**PERBANDINGAN PENERAPAN METODE *BACK PROPAGATION NEURAL NETWORK* (BPNN) DAN *LEARNING VECTOR QUANTIZATION* (LVQ) DENGAN KASUS PENYAKIT KANKER PAYUDARA**

**Topan Try Harmanda1, Sukarna2, Yanto Hermawan3**

**Program Studi S2 Ilmu Komputer, FMIPA Institut Pertanian Bogor**

**1.** **PENDAHULUAN**

Kanker yang paling umum pada wanita adalah kanker payudara, kanker ini merupakan salah satu penyebab utama kematian pada wanita di seluruh dunia saat ini. Pada kanker payudara merupkan sel kanker terbentuk dalam jaringan payudara wanita. Payudara terdiri dari lobus mengandung 15 sampai 20 bagian dan saluran. Jenis yang paling umum dari kanker payudara dimulai di sel-sel saluran. Kanker payudara didefinisikan sebagai suatu penyakit neoplasma yang ganas yang berasal dari parenchyma.3 Sejak tahun 2008, kejadian kanker payudara telah meningkat lebih dari 20% sedangkan angka kematian akibat kanker payudara telah meningkat sebesar 14% Kanker payudara juga merupakan penyebab paling umum kematian pada kanker di kalangan perempuan (522.000 kematian pada tahun 2012) dan kanker yang paling sering didiagnosis pada wanita di 140 dari 184 negara di seluruh dunia dan saat ini menempati peringkat 1 dari 4 kanker yang mematikan pada wanita. Terdapat 70% penderita dalam kondisi stadium lanjut akibat berbagai masalah kesehatan dalam penanganannya.

Kanker yang dimulai di lobus atau lobulus ditemukan di kedua payudara jenis lain dari kanker payudara. Hangat, merah, dan bengkak payudara merupakan indikator untuk kanker payudara. Usia dan riwayat kesehatan dapat mempengaruhi risiko mengembangkan kanker payudara. Untuk mendeteksi tahapan yang berbeda dari kanker payudara, dada X-ray, CT scan, pemindaian tulang dan PET scan secara luas digunakan. Jumlah diagnosis kanker payudara dihitung menjadi 1,2 juta di antara perempuan setiap tahun menurut proyeksi oleh Organisasi Kesehatan Dunia.

Pada Penelitian terdahulu dijelaskan mengenai Klasifikasi Penyakit Kanker Payudara dengan menggunakan Data mining, hasil dari penelitian ini ialah mengembangkan model untuk mengindetifikasi penyakit dan mengklasifikasi penyakit dengan akurasi yang tinggi. Penelitian berikutnya adalah penelitian berjudul Diagnosis Kanker Payudara pada Tiga Datasets berbeda, Hasil eksperimen dalam acara dataset WBC bahwa perpaduan antara MLP dan J48 mengklasifikasi dengan fitur seleksi (PCA) lebih unggul pengklasifikasi lainnya. Di dataset sisi WDBC lain menunjukkan bahwa menggunakan pengklasifikasi tunggal (SMO) atau menggunakan fusi SMO dan MLP atau SMO dan IBK lebih baik dari pengklasifikasi lainnya. Akhirnya, fusi MLP, J48, SMO dan IBK lebih unggul pengklasifikasi lainnya di WPBC dataset.

Penelitian ini membahas perbandingan kedua metode menggunakan *Back Propagation Neural Network (BPNN)* dan *Learning Vector Quantization* (LVQ) dengan Dataset dengan kasus Penyakit Kanker Payudara. BPNN dan LVQ adalah metode jaringan saraf tiruan yang dapat digunakan untuk identifikasi pola citra. BPNN merupakan jaringan saraf tiruan dengan metode pembelajaran supervised dimana output yang diharapkan sudah diketahui sebelumnya, sedangkan LVQ dengan metode unsupervised dimana tidak memerlukan target ouput. BPNN melatih jaringan untuk mendapatkan keseimbanan antara kemanpuan jaringan untuk mengenali pola yang digunkan selama pelatihan serta kemampuan jaringan untuk mngenali pola masukan yang serpa dengan pola yang diakai selama pelatihan (Siang, 2005). LVQ merupkan metode pembelajaran pada lapisa kompetitif yang akan secara otomatis belajar untuk mengklasifikasikan vector-vektor masukan (Kusumadewi, 2004).

