PEMBELAJARAN MESIN
Teori dan Studi Kasus

Tim Penyusun

Penerbit :

Canting Mas Anyar

PEMBELAJARAN MESIN
Teori Dan Studi Kasus

Tim Penyusun : Emi Susilowati, S.Kom., M.Kom.
Priadhana Edi Kresnha, S.Kom., M.Kom.
Aulia Syifa
Noviarum Widyasmara L
Amelia Tri Hapsari
Muhammad Faizal
Andri Nurhadi
Fernanda Awalia
Yudo Witnu Prasetyo
Yusup Hidayat Winata

Layout dan Desain : Sugiartowo

Penerbit : Canting Mas Anyar
Perum Agatama Regency Banguntapan A8
Yogyakarta

ISBN : 978-602-521692-9-9

" Buku Ini Merupakan Bagian Dari Materi Kuliah Pembelajaran Mesin."

KATA PENGANTAR

Assalamua'alaikum wr. wb.
Bismillahirrohmanirrohiim

Puja dan puji syukur kami haturkan sepenuhnya kepada Allah SWT, yang atas rahmat dan nikmat yang diberikan, buku Pembelajaran Mesin – Teori & Studi Kasus - ini dapat diselesaikan. Berkat doa tim penulis dan pihak-pihak pendukung pula, buku ini bisa diselesaikan dengan baik, walaupun mungkin masih jauh dari kata sempurna. Bisa jadi ada beberapa poin yang akan diketahui nanti, yang harus diperbaiki untuk menjadikan buku ini lebih sempurna.

Buku ini disusun untuk membantu Perkuliahan Pembelajaran Mesin (*machine learning*), dimana penulisnya kesemuanya adalah peserta kuliah pembelajaran mesin. Jadi buku ini adalah hasil *learning by doing* dari peserta kuliahnya. Karena buku ini disusun bersamaan dengan kuliahnya, dan tentu dalam kuliah selalu dibahas teori dan problem secara detail, maka secara otomatis buku ini merupakan karya *fresh from the oven*. Dibuat ketika ilmu masih melekat dengan erat di kepala seluruh penulis, sehingga hasilnya sangat memuaskan serta dijamin lengkap dan detail.

Akhir kata, silakan membaca dan mempelajari buku ini. Semoga buku ini bermanfaat bagi para pembaca, dan bisa menambah ilmu dan khazanah, serta mengembangkan kemampuan iptek di negeri kita.

Wassalamu'alaikum wr. wb.
Jakarta, Juli 2020

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1. REGRESI

1.1 Pendahuluan

Regresi adalah metode yang digunakan untuk mengetahui hubungan atau pengaruh variabel x (variabel independen) terhadap variabel y (variabel dependen). Hubungan atau pengaruh tersebut kemudian akan sdirepresentasikan dalam bentuk model regresi (fungsi/persamaan matematis). Perlu diingat, dalam regresi variabel x merupakan variabel yang mempengaruhi nilai variable y. Umumnya regresi digunakan untuk memprediksi/mengestimasikan nilai variabel y berdasarkan model regresi yang dibentuk dan nilai variabel x yang diketahui. Regresi termasuk metode yang mudah digunakan dikarenakan rumusnya yang sederhana dan mudah diaplikasikan di berbagai bidang, contohnya dalam bidang Pembelajaran Mesin (*machine learning*). Pada buku ini akan dibahas tiga jenis regresi, yaitu:

1. Regresi Linear
2. Regresi Polinomial
3. Regresi Multivariate

1.2 Regresi Linear

1.2.1 Pendahuluan

Regresi linear merupakan regresi yang paling sederhana karena hanya terdiri dari dua buah variabel, yaitu sebuah variabel x dan sebuah variabel y. Sesuai dengan namanya, regresi linear akan memodelkan hubungan antara variabel x dan variabel y dalam bentuk fungsi linear (garis).

1.2.2 Persamaan Matematis Regresi Linear

Model (persamaan matematis) regresi linear secara umum dapat dituliskan sebagai berikut:

$y = a + bx$ .....(1.1)

dimana:

- x = variable independen
- y = variabel dependen
- a = nilai y ketika x bernilai nol (*intercept*)
- b = koefisien regresi

Nilai koefisien regresi (b) bisa dicari menggunakan rumus berikut:

$b = \frac{N \sum xy - (\sum x \sum y)}{N \sum x^2 - (\sum x)^2}$ .....(1.2)

dimana N merupakan banyaknya data dalam dataset/tabel.

Sedangkan nilai *intercept* (a) bisa dicari menggunakan rumus berikut:

$a = \bar{y} - b\bar{x}$ .....(1.3)

dimana:

- $\bar{y}$ = rata-rata nilai variable dependen
- $\bar{x}$ = rata-rata nilai variable independent
- b = koefisien regresi (bisa dicari menggunakan persamaan 2)

1.2.3 Prosedur / Algoritma Regresi Linear

Prosedur atau algoritma regresi linear adalah sebagai berikut:

1. Tentukan variabel x (variabel independen) dan variabel y (variabel dependen) dari dataset/tabel.
2. Cari nilai *intercept* (b) dengan menggunakan persamaan (1.2).
3. Cari nilai koefisien regresi (a) dengan menggunakan persamaan (1.3).
4. Buat model regresi linear dengan masukkan nilai *intercept* (a) dan koefisien regresi (b) ke persamaan (1.1).
5. Lakukan prediksi variabel y (variabel dipenden) menggunakan model regresi linear yang telah dibuat pada langkah ke empat.
6. Buat garis regresi linear pada *scatters* dataset/tabel berdasarkan model regresi linear yang telah dibuat pada langkah ke empat.

1.2.4 Sample Case (Sucahyono, 2015)

Seorang bidan membuka praktik bersalin di rumahnya. Bidan tersebut selalu mencatat berat badan ibu hamil (kg) dan berat badan bayi (gram) yang dilahirkan. Sejauh ini terdapat 10 data yang berhasil dikumpulkan oleh si bidan. Data tersebut dapat dilihat pada tabel 1.1. Tentukan prediksi berat badan bayi yang akan lahir jika berat badan ibu yang mengandungnya saat ini adalah 80kg!

Tabel 1. 1 Berat Badan Bayi yang Dilahirkan Berdasarkan Berat Badan Ibu Hamil

Berat Badan Ibu Hamil (kg)	Berat Badan Bayi (gram)
49.4	3515
63.5	3742
68	3629
52.5	2880
54.4	3008
70.3	4068
50.8	3373
73.9	4124
65.8	3572
54.4	3359

Berdasarkan tabel 1.1 berat badan ibu hamil merupakan variabel x (variabel independen) dikarenakan **memberikan pengaruh** terhadap berat badan bayi yang dilahirkan. Sedangkan berat badan bayi yang **dipengaruhi** oleh berat badan ibu yang mengandungnya merupakan variabel y (variabel dependen). Nilai a dan nilai b dapat dicari setelah variabel x dan y diketahui.

Tabel 1. 2 Mencari Nilai Tiap Elemen Pada Persamaan (1.2) dan (1.3)

No.	x	y	xy	x^2
1	49.4	3515	173641	2440.36
2	63.5	3742	237617	4032.25
3	68	3629	246772	4624
4	52.5	2880	151200	2756.25

5	54.4	3008	163635.2	2959.36
6	70.3	4068	285980.4	4942.09
7	50.8	3373	171348.4	2580.64
8	73.9	4124	304763.6	5461.21
9	65.8	3572	235037.6	4329.64
10	54.4	3359	182729.6	2959.36
Jumlah	603	35270	2152724.8	37085.16
Rata-rata	60.3	3527		

Cari nilai b dengan memasukkan nilai elemen-elemen yang dibutuhkan pada tabel 1.2 ke persamaan (1.2).

$$b = \frac{10\,(2{,}152{,}724.8) - (603)(35{,}270)}{10\,(37{,}085.16) - (603)^2}$$

$$b = 35.8211$$

Cari nilai a dengan memasukkan nilai a, $\bar{x}$, dan $\bar{y}$ ke persamaan (1.3).

$$a = 3527 - 35.8211\,(60.3)$$

$$a = 1366.9868$$

Buat model regresi linear dengan memasukkan nilai a dan b ke persamaan (1.1).

$$y = a + bx$$

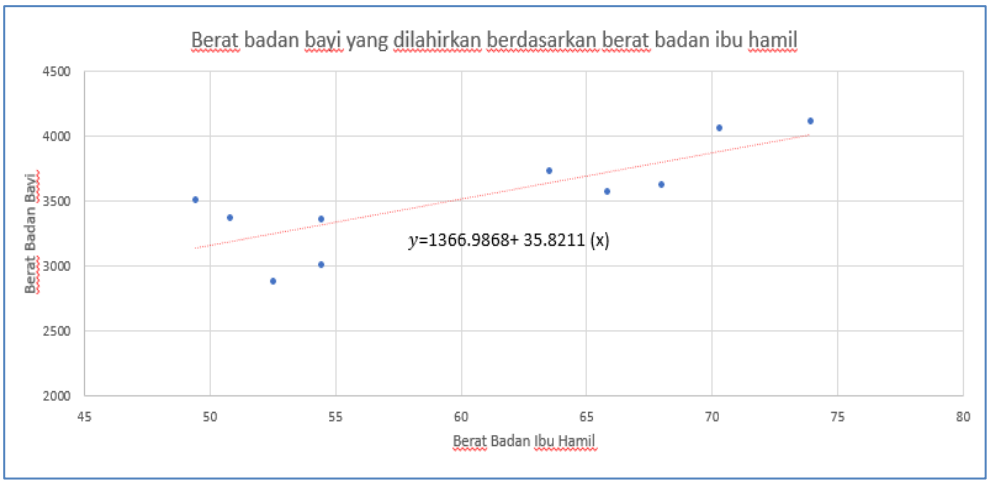
$$y = 1366.9868 + 35.8211\,(x)$$

Lakukan prediksi berat badan bayi yang akan dilahirkan dengan memasukkan berat badan ibu hamil yang diketahui dalam soal ke model regresi linear yang telah dibuat.

$$y = 1366.9868 + 35.8211\,(80)$$

$$y = 4232.676$$

Jadi prediksi berat badan bayi yang akan dilahirkan jika ibu yang mengandungnya memiliki berat badan 80kg adalah sebesar 4232.676 gram. Selain untuk melakukan prediksi, model regresi linear yang telah dibuat sebelumnya juga dapat digunakan untuk membuat garis regresi linear pada *scatters* dataset/tabel seperti yang dapat dilihat pada gambar 1.1.



Gambar 1. 1 Garis Regresi Linear Pada Scatters Dataset/Tabel

1.3 Regresi Polinomial

1.3.1 Pendahuluan

Regresi polinomial biasa dikenal dengan nama "regresi non-linear". Sama halnya dengan regresi linear, regresi polinomial juga hanya terdiri dari sebuah variabel x dan sebuah variabel y. Perbedaannya regresi polinomial akan memodelkan hubungan antara variabel x dan variabel y dalam bentuk fungsi polinomial (kurva).

Regresi polinomial digunakan ketika error yang dihasilkan oleh model regresi linear sangat besar (tidak akurat). Dalam regresi polinomial dikenal "ordo regresi". Semakin tinggi ordo yang digunakan maka semakin pas/fit model yang dibentuk terhadap data, namun semakin sensitif pula data terhadap model regresi. Data yang terlalu sensitif dapat menurunkan tingkat akurasi prediksi. Salah satu solusi untuk menangani masalah ini adalah dengan menggunakan model regresi polinomial ber-ordo rendah.

1.3.2 Persamaan Matematis Regresi Polinomial

Model (persamaan matematis) regresi linear secara umum dapat dituliskan sebagai berikut:

$$y = b_0 + b_1(x) + b_2(x^2) + \cdots + b_n(x^n) \quad \text{.....(1.4)}$$

dimana:

- x = variable independen
- y = variabel dependen
- $b_0 \ldots b_n$ = koefisien regresi ke-0 sampai dengan ke-n

Nilai $b_0 \ldots b_n$ dapat dicari dengan menggunakan Eliminasi Gauss Jordan sebagai berikut:

$$\begin{bmatrix} N & \sum x & \sum x^2 \cdots \\ \sum x & \sum x^2 & \sum x^3 \cdots \\ \sum x^2 & \sum x^3 & \sum x^4 \cdots \\ \cdots & \cdots & \ddots \end{bmatrix}^{-1} \text{X} \begin{bmatrix} \sum y \\ \sum x^2 y \\ \sum x^3 y \\ \sum x^n y \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_n \end{bmatrix} \quad \text{.....(1.5)}$$

N merupakan banyaknya data dalam dataset/tabel. Banyaknya ordo matriks pada Eliminasi Gauss Jordan tergantung pada jumlah ordo regresi polinomial yang digunakan.

1.3.3 Prosedur / Algoritma Regresi Polinomial

Prosedur atau algoritma regresi polinomial adalah sebagai berikut:

1. Tentukan variabel x (variabel independen) dan variabel y (variabel dependen) dari dataset/tabel.
2. Tentukan ordo regresi polinomial yang akan digunakan. Semakin tinggi ordo regresi polinomial yang digunakan maka akan semakin tinggi pula tingkat akurasi dan sensitivitas hasil prediksi.
3. Cari nilai $b_0 \ldots b_n$ dengan menggunakan persamaan (1.5).
4. Buat model regresi polinomial dengan masukkan nilai $b_0 \ldots b_n$ ke persamaan (1.4).
5. Lakukan prediksi variabel y (variabel dipenden) menggunakan model regresi polinomial yang telah dibuat pada langkah ke empat.
6. Buat garis regresi linear pada *scatters* dataset/tabel berdasarkan model regresi polinomial yang telah dibuat pada langkah ke empat.

1.3.4 Sample Case

Sebuah perusahaan minuman mengadakan survei mengenai dampak jumlah mesin minuman yang disebar di daerah Cempaka Putih terhadap jumlah pembeli. Dataset hasil

survei dapat dilihat pada tabel 1.3. Tentukan model persamaan regresi polinomial orde dua dari dataset tersebut dan tentukan berapa jumlah pembeli jika perusahaan memasang mesin minuman sebanyak 5 buah!

Tabel 1. 3 Dampak Jumlah Mesin Minuman Terhadap Jumlah Pembeli

Jumlah Mesin Minuman yang Disebar	Jumlah Pembeli
1	10
3	55
4	68
6	73
7	65
9	30

Berdasarkan tabel 1.3, jumlah mesin minuman yang disebar adalah variabel x (variabel independen) dikarenakan **memberikan pengaruh** terhadap jumlah pembeli. Sedangkan jumlah pembeli yang **dipengaruhi** oleh seberapa banyak mesin minuman disebar merupakan variabel y (variabel dependen). Ordo regresi polinomial yang diminta pada soal adalah bernilai 2, maka ordo martiks pada Eliminasi Gauss Jordan akan berjumlah 3X3 dan 3X1. Selanjutnya Nilai $b_0 \dots b_2$ dapat dicari setelah variabel x, variabel y, dan ordo matriks diketahui.

Tabel 1. 4 Mencari Nilai Tiap Elemen Pada Persamaan (1.5)

No.	x	x	x^2	x^3	x^4	x	x^2y
1	1	10	1	1	1	10	10
2	3	55	9	27	81	165	495
3	4	68	16	64	256	272	1088
4	6	73	36	216	1296	438	2628
5	7	65	49	343	2401	455	3185
6	9	30	81	729	6561	270	2430
Jumlah	30	301	192	1380	10596	1610	9836

Cari nilai $b_0 \dots b_2$ dengan memasukkan nilai elemen-elemen yang dibutuhkan pada tabel 1.4 ke persamaan (1.5).

$$\begin{bmatrix} 6 & 30 & 192 \\ 30 & 192 & 1380 \\ 192 & 1380 & 10596 \end{bmatrix}^{-1} \text{X} \begin{bmatrix} 301 \\ 1610 \\ 9836 \end{bmatrix} = \begin{bmatrix} 2.0476 & -0.8333 & 0.0714 \\ -0.8333 & 0.4206 & -0.0397 \\ 0.0714 & -0.0397 & 0.004 \end{bmatrix} \text{X} \begin{bmatrix} 301 \\ 1610 \\ 9836 \end{bmatrix} = \begin{bmatrix} -22.7619 \\ 36.0714 \\ -3.3571 \end{bmatrix}$$

Buat model regresi polinomial dengan memasukkan nilai $b_0 \dots b_2$ ke persamaan (1.4).

$y = -22.7619 + 36.0714(x) - 3.3571(x^2)$

Lakukan prediksi jumlah pembeli dengan memasukkan jumlah mesin minuman yang disebar (dalam soal) ke model regresi polinomial yang telah dibuat.

