Lembar Kerja Ujian Tengah Semester
Mata Kuliah Sistem Cerdas

Kode Dosen : AGB

Anggota Kelompok:

- Shibgotalloh Sabilana G64180002
- Muhammad Rayhan Adyatma G64180064
- Muhamad Khoiru Tobi Albertino G64180065
- Rafi Solichin G64180068
- Ihsan Fadilla Wika Putra G64180071

Permasalahan

Perhatikan 60 observasi dari 3 kelas (kelas 1, kelas 2 dan kelas 3) yang masing-masing obyek diamati 3 peubah (yaitu X1, X2, dan X3). Oleh karena mahalnya melakukan pengukuran terhadap kelas, maka beberapa tidak diketahui kelasnya (ND=No Data), tetapi hanya bisa diamati peubah X1 hingga X3-nya. Berdasar data tersebut, anda diminta untuk mendapatkan model LVQ terbaik, dan tentukan tingkat akurasi dari model terbaiknya tersebut. Silakan dikerjakan berkelompok (1 kelompok terdiri 5 orang), dan dikumpulkan paling lambat 24 April ke email : agusbuono@apps.ipb.ac.id (silakan dikerjakan dengan bantuan komputer)

petunjuk:

- Bagilah data menjadi data training, data validasi dan data uji
- Hitung akurasi model terbaik untuk data uji yg anda tentukan

Learning Vector Quantization (LVQ) adalah salah suatu metode klasifikasi dari Jaringan Syaraf Tiruan. LVQ bekerja dengan setiap unit output mempresentasikan sebuah kelas. Metode pembelajaran pada lapisan kompetitif yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor *input*. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor *input*. Jika dua vektor *input* mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor *input* tersebut kedalam kelas yang sama. Tujuan dari algoritma ini adalah untuk mendekati distribusi kelas vektor untuk meminimalkan kesalahan dalam pengklasifikasian.

Langkah-langkah dari algoritma LVQ sendiri terdiri atas :

- **Step 1** inisialisasi layer input, epoch, bobot dan learning rate

- **Step 2** hitung bobot terdekat untuk setiap input
- **Step 3** pada setiap input, update bobot yang jaraknya paling dekat dengan input tersebut
- **Step 4** jika epoch belum terpenuhi, ulangi langkah ke-2 dan ke-3
- **step 5** prediksi kelas data baru dengan cara mengukur jarak antara data baru dan bobot. bobot terdekat akan dianggap kelas dari data baru tersebut.

Terdapat beberapa faktor yang mempengaruhi nilai akurasi dari proses LVQ yaitu epoch (iterasi), alpha (learning rate),dan weight (bobot). Disini kelompok kami menggunakan aplikasi R untuk membantu dalam proses pengerjaannya.

Dan kami menggunakan 3 buah library untuk membantu kami dalam membuat model yaitu:

```
library(class)
library(caret)
library(DMwR2)
```

Pertama-tama masukan dataset lalu lakukan normalisasi terhadap dataset tersebut

```
#Fungsi Normalisasi
normalize <- function(data){
  temp <- NULL
  for (i in data) {
    temp <- append(temp, (2*(i - min(data))/(max(data)-min(data))) - 1)
  }
  return(temp)
}

#Memasukkan dataset
df = read.csv("Sisdas_UTS_dataset.csv")
df$x1 <- normalize(df$x1)
df$x2 <- normalize(df$x2)
df$x3 <- normalize(df$x3)
df$kelas = as.factor(df$kelas)
View(df)
```

Dikarenakan terdapat 22 data yang tidak memiliki data, maka dilakukan proses pengisian kelas terlebih dahulu terhadap data-data tersebut dengan menggunakan KNN imputation

```
#isi NA dengan KNN imputation
```

```
df <- knnImputation(df)
View(df)
```

Selanjutnya bagi dataset tersebut menjadi tiga yaitu data train, data validasi dan data uji. Adapun proporsi dari pembagian data tersebut adalah 50% data train, 25% data validasi dan 25% data uji.

```
#Ambil 50% data train, 25% data validasi, 25% data uji
#Split data test dan train
set.seed(69420)
index <- sample(1:nrow(df), round(0.5*nrow(df)))
train <- df[index,]
test <- df[-index,]

#Split Test menjadi validasi dan test
newindex <- sample(1:nrow(test), round(0.5*nrow(test)))
val <- test[newindex,]
test <- test[-newindex,]
```

Setelah itu bentuk ketiga data tersebut ke dalam matriks

```
#Merubah dataset menjadi matriks.
train_m = data.matrix(train[, c("x1","x2","x3")])
test_m = data.matrix(test[, c("x1","x2","x3")])
val_m = data.matrix(val[, c("x1", "x2", "x3")])
```