Kedua metode ini memiliki kekurangan dan kelebihan masing masing, BPNN menggunakan 2 alur dalam penghitungan bobot ( yaitu propagasi maju/forward dan propagasi mundur/backward), sedangkan LVQ hanya menggunakan 1 alur (alur maju) dalam perhitungannya. Keduanya memiliki perbedaan dalam penentuan hasil pengenalan, maka penelitian ini mengambil judul Perbandingan Penerapan Metode *Back Propagation Neural Network* (BPNN) Dan *Learning Vector Quantization* (LVQ) pada Kasus Penyakit Kanker Payudara

**2.** **KAJIAN TEORI**

**2.1 Backpropagation**

*Backpropagation* adalah salah satu metode dari jaringan syaraf tiruan yang dapat diaplikasikan dengan baik dalam bidang peramalan (*forecasting*). *Backpropagation* melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan mengenali pola yang digunakan selama *training* serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa namun tidak sama dengan pola yang dipakai selama pelatihan (Siang, 2005).

Menurut Kusumadewi (2003), dalam pelatihan dengan *backpropagation* sama halnya seperti pelatihan pada jaringan syaraf yang lain. Pada jaringan *feedfoward* (umpan maju), pelatihan dilakukan dalam rangka perhitungan bobot sehingga pada akhir pelatihan akan diperoleh bobot-bobot yang baik. Selama proses pelatihan, bobot-bobot diatur secara iteratif untuk meminimumkan *error* (kesalahan) yang terjadi. Kesalahan dihitung berdasarkan rata-rata kuadrat kesalahan (MSE). Rata-rata kuadrat kesalahan dasar perhitungan untuk kerja fungsi aktivasi.

Sebagian besar pelatihan untuk jaringan *feedfoward* (umpan maju) menggunakan *gradient* dari fungsi aktivasi untuk menentukan bagaimana mengatur bobot-bobot dalam rangka meminimumkan kinerja. *Gradient* ini ditentukan dengan menggunakan suatu teknik yang disebut *backpropagation*. Prinsip dasar algoritma ini adalah memperbaiki bobot-bobot jaringan dengan arah yang membuat fungsi aktivasi menjadi turun dengan cepat. *Backpropagation* memiliki beberapa unit yang ada dalam satu atau lebih layar tersembunyi (Siang, 2005).

Fungsi aktivasi jaringan syaraf tiruan *backpropagation* yang dipakai harus memenuhi beberapa syarat yaitu: kontinyu, terdiferensial dengan mudah, dan merupakan fungsi yang tidak turun (Siang, 2005). Fungsi aktivasi diharapkan jenuh (mendekati nilai-nilai maksimum dan minimum secara asimtot). Beberapa fungsi yang digunakan menurut Puspitaningrum (2006) yaitu sigmoid biner, sigmoid bipolar, tangent hiperbolik.

**2.2 Learning Vector Quantization (LVQ)**

LVQ merupakan suatu metode untuk melakukan pelatihan terhadap lapisan-lapisan kompetitif yang terawasi. Lapisan kompetitif akan belajar secara otomatis untuk melakukan klasifikasi terhadap vektor input yang diberikan. Apabila beberapa vektor input memiliki jarak yang sangat berdekatan, maka vektor-vektor input tersebut akan dikelompokkan dalam kelas yang sama.

LVQ merupakan metode klasifikasi pola masing-masing unit keluaran mewakili kategori atau kelas tertentu (beberapa unit keluaran seharusnya digunakan untuk masing-masing kelas). Keunggulan dari metode LVQ adalah kemampuannya untuk memberikan pelatihan terhadap lapisan-lapisan kompetitif sehingga secara otomatis dapat mengklasifikasikan vektor input yang diberikan (Nugroho, 2011).