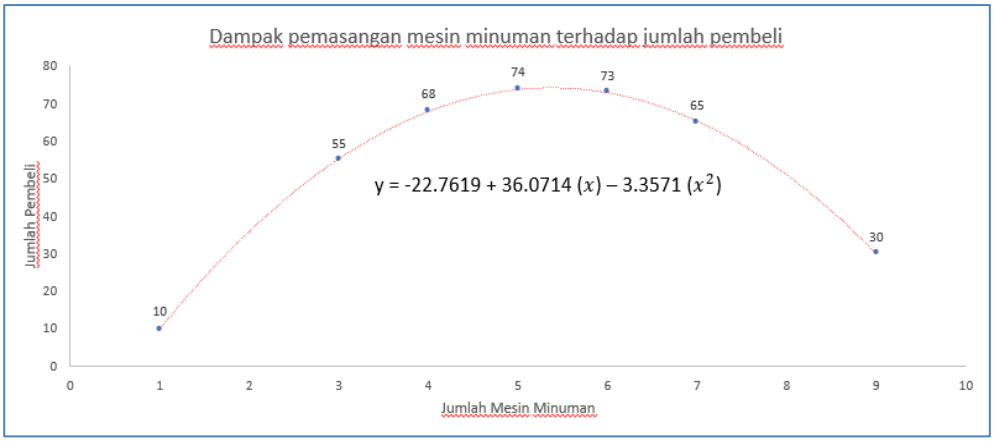
$y = -22.7619 + 36.0714(5) - 3.3571(5^2)$

$y = 73.667$

$y \approx 74\ orang$

Jadi prediksi jumlah pembeli jika pabrik memasang 5 buah mesin minuman secara tersebar di daerah Cempaka Putih adalah sebanyak 74 orang. Selain untuk melakukan

prediksi, model regresi polinomial yang telah dibuat sebelumnya juga dapat digunakan untuk membuat garis regresi linear pada *scatters* dataset/tabel seperti yang dapat dilihat pada gambar 1.2.



Gambar 1. 2 Garis Regresi Polinomial Pada Scatters Dataset/Tabel

1.4 Regresi Multivariate

1.4.1 Pendahuluan

Berbeda dengan regresi linear dan regresi polinomial, regresi multivariate merupakan regresi yang memodelkan hubungan antara lebih dari satu variabel x dengan sebuah variabel y. Regresi ini digunakan ketika sebuah variabel dependen dipengaruhi oleh lebih dari satu variabel independen.

1.4.2 Persamaan Matematis Regresi Multivariate

Model (persamaan matematis) regresi linear secara umum dapat dituliskan sebagai berikut:

$$y = b_0 + b_1(x_1) + b_2(x_2) + \cdots + b_n(x_n) \quad \text{.....(1.6)}$$

dimana:

- $x_0 \ldots x_n$ = variable independen ke-1 sampai dengan ke-n
- y = variabel dependen
- $b_0 \ldots b_n$ = koefisien regresi ke-0 sampai dengan ke-n

Nilai $b_0 \ldots b_n$ dapat dicari dengan menggunakan Eliminasi Gauss Jordan sebagai berikut:

$$\begin{bmatrix} N & \sum x_1 & \sum x_2 & \cdots \\ \sum x_1 & \sum {x_1}^2 & \sum x_1 x_2 & \cdots \\ \sum x_2 & \sum x_1 x_2 & \sum {x_2}^2 & \cdots \\ \cdots & \cdots & \cdots & \ddots \end{bmatrix}^{-1} \text{X} \begin{bmatrix} \sum y \\ \sum x_1 y \\ \sum x_2 y \\ \sum x_n y \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_n \end{bmatrix} \quad \text{.....(1.7)}$$

N merupakan banyaknya data dalam dataset/tabel. Banyaknya ordo matriks pada Eliminasi Gauss Jordan tergantung pada jumlah variabel x (variabel independen).

1.4.3 Prosedur / Algoritma Regresi Multivariate

Prosedur atau algoritma regresi polinomial adalah sebagai berikut:

1. Tentukan variabel $x_0 \ldots x_n$ (variabel independen) dan variabel y (variabel dependen) dari dataset/tabel.
2. Cari nilai $b_0 \ldots b_n$ dengan menggunakan persamaan (1.7).

3. Buat model regresi multivariate dengan masukkan nilai $b_0 \dots b_n$ ke persamaan (1.6).
4. Lakukan prediksi variabel y (variabel dipenden) menggunakan model regresi multivariate yang telah dibuat pada langkah ke empat.
5. Buat garis regresi linear pada *scatters* dataset/tabel berdasarkan model regresi multivariate yang telah dibuat pada langkah ke empat.

1.4.4 Sample Case

Agnes melakukan penelitian pengaruh pemberian pupuk kandang (gram) dan arang sekam (gram) terhadap rata-rata pertumbuhan tinggi tanaman cabai (mm) perhari. Penelitian dilakukan selama satu minggu dengan menggunakan 8 tanaman cabai. Hasil penelitian Agnes dapat dilihat pada tabel 1.5. Tentukan rata-rata pertumbuhan tinggi tanaman cabai jika Agnes memberikan pupuk kandang sebanyak 43 gram dan arang sekam sebanyak 52 gram!

Tabel 1. 5 Pengaruh Pemberian Pupuk dan Arang Sekam Terhadap Pertumbuhan Tanaman Cabai

Pupuk Kandang (gram)	Arang Sekam (gram)	Tinggi Tanaman Cabai (mm)
10	8	3
12	11	4
15	16	6
20	18	8
24	22	12
28	25	14
29	31	17
38	37	23

Berdasarkan tabel 1.5, pupuk kandang dan arang sekam adalah variabel x_1 dan x_2 (variabel independen) dikarenakan **memberikan pengaruh** terhadap pertumbuhan tinggi tamaman cabai. Sedangkan tinggi tanaman cabai yang **dipengaruhi** oleh pemberian pupuk dan arang sekam merupakan variabel y (variabel dependen). Terdapat dua variabel independen pada dataset/tabel, maka ordo martiks pada Eliminasi Gauss Jordan akan berjumlah 3X3 dan 3X1. Selanjutnya Nilai $b_0 \dots b_2$ dapat dicari setelah variabel x_1 & x_2, variabel y, dan ordo matriks diketahui.

Tabel 1. 6 Mencari Nilai Tiap Elemen Pada Persamaan (1.7)

No.	x_1	x_2	y	${x_1}^2$	${x_2}^2$	x_1x_2	x_1y	x_2y
1	10	8	3	100	64	80	30	24
2	12	11	4	144	121	132	48	44
3	15	16	6	225	256	240	90	96
4	20	18	8	400	324	360	160	144
5	24	22	12	576	484	528	288	264
6	28	25	14	784	625	700	392	350
7	29	31	17	841	961	899	493	527
8	38	37	23	1444	1369	1406	874	851
Jumlah	176	168	87	4514	4204	4345	2375	2300

Cari nilai $b_0 \ldots b_2$ dengan memasukkan nilai elemen-elemen yang dibutuhkan pada tabel 1.6 ke persamaan (1.7).

$$\begin{bmatrix} 8 & 176 & 168 \\ 176 & 4514 & 4345 \\ 168 & 4345 & 4204 \end{bmatrix}^{-1} \text{X} \begin{bmatrix} 87 \\ 2375 \\ 2300 \end{bmatrix} = \begin{bmatrix} 0.9561 & -0.0972 & 0.0622 \\ -0.0972 & 0.0528 & -0.0507 \\ 0.0622 & -0.0507 & 0.0502 \end{bmatrix} \text{X} \begin{bmatrix} 87 \\ 2375 \\ 2300 \end{bmatrix} = \begin{bmatrix} -4.4885 \\ 0.3642 \\ 0.35 \end{bmatrix}$$

Buat model regresi multivariate dengan memasukkan nilai $b_0 \ldots b_2$ ke persamaan (1.6).

$y = -4.4885 + 0.3642\ (x_1) + 0.35\ (x_2)$

Lakukan prediksi pertumbuhan tinggi rata-rata tanaman cabai dengan memasukkan jumlah pupuk kandang dan arang sekam (yang diketahui dalam soal) ke model regresi multivariate yang telah dibuat.

$y = -4.4885 + 0.3642\ (43) + 0.35\ (52)$

$y = 29.3744mm$

Jadi prediksi pertumbuhan tinggi rata-rata tanaman cabai jika Agnes memberikan 43 gram pupuk kandang dan 52 gram arang sekam adalah 29.3744 milimeter. Garis regresi multivariate tidak bisa disajikan dalam *scatter* karena berbentuk empat dimensi (4D) atau lebih.

1.5 Ringkasan Regresi

Regresi hanya menangani dua jenis variabel, yaitu variabel x dan variabel y, dimana variabel x (variabel independen) menentukan/mempengaruhi nilai variabel y (variabel dependen). Regresi adalah metode yang digunakan untuk mengetahui hubungan antara dua jenis variabel tersebut dengan merepresentasikannya ke dalam bentuk model regresi (fungsi/persamaan matematis). Regresi umumnya digunakan untuk memprediksi/mengestimasikan nilai variabel y berdasarkan model regresi yang dibentuk dan nilai variabel x yang diketahui. Buku ini membahas tiga jenis regresi, yakni:

1. Regresi linear
2. Regresi polinomial
3. Regresi multivariate

Regresi linear merupakan regresi yang paling sederhana. Regresi polinomial digunakan ketika model regresi linear memiliki error yang tinggi. Regresi multivariate digunakan ketika sebuah variabel y dipengaruhi oleh lebih dari satu variabel x. Nilai $b_0 \ldots b_n$ pada regresi polinomial dan multivariate dapat dicari menggunakan Eliminasi Gauus.

1.6 Latihan Regresi

Seorang seniman ingin membuat sebuah patung modern menggunakan bahan logam campuran. Untuk mendapatkan kelengkungan patung yang sesuai, seniman tersebut melakukan penelitian pengaruh suhu terhadap pertambahan panjang logam campuran (pemuaian). Hasil penelitian si seniman dapat dilihat pada tabel 1.6.

Tabel 1. 7 Pengaruh Suhu Terhadap Pertambahan Panjang Logam Campuran

Suhu (^oC)	Pertambahan Panjang Logam (micrometer)
101	48555
153	33567
208	20812
252	17670

309	15905
346	19821
399	21316
461	32078
512	40000
541	53115

Tentukan prediksi pertambahan panjang logam campuran jika logam campuran tersebut dipanaskan pada suhu 1000 ^{o}C! Kerjakan menggunakan regresi yang paling cocok dengan dataset/tabel 1.6 (regresi yang menghasilkan model dengan error terkecil)!

BAB 2. K-MEANS CLUSTERING

2.1.Pendahuluan

Clustering merupakan metode pengelompokan data kedalam suatu wilayah yang akan berisi data yang sama atau hampir serupa. Clustering ini dibedakan menjadi 2 (dua) yaitu Metode *Hierarchical* dan Metode *Non-Hierarchical*. Metode *Hierarchical* yaitu metode mengelompokkan dua atau lebih data yang memiliki kesamaan hampir serupa. Sedangkan Metode *Non-Hierarchical* yaitu metode mengelompokkan data dengan menentukan jumlah cluster yang diinginkan, dan barulah proses cluster dilakukan tanpa mengikuti proses hirarki. Salah satu contoh metode *Non-Hierarchical* yaitu K-Means Clustering.

K-Means Clustering ini disebut sebagai metode *Partitional Clustering*, yaitu metode pengelompokan data kedalam satu cluster yang sama, dan apabila terdapat data yang mempunyai karakteristik beberapa maka akan dikelompokan kedalam cluster lain. Metode K-Means Clustering ini memiliki beberapa kelebihan antara lain, mudah diimplementasikan pada kehidupan sehari hari, algoritma yang digunakan tidak rurmit, dan waktu yang dibutuhkan relative lebih cepat. Namun setiap kelebihan pasti selalu ada kekurangan. Pada K-Means Clustering ini memiliki beberapa kekurangan yaitu proses penentuan cluster awal dilakukan secara random sehingga pengelompokkan data bisa menjadi tidak optimal, dan apabila ingin mengelompokkan data dengan puluhan atau ratusan dimensi akan terjadi kesulitan untuk mencari jarak terdekat antara cluster awal dengan cluster lainnya. Pada dasarnya, penggunaan algoritma K-Means tergantung pada data yang didapatkan dan hasil yang ingin dicapai.

2.2.Persamaan Matematis K-Means Clustering

Menurut (Wikipedia) berikut rumus umum yang digunakan pada K-Means Clustering untuk meminimalkan jumlah dalam cluster kuadrat adalah sebagai berikut:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 = \arg\min_{S} \sum_{i-1}^{k} |S_i| \, Var \, S_i \quad (2.1)$$

Dimana *k* merupakan pengelompokan data; x_1, x_2, …, x_n = vektor nyata; S_1, S_2, …, S_k = meminimalkan jumlah dalam cluster kuadrat; μ_i = rata-rata poin dalam S_i

Ini sama dengan meminimalkan penyimpangan kuadrat berpasangan dari poin dalam kluster yang sama:

$$\arg\min_{S} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{x,y \in S_i} \|x - y\|^2 \quad (2.2)$$

Dapat disimpulkan:

$$\sum_{x,y \in S_i} \|x - y\|^2 = \sum_{x \neq y \in S_i} (x - \mu_i)(\mu_i - y) \quad (2.2)$$

Karena total varians konstan, ini setara dengan memaksimalkan jumlah deviasi kuadrat antara titik-titik dalam kelompok yang berbeda (antara cluster jumlah kuadrat), yang mengikuti dari hukum total varians.

Selanjutnya menghitung euclidean untuk mengukur jarak data dengan centroid, yang di dalamnya terdapat persamaan sebagai berikut:

$$d(x, y) = |x - y| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2.4)$$

Berdasarkan persamaan (2.4) maka didapat rumus sebagai berikut:

$$dC = |x - y| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \quad (2.5)$$

Dimana *dC* merupakan jarak setiap elemen objek data dengan setiap elemen centroid; *x* = objek data; *y* = centroid

Untuk selanjutnya dilakukan pembaruan centroid untuk mendapatkan nilai terbaru dari jarak data dengan centroid, ini dilakukan dengan cara membagi jumlah masing-masing atribut dari anggota cluster dengan dimensi data. Berikut persamaan yang didapat:

$$New C_i = \frac{(x_1 + x_2 + \cdots + x_n)}{n} \quad (2.6)$$

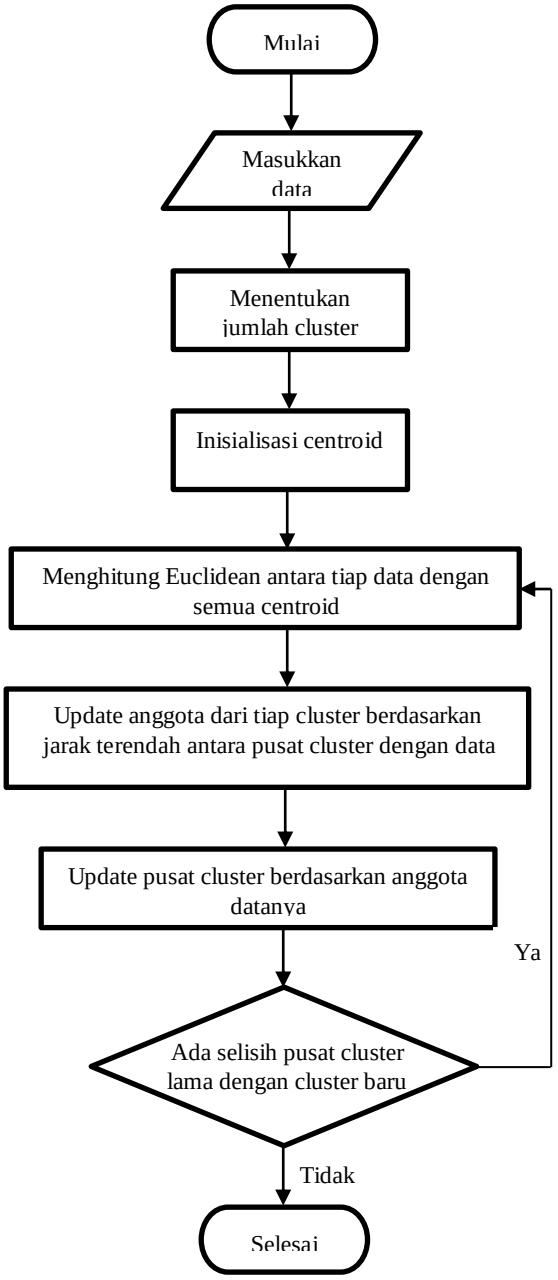
Dimana x_j = adalah data ke-j dari cluster i, dan n = banyak data pada cluster i.

