dalam menentukan kelas berdasarkan jarak terdekat sejumlah nilai K tetangganya dari *train set*. Rumus yang digunakan untuk menghitung jarak diantara data *train set* dan *test set* adalah *Euclidean Distance,* dengan rumus sebagai berikut :

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Dipilihnya *Euclidean Distance* sebagai rumus untuk menghitung jarak karena data yang diberikan (X1, X2, X3, X4, dan X5) berupa nilai kontinu. Sehingga jarak dapat dihitung lebih akurat dan bervariasi. Hasilnya, jarak tiap data akan diurutkan terlebih dahulu secara *ascending*, kemudian dipilih sebanyak K, lalu di *voting*. Kelas yang paling banyak muncul akan menjadi kelas dari *test set*.

Dalam menentukan nilai K terbaik, dilakukanlah proses validasi dengan memecah 2 data pada *train set* menjadi *validation set* dan *train set* baru. Proses validasi lakukan dengan metode *KFold Cross Validation*, yang merupakan proses untuk mengambil data *sampling* secara acak dari *train set* untuk dijadikan *validation set*. Kemudian dihitung rata-rata nilai untuk setiap akurasi pada data yang telah divalidasi.

Dengan membuat sebanyak x data *sampling* yang telah diacak, kemudian dihitung akurasinya dengan *KFold*. Setelah sebanyak 'data acak' dihitung, nilai K untuk *nearest neighbor* berubah menjadi K = *i*4+1*, dimana *i* merupakan iterasi. Perhitungan tersebut dipilih untuk menghindari ambiguitas, karena jumlah *voting* yang bernilai genap dalam menentukan kelas atau terdapat tepat 1 kelas pada jarak K terdekat yang diambil.

Tabel 1 Hasil running program

Try Running	1	2	3	4	5	6	7	8	9	10
Fold	10	10	10	5	5	5	5	8	10	10
Generate random data	100	100	100	100	100	100	100	1000	1000	1000
Estimated Time	1h	1h	1h	45m 20s	45m 20s	45m 20s	58 m	6h 30m	3h 30m	3h 30m

K	13	13	13	9	13	9	13	13	9	13
Accuracy	80.95	80.9375	80.85	83.4375	83.425	83.5	81.54	81.809	80.699	80.7687

Tabel di atas merupakan hasil percobaan *running program* yang dibuat sebanyak 10 kali dengan parameter banyaknya K untuk *KFold Cross Validation* dan data acak yang dibuat dalam sekali *running*. Implementasi program yang dibuat untuk menentukan target dari 200 data *test* yang disediakan adalah dengan bahasa pemrograman Lua dan IDE ZeroBrane Studio.

C. Penjelasan Fungsi

Pada gambar 1, setelah proses *sorting* selesai, dilakukanlah implementasi *KFold Cross Validation*. Dihitung kebenaran akurasi data validasi dengan label aslinya dan nilainya ditampung untuk dilakukan rata-rata dari akurasi tiap K.

```
for i=1,k_fold do -- k-fold cross validation
  for j=1, #result_tab[#result_tab] do -- data validation
    insertion_sort(result_tab[i][j])
    local knearest_data = get_k_neighbor(result_tab[i][j], knn)
    is_label_similar = (
        generate_new_label(knearest_data) == result_tab[i][j][count_cross].true_label
    )

    if not is_label_similar then
      count_label = count_label + 1
    end

    is_label_similar = true
    count_cross = count_cross + 1
  end
  best_folds[i] = count_label -- do save here, most_error
  --print("fold " .. i,"error check",count_label, "---------------------------")
  count_label = 0
  count_cross = 1
end

local error_label = 1
local most_error = best_folds[error_label]
for i=2,#best_folds do
  if best_folds[i] > most_error then
    most_error = best_folds[i]
    error_label = i
  end
end
--print("random data: " .. j, "k-fold: " .. k_fold, "\nknn: " .. knn)
--print("fold: " .. error_label,"high error: " .. most_error)
accuracy = 100 - most_error/#result_tab[#result_tab]*100 -- rata-rata akurasi data untuk kfold
```

Gambar 1 Fungsi KFold Cross Validatiom

Hasil akurasi dipilih dari x data acak yang telah ditentukan, sehingga nilai K di dapat. Dari hasil running di tabel 1, dapat disimpulkan bahwa K optimum adalah 13.

```
knn is on running...
generate 100 new random data train base index position...
range of iterate 7..
with kfold: 10
iterate: 1      in random data: 48      k: 5    85      Final avg:80.1625

iterate: 2      in random data: 11      k: 9    83.75   Final avg:80.9375

iterate: 3      in random data: 91      k: 13   85      Final avg:80.95

iterate: 4      in random data: 5       k: 17   83.75   Final avg:80.3625

iterate: 5      in random data: 20      k: 21   83.75   Final avg:80.2875

iterate: 6      in random data: 59      k: 25   83.75   Final avg:80.1

iterate: 7      in random data: 19      k: 29   83.75   Final avg:79.1875

Execute to test_list
picked k from validation        13      that accuracy 80.95
file saved to CSV file
```

Gambar 2 Output running program