Kelebihan dari LVQ adalah :

1. Nilai error yang lebih kecil dibandingkan dengan jaringan syaraf tiruan seperti backpropagation.

2. Dapat meringkas data set yang besar menjadi vektor codebook berukuran kecil untuk klasifikasi.

3. Dimensi dalam codebook tidak dibatasi seperti dalam teknik nearest neighbour.

4. Model yang dihasilkan dapat diperbaharui secara bertahap.

Kekurangan dari LVQ adalah:

1. Dibutuhkan perhitungan jarak untuk seluruh atribut.

2. Akurasi model dangan bergantung pada inisialisasi model serta parameter yang

digunakan (learning rate, iterasi, dan sebagainya).

3. Akurasi juga dipengaruhi distribusi kelas pada data training.

4. Sulit untuk menentukan jumlah codebook vektor untuk masalah yang diberikan.

(Fausett, 1994)

**3.** **METODE**

**3.1 Data Penelitian**

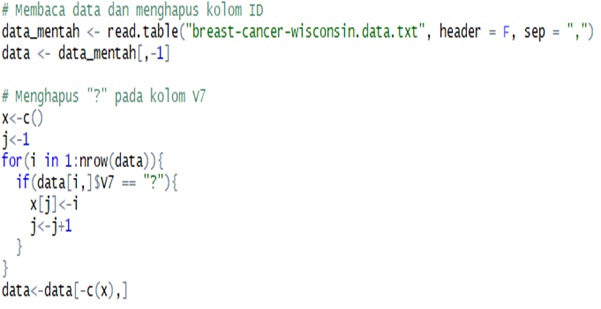
Data penelitian dalam penelitian ini menggunakan Breast Cancer Wisconsin (Original) Dataset yang dapat diunduh pada situs UCI Machine Learning Dataset. Dataset ini terdiri dari dua kelas yaitu kelas *benign* dengan 458 *record* dan kelas *malignant* dengan 241 *record*. Atribut-atribut dari dataset yang digunakan dapat dilihat pada Tabel 1.

Tabel 1. Atribut-atribut dari Breast Cancer Wisconsin (Original) Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Nama Atribut | Karakteristik | Tipe | Nilai |
| 1 | id\_number (V1) | Nomor ID pasien | numerik | random |
| 2 | clump\_thickness (V2) | Ketebalan kanker | numerik | 1 - 11 |
| 3 | cell\_size (V3) | Ukuran sel kanker | numerik | 1 - 11 |
| 4 | cell\_shape (V4) | Bentuk sel kanker | numerik | 1 - 11 |
| 5 | marginal\_adhesi (V5) | Adhesi | numerik | 1 - 11 |
| 6 | epithelial\_size (V6) | Ukuran sel epitel | numerik | 1 - 11 |
| 7 | bare\_nuclei (V7) | Nuklei atau inti | numerik | 1 - 11 |
| 8 | bland\_chromatin (V8) | Kromatin lunak | numerik | 1 - 11 |
| 9 | normal\_nucleoli (V9) | Nukleolus normal | numerik | 1 - 11 |
| 10 | mitoses (V10) | Mitosis | numerik | 1 – 11 |
| 11 | class (V11) | Kelas data | numerik | 2 (Benign) dan 4 (Malignant) |

**3.2 Praproses**

Data set ini harus dilakukan praproses tersebut seperti penghapusan kolom id\_number (V1) karena fitur tersebut tidak mempengaruhi model yang akan dibuat dan penghapusan *record* yang memiliki nilai “?” pada kolom bare\_nuclei (V7) karena *record* tersebut tidak dapat dipakai untuk pembentukan model. Kode di bahasa pemrograman R untuk praproses dapat dilihat pada Gambar 1.

****

Gambar 1. Kode praproses

Ketika dilakukan praproses maka kolom ID akan hilang sehingga atribut yang akan dipakai untuk pembuatan model adalah atribut V2 - V10 atau sebanyak 9 atribut dan atribut V11 sebagai kelas. *Record* dalam dataset setelah dilakukan praproses berubah menjadi sebanyak 683 *record*.