2.3.Prosedur / Algoritma K-Means Clustering

Langkah-langkah dalam perhitungan algoritma dasar dalam K-Means Clustering menurut (Wardhani, 2016) dapat dijelaskan sebagai berikut:

1. Menentukan jumlah cluster
   Untuk menentukan banyaknya cluster pada objek data, maka diperlukan pertimbangan secara teori dan terencana yang diuslkan untuk menentukan berapa banyak cluster.
2. Inisialisasi centroid/penentuan pusat awal cluster
   Ditentukan secara random sehingga mendapatkan hasil titik pusat yang objektif.
3. Menghitung Euclidean
   Nilai Euclidean didapatkan dengan melakukan pengurangan antara objek data dengan centroid, perhitungan tersebut dapat dilihat pada persamaan (2.4) diatas. Hal ini dilakukan untuk mengukur jarak antara data dengan centroid.
4. Perbarui centroid
   Nilai centroid harus diperbarui dengan keanggotaan cluster yang baru, dimana x = banyaknya anggota cluster baru dibagi dengan n = dimensi data. Untuk perhitungan update centroid dapat dilihat pada persamaan (2.6) diatas.
5. Hitung ulang Euclidean
   Terakhir ulangi menghitung Euclidean seperti pada persamaan (2.4) dan perbarui centroid seperti pada persamaan (2.6) hingga kondisi nilai konvergen, yaitu dimana ketika tidak ada data yang berpindah cluster dan perubahan posisi centroid sudah berada diambang batas.

Langkah-langkah dalam perhitungan algoritma dasar dalam K-Means Clustering dapat dijelaskan melalui flowchart berikut:



Gambar 2. 1 Flowchart K-Means Clustering

2.4.Sample Case

Dilansir dari laman yang dipublish oleh (Syafrudin, 2015) terdapat sampel dataset Padi pada tahun 2013 di Provinsi Jawa Timur yaitu sebanyak 12 Kabupaten. Dataset yang diambil menggunakan luas lahan dan produksi. Percobaan dilakukan dengan menggunakan parameter berikut:
Jumlah cluster = 3
Jumlah data = 12
Jumlah atribut = 2

Tabel 2. 1 Sample Case

No	Kota/Kab	Luas Lahan	Produksi
1	Kab. Ponorogo	66,693.00	402,047.00
2	Kab. Trenggalek	31,136.00	182,848.00
3	Kab. Tulungagung	49,230.00	259,581.00
4	Kab. Blitar	50,577.00	289,494.00
5	Kab. Kediri	51,083.00	281,392.00
6	Kab. Malang	65,597.00	464,498.00
7	Kab. Lumajang	72,552.00	387,168.00
8	Kab. Jember	162,619.00	964,001.00
9	Kab. Banyuwangi	113,609.00	706,419.00
10	Kab. Bondowoso	61,330.00	329,557.00
11	Kab. Situbondo	48,902.00	290,954.00
12	Kab. Probolinggo	59,130.00	311,258.00

ITERASI 1

1. Penetuan Pusat Awal Cluster

Pusat awal cluster menggunakan 3 sampel data yaitu menggunakan data ke-8, data ke-7, dan data ke-2.

Tabel 2. 2 Pusat Awal Cluster Iterasi 1

Di ambil data ke-8 sebagai pusat cluster ke-1	162.619	964.001
Di ambil data ke-7 sebagai pusat cluster ke-2	72.552	387.168
Di ambil data ke-2 sebagai pusat cluster ke-3	31.136	182.848

2. Perhitungan Jarak Pusat Cluster

Untuk mengukur jarak dengan pusat cluster menggunakan Rumus Euclidian Distance seperti pada persamaan:

$$d(x,y) = |x-y| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Berikut adalah cara perhitungan untuk mendapatkan nilai pada cluster C(1,1) :

C(1,1) $= \sqrt{(66.693 - 162.619)^2 + (402.047 - 964.001)^2}$
$= \sqrt{(-95.926)^2 + (-561.9554)^2}$
$= \sqrt{9201.7974 + 315798.8715}$
$= 570.0825$

- Berikut adalah cara perhitungan untuk mendapatkan nilai pada cluster C(2,1) :

C(2,1) $= \sqrt{(66.693 - 72.552)^2 + (402.047 - 387.168)^2}$
$= \sqrt{(-5.859)^2 + (14.879)^2}$
$= \sqrt{34.3278 + 221.3846}$
$= 15.99101$

- Berikut adalah cara perhitungan untuk mendapatkan nilai pada cluster C(3,1) :

$$C(3,1) = \sqrt{(66.693 - 31.136)^2 + (402.047 - 182.848)^2}$$
$$= \sqrt{(35.557)^2 + (219.199)^2}$$
$$= \sqrt{1264.3002 + 48048.2016}$$
$$= 222.0642$$

Lakukan hal yang sama hingga nilai pada cluster C1, C2, dan C3 telah terpenuhi seluruhnya. Setelah didapatkan nilai pada tiap cluster, selanjutnya tentukan jarak terpendek dengan membandingkan nilai pada tiap cluster yang ada.

Tabel 2. 3 Jarak Pusat Cluster Iterasi 1

No	Kota/Kab	Luas Lahan	Produksi	C1	C2	C3	Jarak Terpendek
1	Kab. Ponorogo	66.693	402.047	570.0825	15.99101	222.0642	15.99101
2	Kab. Trenggalek	31.136	182.848	792.1413	208.4753	0	0
3	Kab. Tulungagung	49.23	259.581	713.4876	129.701	78.83747	78.83747
4	Kab. Blitar	50.577	289.494	683.7493	100.1155	108.4035	100.1155
5	Kab. Kediri	51.083	281.392	691.6613	107.9328	100.5425	100.5425
6	Kab. Malang	65.597	464.498	508.8384	77.64213	283.7504	77.64213
7	Kab. Lumajang	72.552	387.168	583.8222	0	208.4753	0
8	Kab. Jember	162.619	964.001	0	583.8222	792.1413	0
9	Kab. Banyuwangi	113.609	706.419	262.2031	321.8802	530.0268	262.2031
10	Kab. Bondowoso	61.33	329.557	642.4785	58.69379	149.7839	58.69379
11	Kab. Situbondo	48.902	290.954	682.5861	99.07803	109.5561	99.07803
12	Kab. Probolinggo	59.13	311.258	660.8959	77.08747	131.426	77.08747

Pada tabel 2.3 diatas, C1 merupakan inisialisasi untuk menjelaskan tingkat produksi padi tinggi, sedangkan C2 merupakan tingkat produksi padi sedang, dan C3 merupakan tingkat produksi padi rendah.

3. Pengelompokan Data

Setelah nilai jarak telah didapatkan. Maka selanjutnya akan dilakukan perbandingan dan dipilih jarak terpendek antara data dengan pusat cluster. Berikut adalah tampilan data jarak terpendek yang didapatkan yaitu:

Tabel 2. 4 Pengelompokan Data Iterasi 1

No	C1	C2	C3
1		✓	
2			✓
3			✓
4		✓	
5			✓
6		✓	

7		✓	
8	✓		
9	✓		
10		✓	
11		✓	
12		✓	

ITERASI 2

1. Penentuan Pusat Awal Cluster

Pusat awal cluster pada iterasi ke 2 ini berbeda dengan pusat cluster pada iterasi ke 1.

- Untuk cluster C(1,1) nilai didapatkan dari Luas Lahan data ke 8 dan data ke 9. Berikut adalah cara perhitungannya:

  C(1,1) $= \frac{x_1+x_2}{n}$

  $= \frac{162.619+113.609}{2}$

  $= 138.114$

- Untuk cluster C(1,2) nilai didapatkan dari data Produksi ke 8 dan data ke 9. Berikut adalah cara perhitungannya:

  C(1,2) $= \frac{y_1+y_2}{n}$

  $= \frac{964.001+706.419}{2}$

  $= 835.21$

- Untuk cluster C(2,1) nilai didapatkan dari data Luas Lahan ke 1, 4, 6, 7, 10, 11, dan data ke 12. Berikut adalah cara perhitungannya:

  C(2,1) $= \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7}{n}$

  $= \frac{66.693+50.577+65.597+72.552+61.33+48.902+59.13}{7}$

  $= 60.683$

- Untuk cluster C(2,2) nilai didapatkan dari data Produksi ke 1, 4, 6, 7, 10, 11, dan data ke 12. Berikut adalah cara perhitungannya:

  C(2,2) $= \frac{y_1+y_2+y_3+y_4+y_5+y_6+y_7}{n}$

  $= \frac{402.047+289.494+ 464.498+387.168+329.557+290.954+311.258}{7}$

  $= 353.568$

- Untuk cluster C(3,1) nilai didapatkan dari data Luas Lahan ke 2, 3, dan data ke 5. Berikut adalah cara perhitungannya:

  C(3,1) $= \frac{x_1+x_2+x_3}{n}$

  $= \frac{31.136+ 49.23+ 51.083}{3}$

  $= 43.81633$

- Untuk cluster C(3,2) nilai didapatkan dari data Produksi ke 2, 3, dan data ke 12. Berikut adalah cara perhitungannya:

  C(3,2) $= \frac{y_1+y_2+y_3}{n}$

  $= \frac{182.848+259.581+311.258}{3}$

  $= 241.2737$

Maka didapatkan pusat awal cluster baru pada iterasi ke-2 yaitu:

Tabel 2. 5 Pusat Cluster Awal Iterasi 2

Cluster baru yang ke-1	138.114	835.21
Cluster baru yang ke-2	60.683	353.568
Cluster baru yang ke-3	43.81633333	241.2736667

2. Perhitungan Jarak Pusat Cluster

- Berikut adalah cara perhitungan untuk mendapatkan nilai pada cluster C(1,1) pada iterasi ke-2 :

$$C(1,1) = \sqrt{(66.693 - 138.114)^2 + (402.047 - 835.21)^2}$$
$$= \sqrt{(-71.421)^2 + (-433.163)^2}$$
$$= \sqrt{5100.9592 + 187630.1845}$$
$$= 439.0116$$

Lakukan hal yang sama hingga nilai pada cluster C1, C2, dan C3 telah terpenuhi seluruhnya. Setelah didapatkan nilai pada tiap cluster, selanjutnya tentukan jarak terpendek dengan membandingkan nilai pada tiap cluster yang ada.

Tabel 2. 6 Jarak Pusat Cluster Iterasi 2

No	Kota/Kab	Luas Lahan	Produksi	C1	C2	C3	Jarak Terpendek
1	Kab. Ponorogo	66.693	402.047	439.0116	48.85011	162.3928	48.85011301
2	Kab. Trenggalek	31.136	182.848	661.0752	173.258	59.78586	59.7858627
3	Kab. Tulungagung	49.23	259.581	582.451	94.68224	19.091	19.09099894
4	Kab. Blitar	50.577	289.494	552.6922	64.86608	48.69196	48.69196197
5	Kab. Kediri	51.083	281.392	560.6146	72.81164	40.77113	40.77113089
6	Kab. Malang	65.597	464.498	377.7381	111.0388	224.2844	111.0387873
7	Kab. Lumajang	72.552	387.168	452.8134	35.63472	148.6973	35.63471848
8	Kab. Jember	162.619	964.001	131.1016	618.8856	732.4267	131.1015511
9	Kab. Banyuwangi	113.609	706.419	131.1016	356.7982	470.3522	131.1015511
10	Kab. Bondowoso	61.33	329.557	511.4496	24.01972	90.00375	24.01971544
11	Kab. Situbondo	48.902	290.954	551.5192	63.71268	49.93996	49.93995921
12	Kab. Probolinggo	59.13	311.258	529.8718	42.33849	71.64018	42.33849205

3. Pengelompokan Data

Setelah nilai jarak telah didapatkan. Maka selanjutnya akan dilakukan perbandingan dan dipilih jarak terpendek antara data dengan pusat cluster. Berikut adalah tampilan data jarak terpendek yang didapatkan yaitu:

Tabel 2. 7 Pengelompokan Data Iterasi 2

No.	C1	C2	C3
1		✓	
2			✓
3			✓
4			✓
5			✓
6		✓	
7		✓	
8	✓		
9	✓		
10		✓	
11			✓
12		✓	

ITERASI 3

1. Penentuan Pusat Awal Cluster

Pusat awal cluster pada iterasi ke-3 ini berbeda dengan pusat cluster pada iterasi ke-1 dan iterasi ke-2.

- Untuk cluster C(1,1) nilai didapatkan dari Luas Lahan data ke 8 dan data ke 9. Berikut adalah cara perhitungannya:

$$C(1,1) = \frac{x_1+x_2}{n}$$
$$= \frac{162.619+113.609}{2}$$
$$= 138.114$$

Lakukan hal yang sama hingga nilai pusat awal pada cluster C1, C2, dan C3 telah terpenuhi. Maka didapatkan pusat awal cluster baru pada iterasi ke-3 yaitu:

Tabel 2. 8 Pusat Cluster Awal Iterasi 3

Cluster baru yang ke-1	138.114	835.21
Cluster baru yang ke-2	65.0604	378.9056
Cluster baru yang ke-3	46.1856	652.1345

2. Perhitungan Jarak Pusat Cluster

- Berikut adalah cara perhitungan untuk mendapatkan nilai pada cluster C(1,1) pada iterasi ke-3 :

$$C(1,1) = \sqrt{(66.693 - 138.114)^2 + (402.047 - 835.21)^2}$$
$$= \sqrt{(-71.421)^2 + (-433.163)^2}$$
$$= \sqrt{5100.9592 + 187630.1845}$$
$$= 439.0116$$

Lakukan hal yang sama hingga nilai pada cluster C1, C2, dan C3 telah terpenuhi seluruhnya. Setelah didapatkan nilai pada tiap cluster, selanjutnya tentukan jarak terpendek dengan membandingkan nilai pada tiap cluster yang ada.

Tabel 2. 9 Jarak Pusat Cluster Iterasi 3

No	Kota/Kab	Luas Lahan	Produksi	C1	C2	C3	Jarak Terpendek
1	Kab. Ponorogo	66.693	402.047	439.0116	23.19892	250.9269	23.19891758
2	Kab. Trenggalek	31.136	182.848	661.0752	198.971	469.5278	198.9709713
3	Kab. Tulungagung	49.23	259.581	582.451	120.3701	392.5653	120.3701031
4	Kab. Blitar	50.577	289.494	552.6922	90.57706	362.6671	90.57705609
5	Kab. Kediri	51.083	281.392	560.6146	98.51025	370.7748	98.51025274
6	Kab. Malang	65.597	464.498	377.7381	85.59408	188.6379	85.59408202
7	Kab. Lumajang	72.552	387.168	452.8134	11.15309	266.2751	11.15308587
8	Kab. Jember	162.619	964.001	131.1016	593.1731	332.8926	131.1015511
9	Kab. Banyuwangi	113.609	706.419	131.1016	331.0921	86.56051	86.56050952
10	Kab. Bondowoso	61.33	329.557	511.4496	49.48939	322.9328	49.48939489
11	Kab. Situbondo	48.902	290.954	551.5192	89.42359	361.1907	89.42358656
12	Kab. Probolinggo	59.13	311.258	529.8718	67.90705	341.1222	67.90704993

3. Pengelompokan Data

Setelah nilai jarak telah didapatkan. Maka selanjutnya akan dilakukan perbandingan dan dipilih jarak terpendek antara data dengan pusat cluster. Berikut adalah tampilan data jarak terpendek yang didapatkan yaitu:

Tabel 2. 10 Pengelompokkan Data Iterasi 3

No.	C1	C2	C3
1		✓	
2		✓	
3		✓	
4		✓	
5		✓	
6		✓	
7		✓	
8	✓		
9			✓
10		✓	
11		✓	
12		✓	

ITERASI 4

1. Penentuan pusat awal cluster

Pusat awal cluster pada iterasi ke-4 ini berbeda dengan pusat cluster pada iterasi ke-1, iterasi ke-2, dan iterasi ke-3

- Untuk cluster C(1,1) nilai didapatkan dari Luas Lahan data ke 1. Berikut adalah cara perhitungannya :

$$C(1,1) = \frac{x_1}{n}$$
$$= \frac{162.619}{1}$$
$$= 162.619$$

Lakukan hal yang sama hingga nilai pusat awal pada cluster C1, C2, dan C3 telah terpenuhi. Maka didapatkan pusat awal cluster baru pada iterasi ke-4 yaitu:

Tabel 2. 11 Pusat Awal Cluster Iterasi 4

Cluster baru yang ke-1	162.619	964.001
Cluster baru yang ke-2	55.623	378.9056
Cluster baru yang ke-3	113.609	706.419

2. Perhitungan Jarak Pusat Cluster

- Berikut adalah cara perhitungan untuk mendapatkan nilai pada cluster C(1,1) pada iterasi ke-4 :

$$C(1,1) = \sqrt{(66.693 - 162.619)^2 + (402.047 - 964.001)^2}$$
$$= \sqrt{(66530.381)^2 + (-561.954)^2}$$
$$= \sqrt{44262915596.0051 + 315792.2981}$$
$$= 570.082534$$

Lakukan hal yang sama hingga nilai pada cluster C1, C2, dan C3 telah terpenuhi seluruhnya. Setelah didapatkan nilai pada tiap cluster, selanjutnya tentukan jarak terpendek dengan membandingkan nilai pada tiap cluster yang ada.