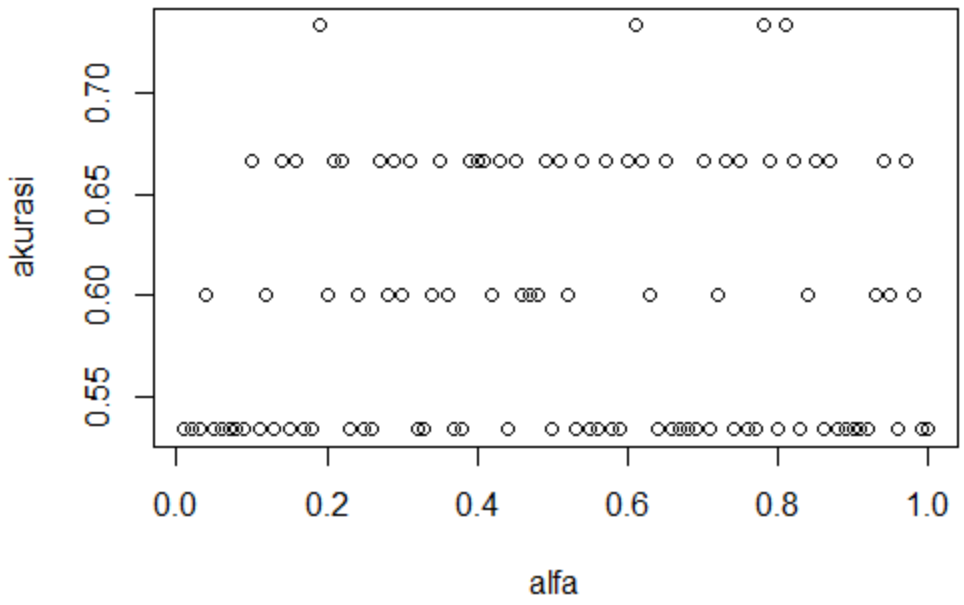
Selanjutnya lakukan proses pencarian model terbaik dengan menggunakan metode LVQ. untuk melakukan hal tersebut dilakukan pengujian terhadap data validasi

```
#Cari model terbaik
akurasi = NULL
alfa = NULL
for (i in 1:100) {
  #Membuat codebook untuk LVQ
  codeBook = lvqinit(train_m, train_label, size = 100)
  buildCodeBook = olvq1(train_m, train_label, codeBook, alpha = i/100)

  #Melakukan prediksi
  predict = lvqtest(buildCodeBook, val_m)
```

```
 #menghitung akurasi
 sum = 0
 for (j in 1:length(predict)) {
  if(predict[j] == val_label[j]){
   sum = sum+1
  }
 }
 akurasi = append(akurasi, sum/length(val_label))
 alfa = append(alfa, i/100)
}
plot(alfa, akurasi)
```

Didapatkan Hasil plot berikut:



Berdasarkan hasil plot tersebut, ditemukan model terbaik adalah saat alpha / learning rate bernilai 0.19, 0.61, 0.78, dan 0.81 dari hasil tersebut digunakan model terbaik dengan learning rate paling rendah, yaitu 0.19

Melakukan proses training model LVQ dengan alpha =0.19

```
codeBook = lvqinit(train_m, train_label, size = 100)
buildCodeBook = olvq1(train_m, train_label, codeBook, alpha = 0.19)
```

Lalu terakhir kita mengukur tingkat akurasi dari proses yang telah dijalankan

```
predict = lvqtest(buildCodeBook, test_m)
confusionMatrix(test_label, predict)
```

```
Confusion Matrix and Statistics

          Reference
Prediction 1 2 3
         1 7 0 1
         2 1 1 0
         3 0 1 4

Overall Statistics

               Accuracy : 0.8
                 95% CI : (0.5191, 0.9567)
    No Information Rate : 0.5333
    P-Value [Acc > NIR] : 0.03209

                  Kappa : 0.6591

 Mcnemar's Test P-Value : 0.39163

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.8750  0.50000   0.8000
Specificity            0.8571  0.92308   0.9000
Pos Pred Value         0.8750  0.50000   0.8000
Neg Pred Value         0.8571  0.92308   0.9000
Prevalence             0.5333  0.13333   0.3333
Detection Rate         0.4667  0.06667   0.2667
Detection Prevalence   0.5333  0.13333   0.3333
Balanced Accuracy      0.8661  0.71154   0.8500
> |
```

Dan didapat besar akurasi yang diperoleh cukup baik yaitu sebesar 80%.

Kesimpulan :

Pada proses LVQ ini kami membagi input data menjadi 3 dengan proporsi 50% data train, 25% data validasi dan 25% data uji. Setelah semua proses berjalan didapatkan proses terbaik diperoleh saat nilai alpha sebesar 0.19, dan epoch sebanyak **40*30 (120).** Yaitu mendapatkan akurasi sebesar 80%**.**

Dari beberapa percobaan kami sebelumnya, nilai inisiasi bobot awal sangat mempengaruhi proses training untuk mencapai akurasi yang baik, untuk itu kami menggunakan library “class” dan menggunakan “olvq1” (Optimized Learning Vector Quantization 1) untuk memperbaiki model LVQ yang kami buat.