**3.3 Learning Vector Quantization (LVQ)**

Langkah langkah dari metode LVQ adalah sebagai berikut :

a. Dataset hasil praproses kemudian dipisahkan menjadi 2 kelas yaitu benign dan malignant sebagai target.

b. Vektor pewakil memiliki peranan penting dalam mendapatkan nilai akurasi yang paling baik. Vektor pewakil akan diuji coba membandingkan nilai akurasi yang didapat dari menggunkan vektor pewakil yang diambil acak dari setiap kelas dengan menggunakan vektor pewakil yang diambil acak dari keseluruhan data training.

Vector pewakil 1 dengan mengambil salah satu baris acak dari setiap kelas data train dan vector pewakil 2 dengan sebanyak 3 baris acak dari seluruh data train untuk menjadi vektor pewakil.

c. Menetapkan nilai awal learning rate / alfa = 0.4 dan beta = 0.04 dengan iterasi 100

d. Dilakukan training data dengan :

- Perhitungan Euclid satu baris data train dengan setiap vektor pewakil

- Mengambil index dari hasil perhitungan Euclid yang minimum

- Kondisi jika kelas data train sama dengan kelas vektor pewakil yang

perhitungan Euclid nya minimum

- Update vector pewakil yang perhitungan Euclid nya minimum

- Kondisi jika kelas data train tidak sama dengan kelas vektor pewakil yang

perhitungan Euclid nya minimum

- Update vector pewakil yang perhitungan Euclid nya minimum

e. Setelah train selesai kemudian mengupdate alfa dengan mengalilan alfa dan beta.

f. Melakukan data test dengan menghitung Euclid satu baris data train dengan setiap vektor pewakil.

g. Mengindex dari hasil perhitungan Euclid yang minimum

h. Jika kelas data test sama dengan kelas vektor pewakil yang perhitungan Euclid nya minimum

i. Menghitung jumlah benar terklasifikasi dan akurasi dengan membagi jumlah data benar dengan jumlah data test dikalikan 100.

**3.4 Metode Back Propagation Neural Network (BPNN)**

Langkah langkah dari metode BPNN adalah sebagai berikut :

a. Inisialisasi vektor referensi, learning rate, maks.epoch, neuron hidden layer, target error

b. Melakukan load data hasil praproses.

c. Training sampai dengan iterasi/epoch selesai.

d. Feedforward dari lapis masukan ke lapis tersembunyi

e. Feedforward dari lapis tersembunyi ke lapis keluaran

f. Backpropagation dari lapis keluaran ke lapis tersembunyi

g. Backpropagation dari lapis tersembunyi ke lapis masukan

h. Update bobot dengan atau tanpa momentum ()

i. Simpan delta bobot untuk perhitungan update bobot dengan momentum selanjutnya

j. Hitung error untuk seluruh data latih (1 epoch)

k. Lakukan pengujian untuk setiap data uji

l. Hitung akurasi pengujian (Recognition Rate – RR),

**4. HASIL PEMBAHASAN**

4.1 Metode Learning Vector Quantization *(*LVQ)

Hasil akurasi yang diperoleh dari data Breast Cancer Wisconsin dengan variasi beberapa learning rate, beta = 0.04 dan iterasi 100 dapat dilihat pada Tabel 2.

Tabel 2 Akurasi metode LVQ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Train | Data Test | Alpha (Akurasi) | | | | | |
| 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| 50 | 50 | 95,32164 | 95,02924 | 95,32164 | 95,32164 | 95,32164 | 95,32164 |
| 60 | 40 | 93,79562 | 93,79562 | 93,79562 | 93,79562 | 93,79562 | 93,79562 |
| 70 | 30 | 92,68293 | 92,68293 | 93,17073 | 93,17073 | 93,17073 | 93,17073 |
| 80 | 20 | 91,24088 | 91,24088 | 91,24088 | 91,24088 | 91,24088 | 91,24088 |
| 90 | 10 | 89,85507 | 89,85507 | 89,85507 | 89,85507 | 89,85507 | 89,85507 |

Dari tabel 2 didapatkan bahwa jaringan yang optimal dalam kasus ini adalah data train = 50 dan iterasi = 100 diperoleh akurasi sebesar 95.32164% sehingga efisien apabila digunakan dalam pengenalan Breast Cancer Wisconsin.