Tabel 2. 12 Jarak Pusat Cluster Iterasi 4

No	Kota/Kab	Luas Lahan	Produksi	C1	C2	C3	Jarak Terpendek
1	Kab. Ponorogo	66.693	402.047	570.082534	25.65286132	307.9665979	25.65286132
2	Kab. Trenggalek	31.136	182.848	792.1412681	197.5808586	530.0267802	197.5808586
3	Kab. Tulungagung	49.23	259.581	713.4876325	119.4957347	451.4519397	119.4957347
4	Kab. Blitar	50.577	289.494	683.7492982	89.5538739	421.6627665	89.5538739
5	Kab. Kediri	51.083	281.392	691.6612799	97.61922856	429.601503	97.61922856
6	Kab. Malang	65.597	464.498	508.8383982	86.1715708	246.6392556	86.1715708
7	Kab. Lumajang	72.552	387.168	583.8222113	18.83768284	321.8802235	18.83768284
8	Kab. Jember	162.619	964.001	0	594.7980927	262.2031022	0
9	Kab. Banyuwangi	113.609	706.419	262.2031022	332.6069803	0	0
10	Kab. Bondowoso	61.33	329.557	642.4785216	49.67750166	380.470841	49.67750166
11	Kab. Situbondo	48.902	290.954	682.5861267	88.20802562	420.4737353	88.20802562
12	Kab. Probolinggo	59.13	311.258	660.8959049	67.73844429	398.8987056	67.73844429

3. Pengelompokan Data

Setelah nilai jarak telah didapatkan. Maka selanjutnya akan dilakukan perbandingan dan dipilih jarak terpendek antara data dengan pusat cluster. Berikut adalah tampilan data jarak terpendek yang didapatkan yaitu:

Tabel 2. 13 Pengelompokan Data Iterasi 4

No	C1	C2	C3
1		✓	
2		✓	
3		✓	
4		✓	
5		✓	
6		✓	
7		✓	
8	✓		
9			✓
10		✓	
11		✓	
12		✓	

Iterasi akan berhenti apabila Kelompok data terakhir sama dengan nilai kelompok data sebelumnya. Pada contoh kasus diatas, hasil pengelompokan iterasi ke-3 sama dengan hasil pengelompokan iterasi ke-4. Maka dari itu perhitungan berhenti dan telah selesai.

2.5.Ringkasan K-Means Clustering

Pada pembahasan buku ini, Anda telah mempelajari mengenai definisi K-Means Clustering, algoritma, dan cara penyelesaian masalah menggunakan K-Means Clustering, dengan studi kasus Sampel Padi di provinsi Jawa Timur.

Tahapan pemecahan masalah untuk studi kasus Sampel Padi di provinsi Jawa Timur menggunakan K-Means Clustering adalah dimulai dari iterasi-1 hingga berhenti pada iterasi ke-4, hingga dapat diambil kesimpulan bahwa terdapat 1 kabupaten yang memiliki tingkat produksi tinggi, 10 kabupaten yang memiliki tingkat produksi sedang, dan 1 kabupaten yang memiliki tingkat produksi rendah.

Tabel 2. 14 Kesimpulan

No	C1	C2	C3
1		Kab. Ponorogo	
2		Kab. Trenggalek	
3		Kab. Tulungagung	
4		Kab. Blitar	
5		Kab. Kediri	
6		Kab. Malang	
7		Kab. Lumajang	
8	Kab. Jember		
9			Kab. Banyuwangi
10		Kab. Bondowoso	
11		Kab. Situbondo	
12		Kab. Probolinggo	

2.6.Latihan K-Means Clustering

Terdapat sampel data Mahasiswa pada Kampus A yang mengambil Mata Kuliah Pembelajaran Mesin yaitu sebanyak 8 orang. Dataset yang diambil menggunakan nilai UTS, nilai UAS, dan Nilai Tugas. Data tersebut akan dibagi ke dalam 2 cluster. Inisialisasi cluster awal gunakan data No 1 dan No 5.

Tabel 2. 15 Latihan K-Means Clustering

No	NIM	Nama	Tugas	UTS	UAS
1	00160005	Annabelle	60	86	69
2	00160008	Caspeer	55	60	55
3	00160011	Chucky Hakim	85	69	71
4	00160004	Issabelle	80	84	73
5	00160021	Peter Cs	75	71	77
6	00160030	Sabrina	60	50	84
7	00160034	Sinchan Wakican	50	86	65
8	00160037	Susi Suzanna	65	58	91

Jawablah pertanyaan dibawah ini:

a) Jelaskan titik pusat cluster baru pada iterasi ke 2 dan iterasi ke 3!
b) Siapa sajakah yang masuk dalam cluster 1 yang mendapatkan nilai terbaik dan cluster 2 yang mendapatkan nilai kurang baik?
c) Berikan kesimpulan dari perhitungan yang telah dilakukan!

BAB 3. FUZZY C-MEANS CLUSTERING

3.1. Pendahuluan

Pertama kali Fuzzy C-Means diperkenalkan oleh (Dunn, 1973) kemudian dikembangkan oleh (Bezdek, 1981). *Fuzzy c-means* merupakan teknik clustering data yang tiap-tiap datanya ditentukan oleh nilai keanggotaan dimana nilai derajat keanggotaan sudah ditentukan secara random diawal serta memiliki rentan nilai 0 hingga 1 (Munir, 2005). *Fuzzy clustering* adalah salah satu teknik untuk menentukan cluster optimal dalam suatu ruang vektor yang didasarkan pada bentuk normal Euclidian untuk jarak antar vector (Zimmermann, 2001).

Konsep awal Fuzzy C-Means Clustering diawali dengan cara menentukan titik pusat cluster awal yang dimana itu menandai rata – rata untuk tiap cluster terlebih dahulu. Pada kondisi ini nilai awal atau pusat cluster belum akurat. Untuk setiap titik data memiliki yang namanya derajat keanggotaan untuk tiap – tiap clusternya.

Dengan memperbaiki pusat cluster dan derajat keanggotaan tiap-tiap titik data secara berulang dengan pusat cluster yang telah ditentukan diawal lalu menjadi sebuah rujukan untuk cluster-cluster yang berikutnya, sampai kesimpulan dari masing-masing cluster kurang dari sama dengan eror toleransi terkecil yang telah ditentukan (biasanya eror toleransi kurang dari sama dengan 0.1). Perulangan pada setiap iterasi didasarkan pada nilai minimasi dari nilai fungsi objektif yang menggambarkan jarak titik data yang diberikan ke pusat cluster yang terbobot oleh derajat keanggotaan titik data tersebut. (Tanjung, 2016)

3.1.1 Perbedaan Fuzzy C-Means (FCM) dengan K-Means Clustering

Fuzzy c-means merupakan perkembangan dari metode *k-means* dengan memperhitungkan bahwa data dapat tergabung ke dalam beberapa cluster dengan tingkat keanggotaan yang berbeda-beda dan cluster di setiap iterasi yang saling berkaitan. Algoritma yang digunakan dalam *fuzzy c-means* sama dengan algoritma yang digunakan oleh *k-means* (Nur Indah Selviana, Mustakim, 2016).

3.1.2 Kelebihan dan Kekurangan FCM (Kusumadewi, 2010)

Semua metode pasti memiliki beberapa kelebihan serta kekurangannya masing – masing, Fuzzy C Means Clustering juga memiliki beberapa kelebihan dan kekurangan, diantaranya :

A. Kelebihan Fuzzy C Means Clustering :

1. Fuzzy C Means Clustering merupakan metode yang kemungkinan kecil mendapatkan kegagalan untuk konvergen.
2. Merupakan metode yang mudah untuk diimplementasikan.
3. Fuzzy C Means Clustering memiliki kemampuan untuk pengelompokkan data yang besar.
4. Selalu konvergen atau mampu melakukan klusterisasi.
5. Tidak membutuhkan operasi matematis yang rumit.
6. Beban komputasi relatif ringan , sehingga konvergensi dapat tercapai. (relatif tergantung pada banyaknya data dan cluster yang diinginkan)

B. Kekurangan Fuzzy C Means Clustering :

1. Jumlah cluster harus ditentukan diawal
2. Pusat cluster yang diberikan diawal bisa mempengaruhi hasil akhir. (saling berkesinambungan antara pusat cluster satu dengan yang lain)
3. Solusi cluster yang dihasilkan hanya bersifat local optimal sehingga belum dapat dipastikan sudah merupakan hasil yang optimal atau belum.

3.2. Persamaan Matematis *Fuzzy C-Means Clustering* (Ahmadi, A. dan Hartati, S. , 2013)

Algoritma fuzzy c-means clustering ini membagi beberapa data kedalam setiap elemen data yang kemudian akan dimasukkan kedalam cluster yang dipengaruhi oleh kriteria yang akan diberikan. Berikut satu kumpulan data berhingga X = {x_1, x_2,…x_n} dan pusat data.

$J_m(X,U,V) = \sum_{j=1}^{n}\sum_{i=1}^{c}(\mu ij)\text{^}m d\text{^}2(Xj,Vi)$……(3.1)

Dimana μ_{ij} adalah derajat keanggotaan dari X_j dan pusat cluster adalah sebuah bagian dari keanggotaan matriks [μ_{ij}]. d^2 adalah akar dari *Eucliden distance* dan m adalah parameter fuzzy yang rata – rata derajat keanggotaannya dari setiap data tidak lebih besar dari 1,0.

Salah satu komponen perhitungan adalah nilai Random, yang didapat dengan persamaan berikut.

$Q_j = \sum_{k=1}^{c}\mu ik$……(3.2)

Bilangan random $\mu_{ik,}$ i = 1,2,3…,n; k = 1,2,3…,c; dengan j = 1,2,3…m; sebagai elemen – elemen matriks partisi awal U. Kemudian hitung jumlah setiap kolom (atribut) menggunakan persamaan diatas (3.2).

Terdapat komponen perhitungan pusat cluster, yang didalamnya terdapat perhitungan mencari pusat cluster pada data. Persamaan pusat cluster sebagai berikut :

$V_{kj} = \frac{\sum_{i=1}^{n}((\mu ik)\text{^}w * Xij)}{\sum_{i=1}^{n}((\mu ik)\text{^}w}$……(3.3)

Dimana X_{ij} merupakan variabel Fuzzy yang digunakan & *w* adalah bobot. Dengan menggunakan hitung pusat cluster ke-k : V_{kj}, dengan k = 1,2,3,…,c; dan j = 1,2,3,…,m.

Selanjutnya dalam mencari tiap – tiap cluster harus menentukan fungsi objektif terlebih dahulu, dengan persamaan sebagai berikut :

$P_t = \sum_{i=1}^{n}\sum_{k=1}^{c}([\sum_{j=1}^{m}(Xij - Vkj)\text{^}2](\mu ik)\text{^}w)$……(3.4)

Kemudian terdapat suatu komponen untuk perubahan matriks partisi, perubahan ini dilakukan jika sudah melalui perhitungan fungsi objektif. Berikut persamaan perubahan matriks :

$\mu_{ik} = \frac{[\sum_{j=1}^{m}(Xij - Vkj)\text{^}2]\text{^}-1/w-1}{\sum_{k=1}^{c}[\sum_{j=1}^{m}(Xij - Vkj)\text{^}2]\text{^}-1/w-1}$……..(3.5)

(Dengan i = 1,2,3…,n; k = 1,2,3…,c;)

Setelah nilai matriks partisi di update, lalu kita harus menentukan apakah iterasi masih terus berlanjut atau berhenti. Untuk mengetahuinya yaitu dengan menghitung kesimpulan dari setiap iterasi dengan melakukan pengurangan antara nilai fungsi objektif pada iterasi sebelumnya dan pada iterasi sesudahnya.

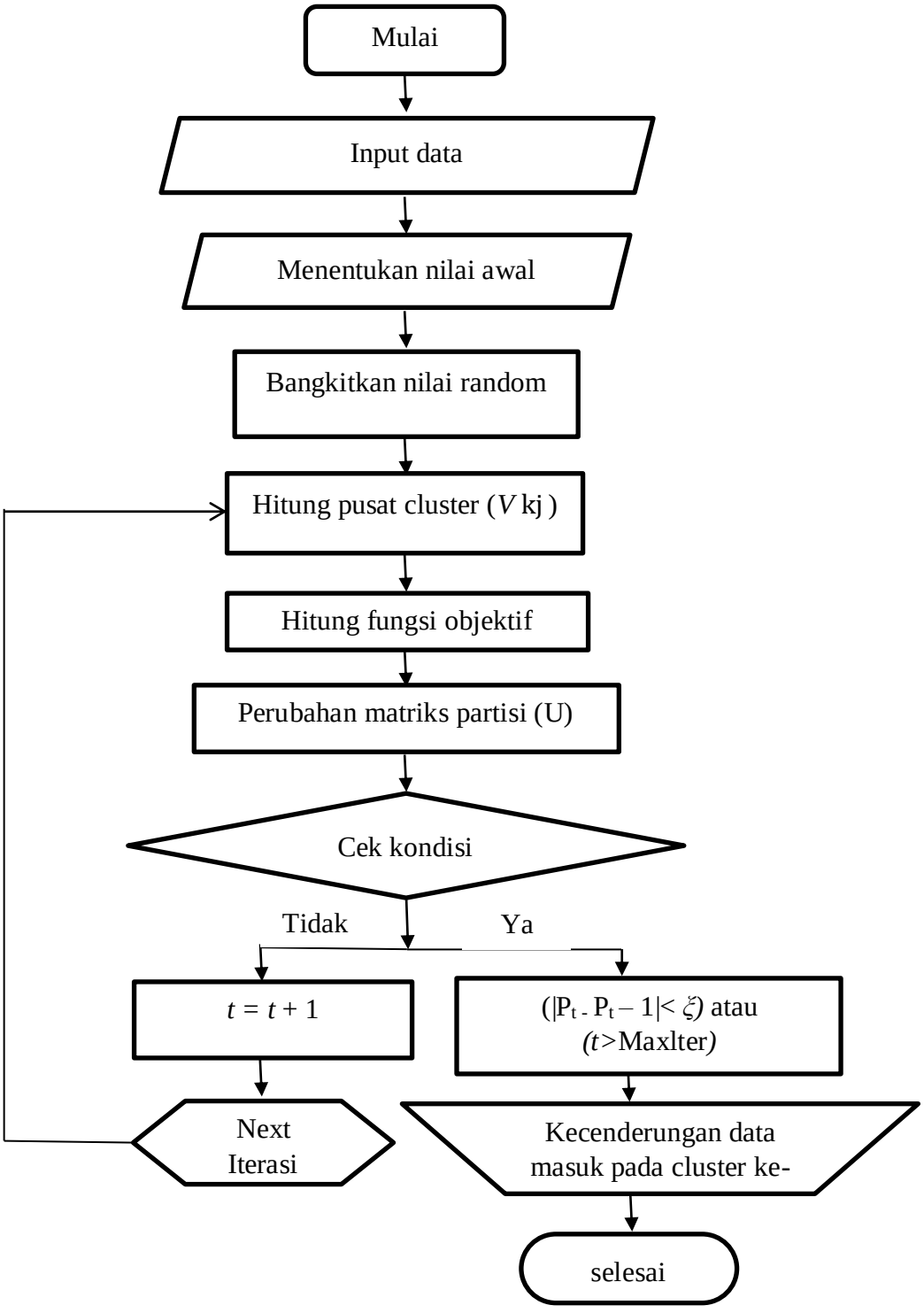
Kesimpulan = $P_{sesudah} - P_{sebelum} <= \varepsilon$ (error yang diharapkan)……..(3.6)

3.3. Prosedur / Algoritma Fuzzy C-Means Clustering

Proses jalannya Algoritma Fuzzy C-Means Clustering adalah sebagai berikut: (Kusumadewi, 2010)