**4.2 Metode Back Propagation Neural Network (BPNN)**

Data hasil praproses akan dilakukan scaling atau normalisasi sebelum dimasukan ke dalam BPNN. Normalisasi dilakukan dengan menggunakan fungsi scale() yang sudah ada di dalam R. Algoritma BPNN menggunakan library neuralnet() yang dapat di download di dalam R studio maupun link library tersebut. Data hasil normalisasi akan dipisahkan dengan perbandingan 80% atau sebanyak 546 *record* untuk data latih dan 20% atau sebanyak 137 *record* untuk data uji. Output dari BPNN berupa confusion matriks yang melingkupi kelas sebenarnya dan kelas prediksi. Hasil *confusion matrix* yang diperoleh dari data Breast Cancer Wisconsin dengan variasi beberapa *hidden layer*, *learning rate* = 0.001 dan 0.0001, dan *threshold* = 0.01 dapat dilihat pada Tabel 3.

Tabel 3 Confusion Matrik Metode BPNN

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hidden Layer | Alpha | Confusion Matrix | | | | | | | | | | Akurasi (%) |
| 5 | 0,001 |  | 1 | 2 | 3 | 4 | 5 | 6 | 10 |  |  | 54,01459854 |
|  |  | 2 | 0 | 53 | 2 | 27 | 1 | 1 | 1 |  |  |  |
|  |  | 4 | 1 | 26 | 2 | 21 | 1 | 1 | 0 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | 0,001 |  | 1 | 2 | 3 | 4 | 5 | 6 | 10 |  |  | 87,59124088 |
|  |  | 2 | 1 | 76 | 1 | 4 | 0 | 0 | 0 |  |  |  |
|  |  | 4 | 0 | 3 | 3 | 44 | 2 | 2 | 1 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15 | 0,001 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 78,10218978 |
|  |  | 1 | 0 | 1 | 77 | 3 | 1 | 0 | 2 | 0 | 1 |  |
|  |  | 2 | 1 | 0 | 3 | 7 | 30 | 9 | 1 | 1 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 | 0,001 |  | -1 | 0 | 1 | 2 | 4 | 6 |  |  |  | 90,51094891 |
|  |  | 2 | 2 | 2 | 1 | 83 | 4 | 0 |  |  |  |  |
|  |  | 4 | 0 | 0 | 0 | 3 | 41 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 25 | 0,001 |  | -1 | 1 | 2 | 4 | 6 |  |  |  |  | 96,35036496 |
|  |  | 2 | 1 | 1 | 82 | 1 | 0 |  |  |  |  |  |
|  |  | 4 | 0 | 0 | 1 | 50 | 1 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 | 0,001 |  | 2 | 3 | 4 |  |  |  |  |  |  | 97,08029197 |
|  |  | 2 | 85 | 2 | 1 |  |  |  |  |  |  |  |
|  |  | 4 | 0 | 1 | 48 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 35 | 0,0001 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |  |  | 77,37226277 |
|  |  | 2 | 0 | 1 | 80 | 3 | 1 | 0 | 0 |  |  |  |
|  |  | 4 | 1 | 0 | 1 | 14 | 26 | 9 | 1 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 40 | 0,0001 |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  | 78,83211679 |
|  |  | 2 | 1 | 89 | 1 | 0 | 1 |  |  |  |  |  |
|  |  | 4 | 2 | 4 | 6 | 19 | 14 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 45 | 0,0001 |  | 1 | 2 | 3 | 4 | 5 | 6 |  |  |  | 80,2919708 |
|  |  | 2 | 0 | 85 | 2 | 1 | 1 | 0 |  |  |  |  |
|  |  | 4 | 2 | 1 | 12 | 25 | 7 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 50 | 0,0001 |  | 0 | 1 | 2 | 3 | 4 | 5 |  |  |  | 8,653846154 |
|  |  | 2 | 1 | 1 | 1 | 85 | 1 | 4 |  |  |  |  |
|  |  | 4 | 0 | 0 | 0 | 3 | 8 | 0 |  |  |  |  |

Dari tabel 2 didapatkan bahwa jaringan yang optimal dalam kasus ini adalah dengan hidden layer sebanyak 30 node dengan akurasi sebesar 97.08% maka dianggap paling cocok untuk digunakan dalam dataset Breast Cancer Wisconsin.