1. Siapkan Input Data
   Data merupakan matrix berukuran n x m dimana *n* adalah jumlah sampel data dan *m* adalah atribut data, kemudian X_{ij} sama dengan data sample ke-i (i = 1,2,3…n), atribut ke-j (j = 1,2,3…n).
2. Tentukan komponen dibawah ini :
   a. Jumlah cluster = c;
   b. Pangkat / Pembobotan = w;
   c. Maksimum iterasi = Maxlter;
   d. Error terkecil yang diharapkan = ε
   e. Fungsi objektif awal = $P_o = 0$;
   f. Iterasi awal = t = 1;
3. Bangkitkan nilai random
   Nilai random didapatkan dengan penjumlahan pada persamaan 3.2 diatas, yang dimana μ_{ik} adalah derajat keanggotaan yang merujuk kepada sebarapa besarkah kemungkinan suatu data bisa menjadi anggota kedalam suatu cluster. Nilai keanggotaan terletak pada interval 0 sampai 1. Pada posisi ini, mengakibatkan posisi awal matriks patrisi belum akurat begitu pun clusternya.
4. Hitung pusat cluster ke-k
   Perhitungan pusat cluster ke-k ini adalah pusat cluster yang didapatkan dari persamaan rumus 3.3 yang dimana pada persamaan tersebut akan mencari nilai pusat cluster, setelah nilai pusat cluster diketahui maka akan didapatkan nilai untuk mencari nilai fungsi objektif.
5. Hitung Fungsi Objektif
   Perhitungan ini didapatkan dengan cara mengetahui nilai pusat cluster V_{ij} terlebih dahulu, kemudian nanti akan dikurang dengan variable fuzzy X_{ij}, kemudian hasil dari masing – masing dikuadratkan selanjutnya hasil dari kuadrat tersebut dijumlahkan untuk dikali dengan kuadrat keanggotaan μ_{ik} untuk tiap cluster. Selanjutnya jumlahkan semua nilai cluster untuk mendapatkan hasil nilai fungsi objektif, atau dapat dilihat pada persamaan 3.4 diatas.
6. Hitung Perubahan matriks
   Untuk dapat mencari perubahan matriks partisi μ_{ik} dikurang dengan fuzzy V_{ij} dilakukan kembali pada pusat cluster lalu dikuadratkan. Kemudian dijumlahkan lalu dipangkatkan dengan$-1/(w-1)$, hasilnya setiap data dipangkatkan dengan -1. Setelah itu lakukan normalisasi semua data derajat keanggotaan, baru jumlahkan derajat keanggotaan baru k=1. Kemudian hasilnya akan dibagi dengan derajat keanggotaan baru, proses ini dilakukan agar derajat keanggotaan yang baru memiliki rentang nilai antara 0 dan tidak lebih dari 1. Dapat dilihat dipersamaan 3.5 diatas.
7. Mengecek kondisi pada setiap iterasi.
   Mengecek kondisi apakah iterasi masih terus berlanjut atau berhenti. Untuk mengetahuinya yaitu dengan menghitung kesimpulan dari setiap iterasi dengan melakukan pengurangan antara nilai fungsi objektif pada iterasi sebelumnya dengan pada iterasi sesudahnya. Dari pengurangan tersebut, jika hasilnya masih lebih dari error yang diharapkan maka iterasi terus berlanjut, tetapi jika hasilnya sudah kurang dari sama dengan error yang diharapkan maka iterasi berhenti. Dapat dilihat pada persamaan 3.6 diatas.

Berikut adalah *flowchart fuzzy c-means* (Muhammad Suhaili, Dana Indra Sensuse, 2015) :



Gambar 3. 1 Flowchart Fuzzy C-Means Clustering

3.4. Sample Case (Fakhriansyah, 2018)

Terdapat suatu contoh kasus Data Kecelakaan Lalu Lintas dengan menerapkan algoritma FCM.

Tabel 3. 1 Data Kecelakaan Lalu Lintas

Data	Nama Jalan	Jumlah Kecelakaan	Jumlah Kendaraan	Jumlah Korban
1	Ring Road Utara	7	14	14
2	Ring Road Selatan	0	0	0
3	Magelan Km 5-10	9	21	13
4	Kaliurang Km 10-15	3	6	9
5	Jogja-Solo Km 10-15	5	10	8
6	Adi Sucipto Km 5-10	4	8	5
7	Godean Km 5-10	3	6	7
8	Wates Km 5-10	6	13	14
9	Palagan	0	0	0
10	Affandi	4	9	5

Berdasarkan data dari tabel 3.1, terdapat tiga atribut (Jumlah Kecelakaan, Jumlah Kendaraan, dan Jumah Korban) yang akan diproses menggunakan Algoritma Fuzzy C-Means Clustering. Dari ketiga atribut tersebut, telah ditentukan beberapa komponen perhitungan yang ditunjukkan dalam tabel dibawah ini.

NILAI AWAL	
Jumlah Cluster (c)	3
Pangkat (w)	2
Maksimum Iterasi	9
Error terkecil yang diharapkan (ε)	X
Fungsi Objektif Awal	0
Iterasi Awal	1

Cluster 1	Tidak Rawan
Cluster 2	Rawan
Cluster 3	Sangat Rawan

ITERASI 1

1. Bangkitkan bilangan acak sebagai matriks partisi awal.

Tabel 3. 2 Bangkitkan Bilangan Acak

μ_{ik}				X_{ij}		
I	K_1	K_2	K_3	Jumlah Kecelakaan	Jumlah Kendaraan	Jumlah Korban
1	0.31	0.15	0.54	7	14	14
2	0.32	0.16	0.52	0	0	0
3	0.11	0.33	0.56	9	21	13
4	0.65	0.22	0.13	3	6	9
5	0.44	0.21	0.35	5	10	8
6	0.14	0.3	0.56	4	8	5
7	0.45	0.41	0.14	3	6	7
8	0.34	0.32	0.34	6	13	14
9	0.66	0.11	0.23	0	0	0
10	0.32	0.33	0.35	4	9	5

Pada tabel diatas, bangkitkan matriks U_{ik} dengan komponen i = banyaknya data; k = banyak cluster (memiliki nilai acak dari 0 – 1).

2. **Menentukan pusat cluster dari tiga cluster yang akan dibentuk.** Perhitungan pusat cluster seperti yang ditunjukkan pada tabel 3.3 s.d. tabel 3.5 sebagai berikut.

Tabel 3. 3 Perhitungan Pusat Cluster 1 (Iterasi 1)

μ_{i1}	μ_{i1}^w	$\mu_{i1}^w * X_i1$	$\mu_{i1}1^w * X_i2$	$\mu_{i1}1^w * X_i3$
0.31	0.0961	0.6727	1.3454	1.3454
0.32	0.1024	0	0	0
0.11	0.0121	0.1089	0.2541	0.1573
0.65	0.4225	1.2675	2.535	3.8025
0.44	0.1936	0.968	1.936	1.5488
0.14	0.0196	0.0784	0.1568	0.098
0.45	0.2025	0.6075	1.215	1.4175
0.34	0.1156	0.6936	1.5028	1.6184
0.66	0.4356	0	0	0
0.32	0.1024	0.4096	0.9216	0.512
Jumlah	1.7024	4.8062	9.8667	10.4999

Tabel 3. 4 Perhitungan Pusat Cluster 2 (Iterasi 1)

μ_{i2}	μ_{i2}^w	$\mu_{i2}^w * X_i1$	$\mu_{i2}^w * X_i2$	$\mu_{i2}^w * X_i3$
0.15	0.0225	0.1575	0.315	0.315
0.16	0.0256	0	0	0
0.33	0.1089	0.9801	2.2869	1.4157
0.22	0.0484	0.1452	0.2904	0.4356

0.21	0.0441	0.2205	0.441	0.3528
0.3	0.09	0.36	0.72	0.45
0.41	0.1681	0.5043	1.0086	1.1767
0.32	0.1024	0.6144	1.3312	1.4336
0.11	0.0121	0	0	0
0.33	0.1089	0.4356	0.9801	0.5445
Jumlah	0.731	3.4176	7.3732	6.1239

Tabel 3. 5 Perhitungan Pusat Cluster 3 (Iterasi 1)

μ_{i3}	μ_{i3}^{w}	$\mu_{i3}^{w} * X_i1$	$\mu_{i3}^{w} * X_i2$	$\mu_{i3}^{w} * X_i3$
0.54	0.2916	2.0412	4.0824	4.0824
0.52	0.2704	0	0	0
0.56	0.3136	2.8224	6.5856	4.0768
0.13	0.0169	0.0507	0.1014	0.1521
0.35	0.1225	0.6125	1.225	0.98
0.56	0.3136	1.2544	2.5088	1.568
0.14	0.0196	0.0588	0.1176	0.1372
0.32	0.1156	0.6936	1.5028	1.6184
0.11	0.0529	0	0	0
0.33	0.1225	0.49	1.1025	0.6125
Jumlah	1.6392	8.0236	17.2261	13.2274

Pada tabel perhitungan cluster 1 sampai 3 ditunjukkan bagaimana cara mendapatkan pusat cluster dengan menggunakan algoritma ke-d. Dimana angka acak dipangkatkan dua, kemudian hasil dari angka acak yang telah diberi pembobot dua dikalikan dengan Data Kecelakaan Lalu Lintas. Masing-masing kolom kemudian dijumlahkan untuk menentukan pusat cluster yang baru dari masing-masing iterasi. Pusat cluster yang baru ditunjukkan pada tabel 3.4.6 berikut ini.

3. Perhitungan untuk menentukan pusat cluster baru dengan menggunakan rumus $(\mu_{ik}2^{w} * X_{ij}) / (\mu_{ik}2^{w})$

Tabel 3. 6 Pusat Cluster Baru (Iterasi 1)

V			
1	2.823190789	5.795758929	6.167704417
2	4.675239398	10.08645691	8.377428181
3	4.894826745	10.50884578	8.069424109

4. Perhitungan fungsi objektif seperti pada tabel di bawah ini.

Tabel 3. 7 Perhitungan Fungsi Objektif (Iterasi 1)

Kuadrat Derajat Keanggotaan data ke-i			$[_{j=1}\sum((X_{ij}-V_{ij})^2) * \mu_{i1}{}^2]$
(K1)^2	(K2)^2	(K3)^2	L1
0.0961	0.0225	0.2916	14.04022546
0.1024	0.0256	0.2704	8.15122489
0.0121	0.1089	0.3136	3.823626396
0.4225	0.0484	0.0169	3.420084341
0.1936	0.0441	0.1225	4.989352416
0.0196	0.09	0.3136	0.149099008
0.2025	0.1681	0.0196	0.155052599
0.1156	0.1024	0.1156	14.25787977
0.4356	0.0121	0.0529	34.67454651
0.1024	0.1089	0.1225	1.332794815

Tabel 3. 8 Perhitungan Fungsi Objektif (Iterasi 1) (Sambungan)

$[_{j=1}\sum((X_{ij}-V_{ij})^2) * \mu_{i2}{}^2]$	$[_{j=1}\sum((X_{ij}-V_{ij})^2) * \mu_{i3}{}^2]$	L1 + L2 + L3
L2	L3	
1.177507018	15.10244299	30.32017546
4.96065995	53.94771327	67.05959811
17.33439134	47.42489476	68.58291249
0.962828594	0.418883909	4.801796845
0.011262976	0.03366363	5.034279022
1.459464461	5.179538777	6.788102246
3.59782169	0.491249041	4.24412333
4.286160677	4.924442975	23.46848342
2.34468693	10.55411994	47.57335338
1.420421683	1.531090119	4.284306618
Fungsi Objektif		262.1571309

5. Perbaikan matriks partisi U

Tabel 3. 9 Perbaikan Matriks Partisi U (Iterasi 1)

$[_{j=1}\sum((X_{ij}-V_{1j})^2)^{-1}]$	$[_{j=1}\sum((X_{ij}-V_{2j})^2)^{-1}]$	$[_{j=1}\sum((X_{ij}-V_{1j})^2)^{-1}]$	LT
L1	L2	L3	L1 + L2 + L3
0.006844619	0.019108166	0.019308134	0.04526092
0.012562529	0.005160604	0.005012261	0.022735393
0.003164535	0.006282309	0.006612561	0.016059405
0.123534965	0.050268553	0.040345307	0.214148825

0.038802631	3.915483857	3.638942035	7.593228523
0.131456274	0.061666455	0.060545932	0.253668661
1.306008417	0.046722716	0.039898297	1.39262943
0.008107797	0.023890845	0.023474736	0.055473379
0.012562529	0.005160604	0.005012261	0.022735393
0.076831031	0.076667374	0.080008354	0.233506759

Tabel 3. 10 Perbaikan Matriks Partisi U (Iterasi 1)(Sambungan)

L1 / LT	L2 / LT	L3 / LT
0.151225812	0.422178035	0.426596153
0.552553847	0.226985457	0.220460695
0.197051831	0.391191886	0.411756282
0.576865015	0.234736533	0.188398452
0.005110162	0.515654684	0.479235153
0.518220397	0.243098438	0.238681165
0.937800386	0.033549999	0.028649615
0.146156545	0.430672253	0.423171202
0.552553847	0.226985457	0.220460695
0.329031295	0.328330427	0.342638279

Dari perhitungan matriks partisi U seperti pada tabel 3.7 s.d. tabel 3.10, maka di dapatkan derajat keanggotaan baru untuk melanjutkan ke iterasi selanjutnya, yaitu :

Tabel 3. 11 Pembaharuan Matriks Partisi

μ_{ik}			
I	K_1	K_2	K_3
1	0.151225812	0.422178035	0.426596153
2	0.552553847	0.226985457	0.220460695
3	0.197051831	0.391191886	0.411756282
4	0.576865015	0.234736533	0.188398452
5	0.005110162	0.515654684	0.479235153
6	0.518220397	0.243098438	0.238681165
7	0.937800386	0.033549999	0.028649615
8	0.146156545	0.430672253	0.423171202
9	0.552553847	0.226985457	0.220460695
10	0.329031295	0.328330427	0.342638279

6. Kesimpulan setiap iterasi (Cek kondisi berhenti)

$|P_1 - P_0| = |262.1571309 - 0| = 262.1571309 \gg \varepsilon$ (0.1) dan Iterasi = 1 <= MaxIter (9) maka proses dilanjutkan ke iterasi kedua.

ITERASI Ke – 2 (Lakukan penyelesaian sama dengan iterasi sebelumnya)

1. **Bangkitkan bilangan acak sebagai matriks partisi awal.** Matriks partisi awal menggunakan pada iterasi kedua ini, menggunakan derajat keanggotaan baru yang ada pada iterasi pertama.

Tabel 3. 12 Matriks Partisi Awal

μ_{ik}			
i	K_1	K_2	K_3
1	0.151225812	0.422178035	0.426596153
2	0.552553847	0.226985457	0.220460695
3	0.197051831	0.391191886	0.411756282
4	0.576865015	0.234736533	0.188398452
5	0.005110162	0.515654684	0.479235153
6	0.518220397	0.243098438	0.238681165
7	0.937800386	0.033549999	0.028649615
8	0.146156545	0.430672253	0.423171202
9	0.552553847	0.226985457	0.220460695
10	0.329031295	0.328330427	0.342638279

2. **Menentukan pusat cluster dari tiga cluster yang akan dibentuk.** Perhitungan pusat cluster seperti yang ditunjukkan pada tabel 3.13 – tabel 3.15 sebagai berikut :

Tabel 3. 13 Perhitungan Pusat Cluster 1 (Iterasi 2)

μ_{i1}	μ_{i1}^w	$\mu_{i1}^w * X_i1$	$\mu_{i1}1^w * X_i2$	$\mu_{i1}1^w * X_i3$
0.151225812	0.02286925	0.160084724	0.320169448	0.320169448
0.552553847	0.30531575	0	0	0
0.197051831	0.03882942	0.349464818	0.815417908	0.504782514
0.576865015	0.33277325	0.998319736	1.996639472	2.994959207
0.005110162	2.6114E-05	0.000130569	0.000261138	0.00020891
0.518220397	0.26855238	1.074209518	2.148419035	1.342761897
0.937800386	0.87946956	2.638408694	5.276817388	6.156286953
0.146156545	0.02136174	0.128170415	0.277702565	0.299064301
0.552553847	0.30531575	0	0	0
0.329031295	0.10826159	0.433046372	0.974354337	0.541307965
Jumlah	2.28277481	5.781834845	11.80978129	12.1595412

Tabel 3. 14 Perhitungan Pusat Cluster 2 (Iterasi 2)

μ_{i2}	μ_{i2}^w	$\mu_{i2}^w * X_i1$	$\mu_{i2}^w * X_i2$	$\mu_{i2}^w * X_i3$
0.422178035	0.17823429	1.247640051	2.495280102	2.495280102
0.226985457	0.0515224	0	0	0
0.391191886	0.15303109	1.377279828	3.213652932	1.989404196

0.234736533	0.05510124	0.16530372	0.33060744	0.495911159
0.515654684	0.26589975	1.329498768	2.658997535	2.127198028
0.243098438	0.05909685	0.236387403	0.472774806	0.295484254
0.033549999	0.0011256	0.003376807	0.006753615	0.007879217
0.430672253	0.18547859	1.112871537	2.411221664	2.596700253
0.226985457	0.0515224	0	0	0
0.328330427	0.10780087	0.431203476	0.970207821	0.539004345
Jumlah	1.10881309	5.90356159	12.55949591	10.54686155

Tabel 3. 15 Perhitungan Pusat Cluster 3 (Iterasi 2)

μ_{i3}	μ_{i3}^{w}	$\mu_{i3}^{w} * X_i1$	$\mu_{i3}^{w} * X_i2$	$\mu_{i3}^{w} * X_i3$
0.426596153	0.18198428	1.273889944	2.547779888	2.547779888
0.220460695	0.04860292	0	0	0
0.411756282	0.16954324	1.525889125	3.560407957	2.204062069
0.188398452	0.03549398	0.10648193	0.212963861	0.319445791
0.479235153	0.22966633	1.14833166	2.296663321	1.837330657
0.238681165	0.0569687	0.227874794	0.455749588	0.284843493
0.028649615	0.0008208	0.002462401	0.004924803	0.005745603
0.423171202	0.17907387	1.074443195	2.327960256	2.507034122
0.220460695	0.04860292	0	0	0
0.342638279	0.11740099	0.46960396	1.056608909	0.58700495
Jumlah	1.06815801	5.82897701	12.46305858	10.29324657

Pada tabel perhitungan pusat cluster 1 sampai 3 ditunjukkan bagaimana cara mendapatkan pusat cluster dengan menggunakan algoritma ke-d. Dimana angka acak dipangkatkan dua, kemudian hasil dari angka acak yang telah diberi pembobot dua dikalikan dengan Data Kecelakaan Lalu Lintas.

3. Perhitungan untuk menentukan pusat cluster baru dengan menggunakan rumus $(\mu_{ik}2^{w} * X_{ij}) / (\mu_{ik}2^{w})$

Tabel 3. 16 Pusat Cluster Baru (Iterasi 2)

V			
1	2.53280999	5.173432454	5.326649453
2	5.324217097	11.32697302	9.511848018
3	5.457036258	11.66780422	9.636445582

4. Perhitungan fungsi objektif seperti pada tabel 3.16 sebagai berikut.

Tabel 3. 17 Perhitungan Fungsi Objektif (Iterasi 2)

Kuadrat Derajat Keanggotaan data ke-i			$[_{j=1}\sum(X_{ij} - V_{ij})^2) * \mu_{i1}{}^2]$
(K1)^2	(K2)^2	(K3)^2	L1
0.022869246	0.178234293	0.181984278	3.958462793
0.305315754	0.051522398	0.048602918	18.79301641
0.038829424	0.153031092	0.169543236	13.63631504
0.332773245	0.05510124	0.035493977	4.79026576
2.61138E-05	0.265899754	0.229666332	0.000953925
0.268552379	0.059096851	0.056968699	2.752347812
0.879469565	0.001125602	0.0008208	3.255429122
0.021361736	0.18547859	0.179073866	3.172294146
0.305315754	0.051522398	0.048602918	18.79301641
0.108261593	0.107800869	0.11740099	1.829833718

Tabel 3. 18 Perhitungan Fungsi Objektif (Iterasi 2)(sambungan)

$[_{j=1}\sum((X_{ij} - V_{ij})^2) * \mu_{i2}{}^2]$	$[_{j=1}\sum((X_{ij} - V_{ij})^2) * \mu_{i3}{}^2]$	L1 + L2 + L3
L2	L3	
5.364287174	4.888185308	14.21093528
12.73236224	12.57736515	44.1027438
18.24835096	18.81183394	50.69649995
1.875680101	1.368864076	8.034809937
1.103924989	1.301842293	2.406721207
1.960778724	2.111964677	6.825091213
0.045123158	0.037027848	3.337580128
4.340052468	3.780278385	11.292625
12.73236224	12.57736515	44.1027438
2.967232601	3.608526247	8.405592567
Fungsi Objektif		193.4153429

5. Perbaikan matriks partisi U

Tabel 3. 19 Perbaikan Matriks Partisi U (Iterasi 2)

$[_{j=1}\sum((X_{ij} - V_{1j})^2)^{-1}]$	$[_{j=1}\sum((X_{ij} - V_{2j})^2)^{-1}]$	$[_{j=1}\sum((X_{ij} - V_{1j})^2)^{-1}]$	LT
L1	L2	L3	L1 + L2 + L3
0.005777305	0.033226091	0.037229415	0.07623281
0.016246235	0.00404657	0.003864316	0.024157121
0.002847501	0.008386023	0.009012584	0.020246108
0.06946864	0.029376672	0.025929511	0.124774823
0.027375072	0.240867592	0.176416401	0.444659065

0.097572109	0.03013948	0.026974267	0.154685855
0.270154727	0.024945116	0.022167111	0.317266954
0.006733845	0.042736486	0.04737055	0.09684088
0.016246235	0.00404657	0.003864316	0.024157121
0.059164716	0.036330441	0.032534332	0.128029489

Tabel 3. 20 Perbaikan Matriks Partisi U (Iterasi 2)(Sambungan)

L1 / LT	L2 / LT	L3 / LT
0.075785017	0.435850268	0.488364714
0.672523621	0.167510449	0.15996593
0.140644374	0.414204196	0.44515143
0.556752056	0.2354375	0.207810444
0.061564183	0.541690501	0.396745316
0.630775894	0.194843154	0.174380952
0.851506037	0.078625006	0.069868957
0.069535144	0.441306249	0.489158606
0.672523621	0.167510449	0.15996593
0.462117882	0.283766195	0.254115922

Dari perhitungan matriks partisi U seperti pada tabel diatas, maka di dapatkan derajat keanggotaan baru untuk melanjutkan ke iterasi selanjutnya, yaitu :

Tabel 3. 21 Pembaharuan Matriks Partisi

μ_{ik}			
I	K_1	K_2	K_3
1	0.075785017	0.435850268	0.488364714
2	0.672523621	0.167510449	0.15996593
3	0.140644374	0.414204196	0.44515143
4	0.556752056	0.2354375	0.207810444
5	0.061564183	0.541690501	0.396745316
6	0.630775894	0.194843154	0.174380952
7	0.851506037	0.078625006	0.069868957
8	0.069535144	0.441306249	0.489158606
9	0.672523621	0.167510449	0.15996593
10	0.462117882	0.283766195	0.254115922

6. Kesimpulan setiap iterasi (Cek kondisi berhenti)

$|P_2 - P_1|$ = | 193.4153429 – 262.1571309 | = 68.74178805 $\gg \varepsilon$ (0.1) dan Iterasi = 2 < = MaxIter (9) maka proses dilanjutkan ke iterasi ketiga.

Catatan Penting!

- **Apabila nilai pada setiap kesimpulan belum mencapai kurang dari sama dengan error toleransi, maka iterasi masih terus berjalan sampai nilai tersebut kurang dari sama dengan error toleransi.**
- **Pada contoh kasus diatas, iterasi akan berhenti pada iterasi ke-sembilan. Dimana nilai dari kesimpulan di iterasi ke-sembilan yaitu 0.069280658. Maka iterasi berhenti.**

ITERASI Ke – 9 (Lakukan penyelesaian sama dengan iterasi sebelumnya)

1. **Bangkitkan bilangan acak sebagai matriks partisi awal.** Matriks partisi awal menggunakan pada iterasi ketiga ini, menggunakan derajat keanggotaan baru yang ada pada iterasi kedua.

Tabel 3. 22 Matriks Partisi Awal

μ_{ik}			
i	K_1	K_2	K_3
1	0.006071709	0.026404754	0.967523537
2	0.999929335	5.64333E-05	1.42312E-05
3	0.040164918	0.115683759	0.844151323
4	0.066198555	0.871569506	0.062231939
5	0.035055395	0.869662127	0.095282478
6	0.027534586	0.952660909	0.019804505
7	0.043255924	0.930928682	0.025815394
8	0.018851498	0.089231015	0.891917487
9	0.999929335	5.64333E-05	1.42312E-05
10	0.032946334	0.936455589	0.030598077

2. **Menentukan pusat cluster dari tiga cluster yang akan dibentuk.** Perhitungan pusat cluster seperti yang ditunjukkan pada tabel sebagai berikut.

Tabel 3. 23 Perhitungan Pusat Cluster 1 (Iterasi 9)

μ_{i1}	μ_{i1}^{w}	$\mu_{i1}^{w} * X_i1$	$\mu_{i1}1^{w} * X_i2$	$\mu_{i1}1^{w} * X_i3$
0.006071709	3.6866E-05	0.00025806	0.000516119	0.000516119
0.999929335	0.99985868	0	0	0
0.040164918	0.00161322	0.014518986	0.033877633	0.020971868
0.066198555	0.00438225	0.013146746	0.026293492	0.039440239
0.035055395	0.00122888	0.006144404	0.012288807	0.009831046
0.027534586	0.00075815	0.003032614	0.006065227	0.003790767
0.043255924	0.00187107	0.005613225	0.01122645	0.013097525
0.018851498	0.00035538	0.002132274	0.004619927	0.004975305
0.999929335	0.99985868	0	0	0
0.032946334	0.00108546	0.004341844	0.009769148	0.005427304
Jumlah	2.01104864	0.049188151	0.104656804	0.098050173

Tabel 3. 24 Perhitungan Pusat Cluster 2 (Iterasi 9)

μ_{i2}	μ_{i2}^{w}	$\mu_{i2}^{w} * X_i1$	$\mu_{i2}^{w} * X_i2$	$\mu_{i2}^{w} * X_i3$
0.026404754	0.00069721	0.004880477	0.009760954	0.009760954
5.64333E-05	3.1847E-09	0	0	0
0.115683759	0.01338273	0.12044459	0.281037376	0.173975518
0.871569506	0.7596334	2.27890021	4.557800421	6.836700631
0.869662127	0.75631222	3.781561077	7.563122154	6.050497723
0.952660909	0.90756281	3.630251232	7.260502463	4.53781404
0.930928682	0.86662821	2.599884634	5.199769268	6.066397479
0.089231015	0.00796217	0.047773044	0.103508263	0.111470437
5.64333E-05	3.1847E-09	0	0	0
0.936455589	0.87694907	3.507796281	7.892541633	4.384745352
Jumlah	4.18912783	15.97149154	32.86804253	28.17136213

Tabel 3. 25 Perhitungan Pusat Cluster 3 (Iterasi 9)

μ_{i3}	μ_{i3}^{w}	$\mu_{i3}^{w} * X_i1$	$\mu_{i3}^{w} * X_i2$	$\mu_{i3}^{w} * X_i3$
0.967523537	0.93610179	6.552712564	13.10542513	13.10542513
1.42312E-05	2.0253E-10	0	0	0
0.844151323	0.71259146	6.413323102	14.96442057	9.263688926
0.062231939	0.00387281	0.011618443	0.023236885	0.034855328
0.095282478	0.00907875	0.045393753	0.090787506	0.072630005
0.019804505	0.00039222	0.001568874	0.003137747	0.001961092
0.025815394	0.00066643	0.001999304	0.003998607	0.004665042
0.891917487	0.7955168	4.773100825	10.34171845	11.13723526
1.42312E-05	2.0253E-10	0	0	0
0.030598077	0.00093624	0.003744969	0.008426181	0.004681212
Jumlah	2.45915652	17.80346183	38.54115108	33.62514199

Pada tabel perhitungan pusat cluster 1 sampai 3 ditunjukkan bagaimana cara mendapatkan pusat cluster dengan menggunakan algoritma ke-d. Dimana angka acak dipangkatkan dua, kemudian hasil dari angka acak yang telah diberi pembobot dua dikalikan dengan Data Kecelakaan Lalu Lintas.

3. Perhitungan untuk menentukan pusat cluster baru dengan menggunakan rumus $(\mu_{ik}2^{w} * X_{ij}) / (\mu_{ik}2^{w})$

Tabel 3. 26 Pusat Cluster Baru (Iterasi 9)

V			
1	0.024458956	0.052040911	0.048755744
2	3.812605436	7.846034747	6.724875264
3	7.239661942	15.67250837	13.67344526

4. Perhitungan fungsi objektif seperti pada tabel berikut ini.

Tabel 3. 27 Perhitungan Fungsi Objektif (Iterasi 9)

Kuadrat Derajat Keanggotaan data ke-i			$[_{j=1}\sum(X_{ij} - V_{ij})^2) * \mu_{i1}{}^2]$
(K1)^2	(K2)^2	(K3)^2	L1
3.68657E-05	0.000697211	0.936101795	0.016141292
0.999858676	3.18472E-09	2.02527E-10	0.005682816
0.001613221	0.013382732	0.712591456	1.108463339
0.004382249	0.759633403	0.003872814	0.544962584
0.001228881	0.756312215	0.009078751	0.229727189
0.000758153	0.907562808	0.000392218	0.078461147
0.001871075	0.866628211	0.000666435	0.173171465
0.000355379	0.007962174	0.795516804	0.141438678
0.999858676	3.18472E-09	2.02527E-10	0.005682816
0.001085461	0.87694907	0.000936242	0.130673989

Tabel 3. 28 Perhitungan Fungsi Objektif (Iterasi 9)(Sambungan)

$[_{j=1}\sum((X_{ij} - V_{ij})^2) * \mu_{i2}{}^2]$	$[_{j=1}\sum((X_{ij} - V_{ij})^2) * \mu_{i3}{}^2]$	L1 + L2 + L3
L2	L3	
0.070389178	2.772134514	2.858664985
3.86371E-07	9.82264E-08	0.005683301
3.202662516	22.75624159	27.06736745
7.022328029	0.516529789	8.083820402
5.805010314	0.629896587	6.66463409
2.753560713	0.056711427	2.888733286
3.59119075	0.104008458	3.868370673
0.671016141	6.989172577	7.801627397
3.86371E-07	9.82264E-08	0.005683301
3.807666562	0.121942222	4.060282774
Fungsi Objektif		63.30486766

Perbaikan matriks partisi U

Tabel 3. 29 Perbaikan Matriks Partisi U (Iterasi 9)

$[_{j=1}\sum((X_{ij} - V_{1j})^2)^{-1}]$	$[_{j=1}\sum((X_{ij} - V_{2j})^2)^{-1}]$	$[_{j=1}\sum((X_{ij} - V_{1j})^2)^{-1}]$	LT
L1	L2	L3	L1 + L2 + L3
0.002283934	0.009905088	0.337682674	0.349871696
175.9442164	0.008242653	0.002061842	175.9545208
0.001455367	0.004178627	0.03131411	0.036948104
0.008041375	0.108174013	0.007497756	0.123713144
0.005349305	0.130286111	0.014413081	0.150048496

0.009662788	0.329596077	0.006916038	0.346174903
0.010804753	0.241320574	0.006407504	0.25853283
0.002512601	0.011865846	0.113821314	0.12819976
175.9442164	0.008242653	0.002061842	175.9545208
0.008306633	0.230311414	0.007677754	0.246295801

Tabel 3. 30 Perbaikan Matriks Partisi U (Iterasi 9) (Sambungan)

L1 / LT	L2 / LT	L3 / LT
0.00652792	0.02831063	0.96516145
0.999941437	4.68454E-05	1.1718E-05
0.039389483	0.113094492	0.847516026
0.06500017	0.874393854	0.060605976
0.035650505	0.868293344	0.096056151
0.027913023	0.952108526	0.019978451
0.041792577	0.933423323	0.0247841
0.019599108	0.092557472	0.887843419
0.999941437	4.68454E-05	1.1718E-05
0.033726248	0.935100855	0.031172897

Dari perhitungan matriks partisi U seperti pada tabel diatas, maka di dapatkan derajat keanggotaan baru untuk melanjutkan ke iterasi selanjutnya, yaitu :

Tabel 3. 31 Pembaharuan Matriks Partisi

μ_{ik}			
I	K_1	K_2	K_3
1	0.00652792	0.02831063	0.96516145
2	0.999941437	4.68454E-05	1.1718E-05
3	0.039389483	0.113094492	0.847516026
4	0.06500017	0.874393854	0.060605976
5	0.035650505	0.868293344	0.096056151
6	0.027913023	0.952108526	0.019978451
7	0.041792577	0.933423323	0.0247841
8	0.019599108	0.092557472	0.887843419
9	0.999941437	4.68454E-05	1.1718E-05
10	0.033726248	0.935100855	0.031172897

5. Kesimpulan setiap iterasi (Cek kondisi berhenti)

$|P_3 - P_2| = |\ 174.4666296 - 193.4153429\ | = 18.94871332 \gg \varepsilon\ (0.1)$ dan Iterasi = 3 <= MaxIter (9) maka proses berhenti.