**5. KESIMPULAN**

Dari hasil penelitian dan pengujian terhadap data Breast Cancer Wisconsin menggunakan BPNN dengan metode LVQ pada penelitian ini, didapatkan simpulan sebagai berikut :

1. Terdapat 326 (95.32164%) hasil pengenalan cocok menggunakan metode LVQ dan 133

(97.01%) hasil pengenalan cocok menggunakan metode BPNN dengan hidden layer sebanyak 30 node.

2. Tingkat kecocokan dan akurasi hasil pengenalan tergantung pada variasi nilai parameter yang digunakan dalam proses pembelajarannya. Dari hasil pengujian, maka didapatkan variasi parameter terbaik dari BPNN yaitu *learning rate* = 0,001 dan jumlah hidden layer = 30. Sedangkan variasi parameter terbaik dari *LVQ* yaitu *learning rate* = 0,4 dan jumlah data train = 50.

**6. DAFTAR PUSTAKA**

Kusumadewi. 2004. *Membangun Jaringan Syaraf Tiruan (Menggunakan Matlab dan Excel Link)*. Yogyakarta: Graha Ilmu

Siang, J.J. 2005. *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan MATLAB.* Yogyakarta: ANDI Puspitaningrum, D. 2006. *Pengantar Jaringan Syaraf Tiruan*. Yogyakarta: ANDI

Sitio. Roma 2019. *Pengalaman Psikososial Pasien Kanker Payudara Yang Menjalani Terapi Kemoterapi Di Blud Dr. Zainoel Abidin Banda Aceh*. Jurnal Keperawatan Priority 2(1) : 2614-4719.

Pusat Data dan Informasi Kementrian Kesehatan RI. (2013). Diakses pada tanggal 29 April 2019 dari https://www.google. com/search/ pusdatinkemenkesri.

Nugroho, F. F. Kurniawan, Arif, Y. M.& Dermawan, D. A., 2011. *Simulasi Multi Atribut di Dasarkan pada Agen untuk Kehandalan Distribusi Energi Listrik menggunakan Metode LVQ.* Seminar on Electrical, Informatics and ITS education, Pp. 55-63

LVQ

# Membaca data dan menghapus kolom ID

data\_mentah <- read.table("d:\\breast-cancer-wisconsin.data.txt", header = F, sep = ",")

data <- data\_mentah[,-1]

# Menghapus "?" pada kolom V7

x<-c()

j<-1

for(i in 1:nrow(data)){

if(data[i,]$V7 == "?"){

x[j]<-i

j<-j+1

}

}

data<-data[-c(x),]

# Mengubah kolom V7 dari vector menjadi integer

data$V7 <- as.integer(data$V7)

str(data)

# Memisahkan kelas benign dan malignant dari data

benign <- subset(data\_mentah, V11 == 2)

malignant <- subset(data\_mentah, V11 == 4)

# Melakukan pemisahan data train dan data test

set.seed(1234)

jumlah\_index <- floor(0.5 \* nrow(data))

index <- sample(seq\_len(nrow(data)), size = jumlah\_index)

train <- data[index,]

test <- data[-index,]

benign <- subset(train, V11 == 2)

malignant <- subset(train, V11 == 4)

# Vektor pewakil

# Pewakil = Mengambil salah satu baris acak dari setiap kelas data train untuk menjadi vektor pewakil

# Pewakil2 = Mengambil sebanyak 3 baris acak dari seluruh data train untuk menjadi vektor pewakil

indvec\_benign <- sample(seq\_len(nrow(benign)), 1)

indvec\_malignant <- sample(seq\_len(nrow(malignant)), 1)

vec\_benign <- benign[indvec\_benign,]

vec\_malignant <- malignant[indvec\_malignant,]

pewakil <- rbind(vec\_benign, vec\_malignant)

ind\_acak <- sample(seq\_len(nrow(train)),3)

pewakil2 <- train[ind\_acak,]