6. Menentukan Cluster / Hasil Akhir

Bagaimana menentukan masing-masing data termasuk cluster yang mana saja ? Jawabannya yaitu mencari nilai maximum / nilai tertinggi dari ketiga cluster.

Tabel 3. 32 Hasil Akhir

Derajat Keanggotaan (m) data pada Cluster ke -			Data Cenderung Masuk Ke Cluster dengan Derajat Keanggotaan	Cluster
1	2	3		
0.00652792	0.02831063	0.96516145	0.96516145	3
0.999941437	4.68454E-05	1.1718E-05	0.999941437	1
0.039389483	0.113094492	0.847516026	0.847516026	3
0.06500017	0.874393854	0.060605976	0.874393854	2
0.035650505	0.868293344	0.096056151	0.868293344	2
0.027913023	0.952108526	0.019978451	0.952108526	2
0.041792577	0.933423323	0.0247841	0.933423323	2
0.019599108	0.092557472	0.887843419	0.887843419	3
0.999941437	4.68454E-05	1.1718E-05	0.999941437	1
0.033726248	0.935100855	0.031172897	0.935100855	2

3.5. Ringkasan Fuzzy C-Means Clustering

Pada pembahasan Bab 2 ini, Anda telah mempelajari mengenai definisi *Fuzzy C-Means Clustering*, Algoritma *Fuzzy C-Means Clustering*, dan cara penyelesaian masalah menggunakan *Fuzzy C-Means Clustering*, dengan studi kasus Data Kecelakaan Lalu Lintas. Tahapan pemecahan masalah untuk studi kasus Data Kecelakaan Lalu Lintas menggunakan *Fuzzy C-Means Clustering* adalah dimulai dari iterasi-1 hingga berhenti pada iterasi ke-9 , hingga dapat diambil kesimpulan bahwa terdapat 2 nama jalan yang tergolong Tidak Rawan kecelakaan, 5 nama jalan yang tergolong Rawan kecelakaan, dan 5 nama jalan yang tergolong Sangat Rawan kecelakaan.

Cluster 1	2
Cluster 2	5
Cluster 3	3
Jumlah	10

K1	K2	K3
2	4	1
9	5	3
	6	8
	7	
	10	

No	C1	C2	C3
1			Ring Road Utara
2	Ring Road Selatan		
3			Magelan Km 5-10
4		Kaliurang Km 10-15	
5		Jogja-Solo Km 10-15	
6		Adi Sucipto Km 5-10	
7		Godean Km 5-10	
8			Wates Km 5-10
9	Palagan		
10		Affandi	

3.6. Latihan Fuzzy C-Means Clustering

Terdapat sampel data rata-rata harga gabah menurut kualitas (rupiah /kg) selama satu tahun. Dalam data tersebut, berapakah harga yang tergolong Gabah Kering Giling, Gabah Kualitas Rendah, dan Gabah Kering Panen dengan menggunakan metode Fuzzy C—Means Clustering ?

Tabel 3. 33 Data Set Gabah

Bulan, Tahun	Kualitas Gabah		
	GKG	GKP	GKR
Januari, 2018	Rp 3,068	Rp 2,596	Rp 2,306
Februari, 2018	Rp 2,700	Rp 2,643	Rp 2,374
Maret, 2018	Rp 3,039	Rp 2,727	Rp 2,396
April, 2018	Rp 3,077	Rp 3,821	Rp 3,643
Mei, 2018	Rp 3,013	Rp 2,693	Rp 2,516
Juni, 2018	Rp 3,035	Rp 3,680	Rp 2,545
Juli, 2018	Rp 3,076	Rp 2,797	Rp 3,876
Agustus, 2018	Rp 3,183	Rp 2,842	Rp 3,805
September, 2018	Rp 3,083	Rp 3,890	Rp 2,710
Oktober, 2018	Rp 3,145	Rp 2,962	Rp 2,842
November, 2018	Rp 3,543	Rp 3,167	Rp 2,734
Desember, 2018	Rp 3,621	Rp 3,188	Rp 3,016

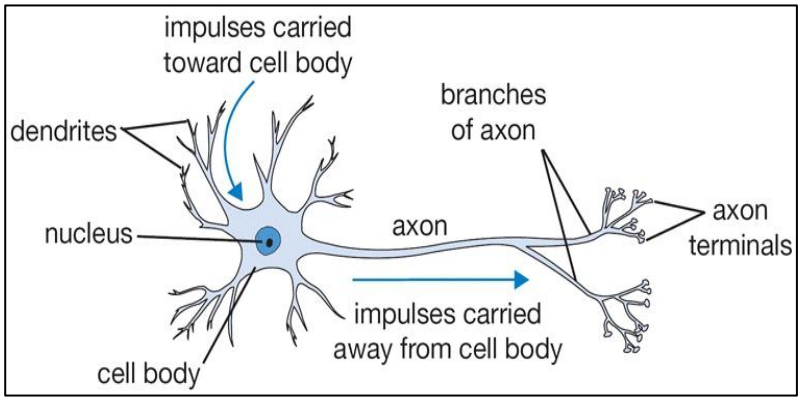
Keterangan :
GKG : Gabah Kering Giling
GKP : Gabah Kering Panen
GKR : Gabah Kualitas Rendah

BAB 4. PERCEPTRON

4.1 Pendahuluan

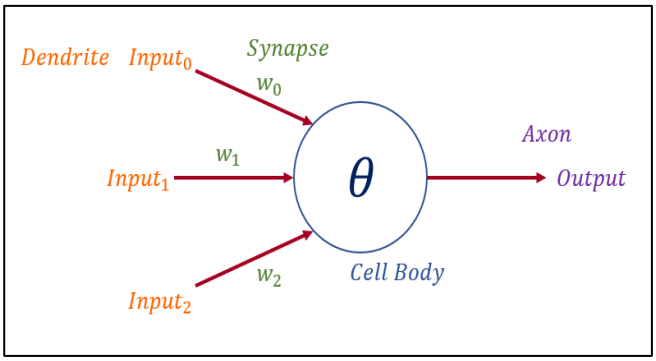
Neural Network Perceptron atau biasa disebut dengan *perceptron* adalah sebuah sistem yang meniru cara kerja *Biological Neural Networks* (jaringan syaraf makhluk hidup). *Biological Neural Network* merupakan kumpulan neuron yang mampu menerima rangsangan (*impulse*) dan melanjutkannya ke neuron lain jika rangsangan tersebut telah melebihi nilai/batas tertentu (*treshold*). Neuron-neuron pada *Biological Neural Networks* menerima rangsangan melalui kumpulan serabut halus yang disebut dengan *dendrite*.

Dendrite selanjutnya akan mengirimkan rangsangan yang diterima ke dalam *cell body* melalui saluran bernama *synapse*. Ketika rangsangan telah sampai di dalam *cell body*, rangsangan tersebut kemudian akan dinilai apakah layak diteruskan ke neuron selanjutnya atau tidak. Rangsangan yang layak akan diteruskan melalui *axon* untuk kemudian diterima oleh neuron selanjutnya melalui *dendrite*. Proses ini akan terus berlangsung hingga neuron terakhir atau rangsangan berhenti dilanjutkan karna tidak memenuhi nilai *threshold*.



Gambar 4. 1 Biological Neural Netwroks (cs231n, 2015)

Perceptron meniru cara kerja *Biological Neural Network* yang dapat menerima rangsangan dan meneruskannya. *Perceptron* mampu menerima lebih dari satu rangsangan (input) dan menghasilkan sebuah output berdasarkan input-input yang diterima. Nilai output *perceptron* didapat dari hasil perhitungan fungsi aktivasi terhadap input-inputnya. Umumnya output yang dihasilkan *perceptron* hanya bernilai nol atau satu (0 atau 1). Jika hasil perhitungan fungsi aktivasi melebihi *threshold* (melebihi nol), maka output *perceptron* akan bernilai satu. Sebaliknya, jika hasil perhitungan fungsi aktivasi bernilai kurang dari sama dengan *threshold* (kurang dari sama dengan nol), maka ouput *perceptron* akan bernilai nol.



Gambar 4. 2 Perceptron

Pada tabel 4.1 dapat dilihat persamaan antara *Biological Neural Networks* dan *Perceptron*.

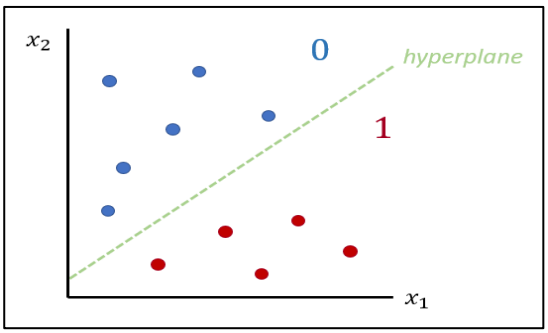
Tabel 4. 1 Persamaan Biological Neural Networks dan Perceptron

Biological Neural Networks	*Perceptron*
Dendrite	Input
Synapse	Bobot (*weight*)
Cell Body	Fungsi Aktivasi (*Treshold*)
Axon	Ouput

4.2 Neural Network Perceptron

4.2.1 Pendahuluan

Neural Network Perceptron atau biasa disebut *perceptron* adalah sistem yang berguna untuk klasifikasi data secara linier. *Perceptron* dapat mengklasifikasikan data ke dalam dua kelas berbeda (kelas nol atau satu) berdasarkan input-input yang diterima. Data tersebut diklasifikasikan menggunakan sebuah garis yang dinamakan *hyperplane*. Sebelum sebuah *perceptron* dapat mengklasifikasikan data, *perceptron* harus melalui proses *training*. Pada proses ini akan dicari *hyperplane* (nilai bobot/*weight*) yang paling optimal untuk diterapkan dalam sistem. Setelah *hyperplane* optimal ditemukan, proses selanjutnya adalah klasifikasi data menggunakan fungsi aktivasi.



Gambar 4. 3 Klasifikasi Data Menggunakan Perceptron

4.2.2 Persamaan Matematis dan Arsitektur Neural Network Perceptron

Model (persamaan matematis) *perceptron* secara umum dapat dituliskan sebagai berikut:

$$y = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n \quad \text{.....(4.1)}$$

dimana:

- y = output
- x_n = input ke-n
- w_n = bobot ke-n

$w_0 x_0$ pada persamaan (4.1) merupakan nilai bias sehingga persamaan (4.1) dapat diubah menjadi:

$$y = w_0 x_0 + \cdots + w_1 x_1 + b \quad \text{.....(4.2)}$$

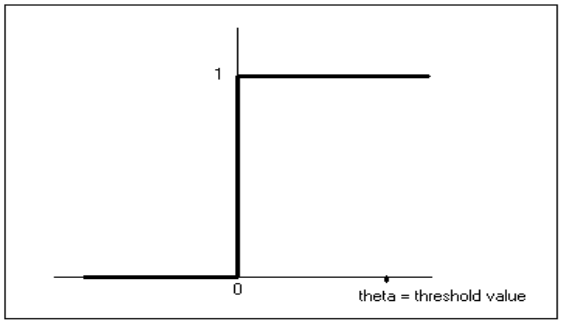
dimana b adalah nilai bias. Nilai bias diperoleh dari nilai w_0 dikali dengan satu. Selanjutnya persamaan (4.2) dapat disederhanakan menjadi:

$$\sum w_n x_n + b \quad \text{.....(4.3)}$$

Dalam mengklasifikasikan data, perceptron menggunakan fungsi aktivasi ***treshold*** terhadap hasil perhitungan persamaan (4.3) sebagai berikut:

$$a(x) \begin{cases} 1 & if\ w * x + b > 0 \\ 0 & otherwise \end{cases} \quad \text{.....(4.4)}$$

dimana $a(x)$ adalah hasil klasifikasi (kelas) data. Fungsi aktivasi *treshold* dapat diterjemahkan menjadi: jika hasil perhitungan persamaan (4.3) bernilai lebih dari nol, maka data yang di-input berada pada kelas satu. Namun, jika hasil perhitungan persamaan (4.3) bernilai kurang dari sama dengan nol, maka data yang di-input berada pada kelas nol.



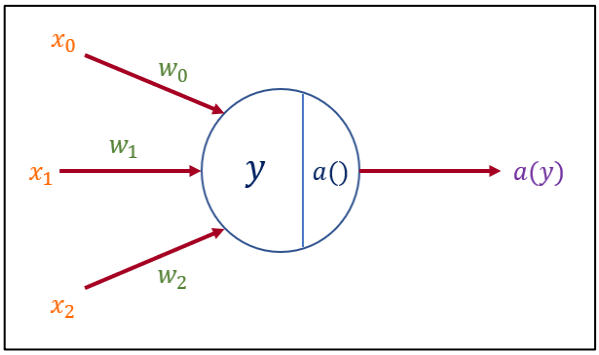
Gambar 4. 4 Fungsi Aktivasi Treshold (Robot, 2006)

Selain persamaan dan fungsi aktivasi di atas, *perceptron* juga memiliki persamaan yang digunakan pada proses *training* (mencari nilai w), yakni sebagai berikut:

$$w_n\ new = w_n\ old + \alpha\ (t - y)\ x_n \quad \text{.....(4.5)}$$

dimana:

- $w_n\ new$ = bobot baru ke-n
- $w_n\ old$ = bobot lama ke-n
- α = *learning rate* (dapat berbeda pada setiap iterasi ataupun *epoch*)
- t = ouput yang diharapkan/seharusnya
- y = output
- x_n = input ke-n



Gambar 4. 5 Arsitektur Neural Network Perceptron

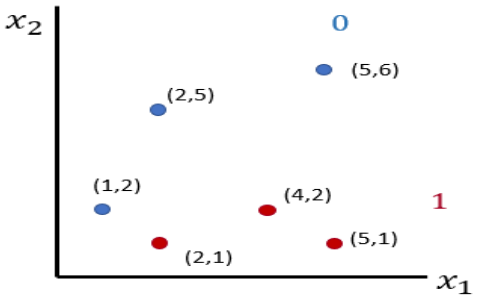
4.2.3 Prosedur / Algoritma Neural Network Perceptron

Prosedur atau algoritma pembentukan model matematis *perceptron* adalah sebagai berikut:

1. Siapkan data *training* dan data *testing*. Data *training* akan digunakan pada proses *training*, sementara data *testing* akan digunakan pada tahap uji coba model atau persamaan matematis yang akan dibentuk.
2. Tentukan nilai w_0 ... w_n awal, batas *epoch*, dan nilai *learning rate*.
3. Input nilai w_0 ... w_n yang telah ditentukan sebelumnya dan nilai x_1 ... x_n data pertama ke dalam persamaan (4.3) untuk mendapatkan nilai output data ke-1. x_0 selalu bernilai satu. Gunakan data *training* pada langkah ini dan seterusnya.
4. Tentukan kelas data pertama dengan menginput nilai output data ke-1 ke dalam fungsi aktivasi ***treshold*** /persamaan (4.4).
5. Cek apakah kelas data telah sesuai dengan kelas yang diharapkan/seharusnya.
6. Jika ya, gunakan nilai w_0 ... w_n untuk menghitung nilai output data ke-2.
7. Jika tidak, *update* nilai $w_0\ new$ ... $w_n\ new$ menggunakan persamaan (4.5), kemudian gunakan nilai $w_0\ new$ ... $w_0\ new$ tersebut untuk menghitung nilai ouput data ke-2.
8. Tentukan kelas data kedua dengan menginput nilai output data ke-2 ke dalam fungsi aktifasi ***treshold***/persamaan (4.4)**.**
9. Ulangi langkah lima, enam, tujuh, dan delapan pada seluruh data *training*. Lakukan hingga kelas setiap data sudah sesuai dengan kelas yang diharapkan/seharusnya atau hingga batas *epoch*. *Epoch* adalah siklus *training* seluruh data *training* (iterasi data pertama hingga terakhir).
10. Jika kelas tiap data *training* sudah sesuai dengan kelas yang diharapkan/seharusnya. Input nilai w_0 ... w_n yang didapat ke dalam persamaan (4.1) untuk mendapatkan model matematis *perceptron*.
11. Model matematis yang dibentuk pada langkah sepuluh siap digunakan untuk klasifikasi data (uji coba menggunakan data *testing*).