# Inisiasi

hasil <- c()

a <- c()

index\_hasil <- c()

alfa <- 0.4

beta <- 0.04

iterasi <- 100

benar <- 0

index\_hasiltest <- c()

# SOM Training

for (k in 1:iterasi) {

for (i in 1:nrow(train)) {

# Perhitungan Euclid satu baris data train dengan setiap vektor pewakil

for (j in 1:2) {

sum = ((train[i,]$V2 - pewakil[j,]$V2)^2)+

((train[i,]$V3 - pewakil[j,]$V3)^2)+

((train[i,]$V4 - pewakil[j,]$V4)^2)+

((train[i,]$V5 - pewakil[j,]$V5)^2)+

((train[i,]$V6 - pewakil[j,]$V6)^2)+

((train[i,]$V7 - pewakil[j,]$V7)^2)+

((train[i,]$V8 - pewakil[j,]$V8)^2)+

((train[i,]$V9 - pewakil[j,]$V9)^2)+

((train[i,]$V10 - pewakil[j,]$V10)^2)

hasil[j] = sqrt(sum)

}

# Mengambil index dari hasil perhitungan Euclid yang minimum

index\_hasil[i] <- which.min(hasil)

a[i] <- hasil[index\_hasil[i]]

# Kondisi jika kelas data train sama dengan kelas vektor pewakil yang perhitungan Euclid nya minimum

if (train[i,]$V11 == pewakil[index\_hasil[i],]$V11){

# Update vector pewakil yang perhitungan Euclid nya minimum

pewakil[index\_hasil[i],]$V2 <- pewakil[index\_hasil[i],]$V2 +

(alfa\*(train[i,]$V2 - pewakil[index\_hasil[i],]$V2))

pewakil[index\_hasil[i],]$V3 <- pewakil[index\_hasil[i],]$V3 +

(alfa\*(train[i,]$V3 - pewakil[index\_hasil[i],]$V3))

pewakil[index\_hasil[i],]$V4 <- pewakil[index\_hasil[i],]$V4 +

(alfa\*(train[i,]$V4 - pewakil[index\_hasil[i],]$V4))

pewakil[index\_hasil[i],]$V5 <- pewakil[index\_hasil[i],]$V5 +

(alfa\*(train[i,]$V5 - pewakil[index\_hasil[i],]$V5))

pewakil[index\_hasil[i],]$V6 <- pewakil[index\_hasil[i],]$V6 +

(alfa\*(train[i,]$V6 - pewakil[index\_hasil[i],]$V6))

pewakil[index\_hasil[i],]$V7 <- pewakil[index\_hasil[i],]$V7 +

(alfa\*(train[i,]$V7 - pewakil[index\_hasil[i],]$V7))

pewakil[index\_hasil[i],]$V8 <- pewakil[index\_hasil[i],]$V8 +

(alfa\*(train[i,]$V8 - pewakil[index\_hasil[i],]$V8))

pewakil[index\_hasil[i],]$V9 <- pewakil[index\_hasil[i],]$V9 +

(alfa\*(train[i,]$V9 - pewakil[index\_hasil[i],]$V9))

pewakil[index\_hasil[i],]$V10 <- pewakil[index\_hasil[i],]$V10 +

(alfa\*(train[i,]$V10 - pewakil[index\_hasil[i],]$V10))

}

# Kondisi jika kelas data train tidak sama dengan kelas vektor pewakil yang perhitungan Euclid nya minimum

else {

# Update vector pewakil yang perhitungan Euclid nya minimum

pewakil[index\_hasil[i],]$V2 <- pewakil[index\_hasil[i],]$V2 -

(alfa\*(train[i,]$V2 - pewakil[index\_hasil[i],]$V2))

pewakil[index\_hasil[i],]$V3 <- pewakil[index\_hasil[i],]$V3 -

(alfa\*(train[i,]$V3 - pewakil[index\_hasil[i],]$V3))

pewakil[index\_hasil[i],]$V4 <- pewakil[index\_hasil[i],]$V4 -

(alfa\*(train[i,]$V4 - pewakil[index\_hasil[i],]$V4))

pewakil[index\_hasil[i],]$V5 <- pewakil[index\_hasil[i],]$V5 -

(alfa\*(train[i,]$V5 - pewakil[index\_hasil[i],]$V5))

pewakil[index\_hasil[i],]$V6 <- pewakil[index\_hasil[i],]$V6 -

(alfa\*(train[i,]$V6 - pewakil[index\_hasil[i],]$V6))

pewakil[index\_hasil[i],]$V7 <- pewakil[index\_hasil[i],]$V7 -

(alfa\*(train[i,]$V7 - pewakil[index\_hasil[i],]$V7))

pewakil[index\_hasil[i],]$V8 <- pewakil[index\_hasil[i],]$V8 -

(alfa\*(train[i,]$V8 - pewakil[index\_hasil[i],]$V8))

pewakil[index\_hasil[i],]$V9 <- pewakil[index\_hasil[i],]$V9 -

(alfa\*(train[i,]$V9 - pewakil[index\_hasil[i],]$V9))

pewakil[index\_hasil[i],]$V10 <- pewakil[index\_hasil[i],]$V10 -

(alfa\*(train[i,]$V10 - pewakil[index\_hasil[i],]$V10))

}

}

# Update alfa

alfa <- beta \* alfa

}

index\_hasil

# Testing data test

for (i in 1:nrow(test)) {

# Perhitungan Euclid satu baris data train dengan setiap vektor pewakil

for (j in 1:2) {

sum = ((test[i,]$V2 - pewakil[j,]$V2)^2)+

((test[i,]$V3 - pewakil[j,]$V3)^2)+

((test[i,]$V4 - pewakil[j,]$V4)^2)+

((test[i,]$V5 - pewakil[j,]$V5)^2)+

((test[i,]$V6 - pewakil[j,]$V6)^2)+

((test[i,]$V7 - pewakil[j,]$V7)^2)+

((test[i,]$V8 - pewakil[j,]$V8)^2)+

((test[i,]$V9 - pewakil[j,]$V9)^2)+

((test[i,]$V10 - pewakil[j,]$V10)^2)

hasil[j] = sqrt(sum)

}

# Mengambil index dari hasil perhitungan Euclid yang minimum

index\_hasiltest[i] <- which.min(hasil)

a[i] <- hasil[index\_hasiltest[i]]

# Kondisi jika kelas data test sama dengan kelas vektor pewakil yang perhitungan Euclid nya minimum

if (pewakil[index\_hasiltest[i],]$V11== test[i,]$V11){

benar = benar + 1

}

}

# Menghitung jumlah benar terklasifikasi dan akurasi

benar

akurasi <- (benar / nrow(test))\*100

akurasi

BPNN

# Membaca data dan menghapus kolom ID

data\_mentah <- read.table("d:\\breast-cancer-wisconsin.data.txt", header = F, sep = ",")

data <- data\_mentah[,-1]

# Menghapus "?" pada kolom V7

x<-c()

j<-1

for(i in 1:nrow(data)){

if(data[i,]$V7 == "?"){

x[j]<-i

j<-j+1

}

}

data<-data[-c(x),]

# Mengubah kolom V7 dari vector menjadi integer

data$V7 <- as.integer(data$V7)

str(data)

temp <- c()

for(i in 1:nrow(data)){

temp[i] <- data[i,]$V11

}

datatotal <- data[,-10]

kelas <- data[,10]

scaleddata <- scale(datatotal)

datatemp <- cbind(scaleddata, kelas)

ind <- floor(0.8 \* nrow(datatemp))

index <- sample(seq\_len(nrow(datatemp)), size = ind)

train <- datatemp[index,]

test <- datatemp[-index,]

library(neuralnet)

nn <- neuralnet(kelas ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10, data = train, hidden = 5, threshold = 0.01, learningrate = 0.001)

nn$result.matrix

#plot(nn)

#Test the resulting output

temp\_test <- subset(test, select = c("V2","V3", "V4", "V5", "V6", "V7", "V8", "V9", "V10"))

head(temp\_test)

nn.results <- compute(nn, temp\_test)

results <- data.frame(actual = test[,10], prediction = nn.results$net.result)

results$prediction <- round(results$prediction)

table(results$actual,results$prediction)