4.2.4 Sample Case

Tentukan model (persamaan matematis) perceptron pada *scatters* data berikut ini:



Berdasarkan *scatters* tersebut, didapat dataset sebagai berikut:

Tabel 4. 2 Dataset Scatters Sample Case

x1	x2	t
1	2	0
2	1	1
2	5	0
4	2	1
5	1	1
5	6	0

Tentukan nilai $w_0 \dots w_n$ dan *learing rate* yang akan digunakan pada proses *training*. Ikuti seluruh langkah pada prosedur/algoritma *Neural Network Perceptron* yang telah dijelaskan pada subsubbab sebelumnya. Contoh proses *training perceptron*:

Tabel 4. 3 Proses Training

Bias	x1	x2	w0	w1	w2	Y	a(y)	T	α	w0 new	w1 new	w2 new
1	1	2	0	0	0	0	0	0	0.2			
1	2	1	0	0	0	0	0	1		0.2	0.4	0.2
1	2	5	0.2	0.4	0.2	2	1	0		0	0	-0.8
1	4	2	0	0	-0.8	-1.6	0	1		0.2	0.8	-0.4
1	5	1	0.2	0.8	-0.4	3.8	1	1				
1	5	6	0.2	0.8	-0.4	1.8	1	0		0	-0.2	-1.6
1	1	2	0	-0.2	-1.6	-3.4	0	0	0.2			
1	2	1	0	-0.2	-1.6	-2	0	1		0.2	0.2	-1.4
1	2	5	0.2	0.2	-1.4	-6.4	0	0				
1	4	2	0.2	0.2	-1.4	-1.8	0	1		0.4	1	-1
1	5	1	0.4	1	-1	4.4	1	1				
1	5	6	0.4	1	-1	-0.6	0	0				
1	1	2	0.4	1	-1	-0.6	0	0	0.2			
1	2	1	0.4	1	-1	1.4	1	1				
1	2	5	0.4	1	-1	-2.6	0	0				
1	4	2	0.4	1	-1	2.4	1	1				
1	5	1	0.4	1	-1	4.4	1	1				
1	5	6	0.4	1	-1	-0.6	0	0				

Berdasarkan tabel 4.2, dibutuhkan tiga *epoch* untuk mendapatkan nilai $w_0 \dots w_n$ yang optimal. Selanjutnya, susun nilai $w_0 \dots w_n$ tersebut ke dalam persamaan (4.1) untuk membentuk model perceptron, seperti berikut:

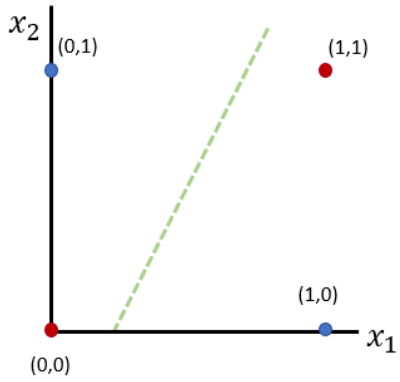
$$y = 0.4 * 1 + 1\,(x_1) - 1\,(x_2)$$

4.3 Multilayer Perceptron

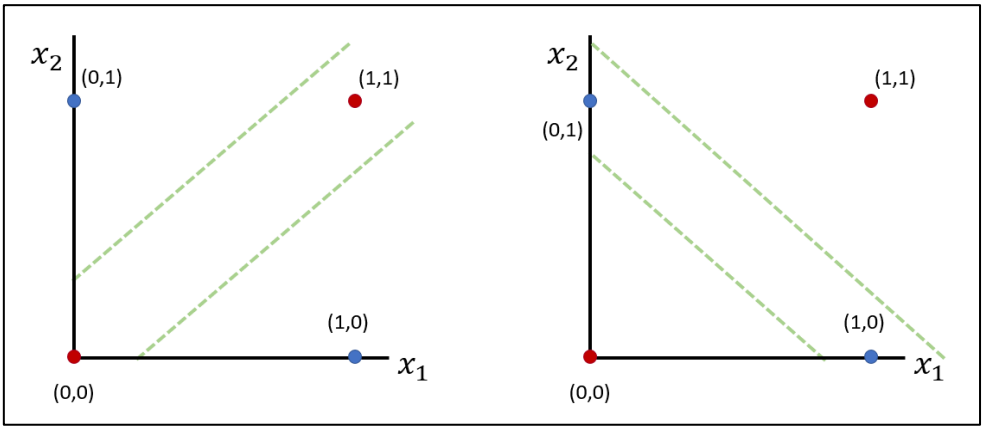
4.3.1 Pendahuluan

Multilayer Perceptron (MLP) merupakan pengembangan lebih lanjut dari *perceptron*. Alasan MLP dikembangkan adalah untuk mengatasi kelemahan *perceptron* yang tidak mampu mengklasifikasikan data non-linier. Contoh data non-linear yang tidak bias diklasifikasin oleh *perceptron* adalah kasus XOR (*Exclusive OR*). Pada kasus XOR seperti yang terlihat pada gambar 4.6, data-data yang tersebar tidak dapat diklasifikasikan jika hanya menggunakan satu *hyperplane* (menggunakan *perceptron*). Dibutuhkan dua *hyperplane* untuk mengklasifikasikan setiap data yang ada.

x1	x2	S(y)
0	0	0
0	1	1
1	0	1
1	1	0

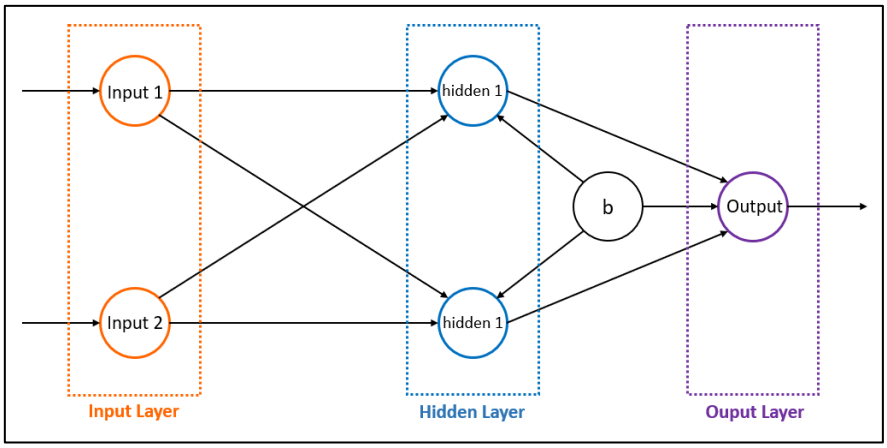


Gambar 4. 6 Penyelesaian Kasus XOR Menggunakan Perceptron



Gambar 4. 7 Penambahan Hyperplane Untuk Menyelesaikan Kasus XOR

Penambahan *hyperplane* dibutuhkan untuk menyelesaikan kasus XOR. Penambahan *hyperplane* berarti juga melakukan pengembangan *perceptron* menjadi sebuah sistem baru. Setelah melewati tahap pengembangan, sistem baru tersebut menghasilkan beberapa lapisan, yakni: input layer, *hidden layer*, dan ouput layer. Dikarenakan terdiri dari banyak lapisan, sistem tersebut dinamakan Multilayer Perceptron (MLP).



Gambar 4. 8 Pengembangan Perceptron Menjadi Multilayer Perceptron (MLP)

4.3.2 Persamaan Matematis dan Arsitektur Multi Layer Perceptron

Dari pengembangan persamaan (4.1), model (persamaan matematis) *perceptron* secara umum dapat dituliskan sebagai berikut:

$$y = w_0x_0 + w_1S(w_0x_0 + w_1x_1 + \cdots + w_nx_n) + w_2S(w_0x_0 + w_1x_1 + \cdots + w_nx_n) + \cdots + w_nS(w_0x_0 + w_1x_1 + \cdots + w_nx_n) \quad \text{.....(4.6)}$$

$S(\dots)$ merupakan fungsi aktivasi pada MLP. Input yang masuk ke dalam sebuah *perceptron* (neuron) dalam sistem MLP merupakan output *perceptron* (neuron) sebelumnya. Setiap w_0x_0 pada persamaan (4.6) merupakan nilai bias sehingga persamaan (4.6) dapat diubah menjadi:

$$y = w_1S(w_1x_1 + \cdots + w_nx_n + b) + w_2S(w_1x_1 + \cdots + w_nx_n + b) + \cdots + w_nS(w_0x_0 + w_1x_1 + \cdots + w_nx_n) + b \quad \text{.....(4.7)}$$

Nilai bias didapatkan dari perkalian antara nilai w_0 dengan satu. Selanjutnya persamaan (4.7) dapat disederhanakan menjadi:

$$y = \sum w_n S(h_n) + b \quad \text{.....(4.8)}$$

h_n merupakan nilai output *layer* sebelumnya ke-n setelah melewati fungsi aktivasi $S(\dots)$.

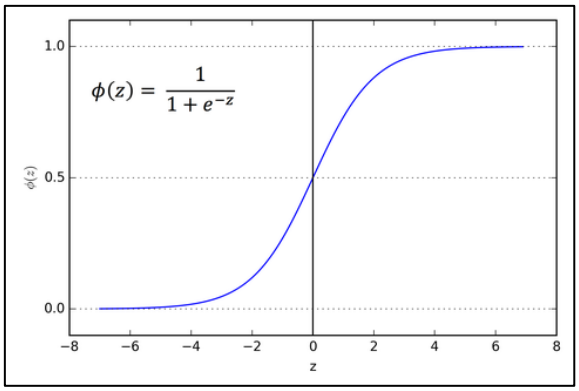
Berbeda dengan *perceptron*. MLP menggunakan fungsi aktivasi ***sigmoid*** dalam mengklasifikasikan data/output tiap neuron yang ada di dalam sistem. Fungsi tersebut dapat dituliskan sebagai berikut:

$$S(x) = \frac{1}{1+e^{-x}} \quad \text{.....(4.9)}$$

dimana:

- x adalah output sebuah neuron
- e adalah logaritma natural
- $S(x)$ adalah hasil klasifikasi output neuron x

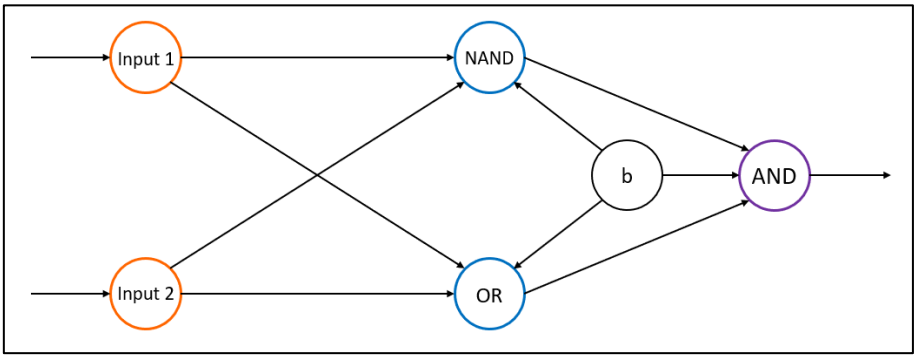
Jika hasil fungsi aktivasi *sigmoid* bernilai lebih dari 0.5, maka data yang di-input berada pada kelas satu. Sedangkan, jika hasil fungsi aktivasi *sigmoid* bernilai kurang dari sama dengan 0.5, maka data yang di-inputkan berada pada kelas nol.



Gambar 4. 9 Fungsi Aktivasi Sigmoid (Kashyap, 2018)

Selain persamaan dan fungsi training, MLP juga memiliki algoritma *training*. Algoritma yang umum digunakan dalam proses *training* MLP adalah *Backpropagation*. Algoritma ini dijelaskan pada bab 5.

Selain menggunakan *Backpropagation* (dalam kasus XOR) MLP juda dapat di-*training* dengan memecah setiap neuron dalam sistem menjadi kumpulan *perceptron* (pada dasarnya MLP kasus XOR merupakan kombinasi antara neuron NAND (Not AND), OR, dan OR). Kumpulan *perceptron* tersebut kemudian dapat di *training* satu per satu menggunakan persamaan (4.5). Proses *training* akan menghasilkan nilai $w_0 \dots w_n$ optimal pada setiap *perceptron*. Nilai $w_0 \dots w_n$ optimal kemudian dapat digunakan untuk menyusun model matematis tiap-tiap *perceptron*. Jadi MLP kasus XOR akan memiliki tiga model matematis yang terdiri dari model matematis *perceptron*: NAND, OR, dan AND.



Gambar 4. 10 MLP Merupakan Kombinasi Perceptron NAND, OR, dan AND

4.3.3 Prosedur / Algoritma Multi Layer Perceptron pada Kasus XOR

Prosedur atau algoritma pembentukan model matematis tiap *perceptron* adalah sebagai berikut:

1. Pecah MLP kasus XOR menjadi tiga buah *perceptron* yakni: NAND, OR, dan AND.
2. Lakukan proses *training* terhadap ketiga *perceptron* tadi untuk membentuk model matematisnya dengan mengikuti prosedur training.
3. Tiga model matematis (NAND, OR, dan AND) yang dibentuk pada proses *training* siap digunakan untuk klasifikasi data (uji coba menggunakan data *testing*). Tiga model matis tersebut juga dapat digabung menjadi satu menggunakan persamaan (4.7).

Prosedur *training* atau algoritma training:

1. Siapkan data *training* untuk setiap *perceptron*. Data *training* dapat diambil dari tabel kebenaran masing-masing *perceptron.*
2. Siapkan data *testing*.
3. Tentukan nilai w_0 ... w_n awal, batas *epoch*, dan nilai *learning rate*.
4. Input nilai w_0 ... w_n yang telah ditentukan sebelumnya dan nilai x_1 ... x_n data pertama ke dalam persamaan (4.3) untuk mendapatkan nilai output data ke-1. x_0 selalu bernilai satu. Gunakan data *training* pada langkah ini dan seterusnya.
5. Tentukan kelas data pertama dengan menginput nilai output data ke-1 ke dalam fungsi aktivasi ***sigmoid*** /persamaan (4.9).
6. Cek apakah kelas data telah sesuai dengan kelas yang diharapkan/seharusnya.
7. Jika ya, gunakan nilai w_0 ... w_n untuk menghitung nilai output data ke-2.
8. Jika tidak, *update* nilai $w_0\ new$ ... $w_n\ new$ menggunakan persamaan (4.5), kemudian gunakan nilai $w_0\ new$ ... $w_0\ new$ tersebut untuk menghitung nilai ouput data ke-2.
9. Tentukan kelas data kedua dengan menginput nilai output data ke-2 ke dalam fungsi aktifasi ***sigmoid***/persamaan (4.9)**.**
10. Ulangi langkah lima, enam, tujuh, dan delapan pada seluruh data *training*. Lakukan hingga kelas setiap data sudah sesuai dengan kelas yang diharapkan/seharusnya atau hingga batas *epoch*. *Epoch* adalah siklus *training* seluruh data *training* (iterasi data pertama hingga terakhir).
11. Jika kelas tiap data *training* sudah sesuai dengan kelas yang diharapkan/seharusnya. Input nilai w_0 ... w_n yang didapat ke dalam persamaan (4.1) untuk mendapatkan model matematis *perceptron*.

4.3.4 Sample Case – XOR Problem

Tentukan model matematis tabel kebenaran XOR (tabel 4.3)

Tabel 4. 4 Tabel Kebenaran XOR

x1	x2	T
0	0	0
0	1	1
1	0	1
1	1	0

Berdasarkan hasil pemecahan XOR didapatkan tiga buah dataset (data *training*) berupa tabel kebenaran NAND, OR, dan AND.

Tabel 4. 5 Dataset Satu (Tabel Kebenaran NAND)

x1	x2	t (NAND)
0	0	1
0	1	1
1	0	1
1	1	0

Tabel 4. 6 Dataset Dua (Tabel Kebenaran OR)

x1	x2	t (OR)
0	0	0
0	1	1
1	0	1
1	1	1

Tabel 4. 7 Dataset Tiga (Tabel Kebenaran AND)

t (NAND)	t (OR)	t (AND)
1	0	0
1	1	1
1	1	1
0	1	0

Lakukan *training* ketiga *perceptron* (NAND, OR, AND) menggunakan dataset pada tabel 4.4, 4.5, dan 4.6. Masing-masing *perceptron* akan memiliki tiga buah input (x1, x2, dan bias) dan satu buah output. Output *perceptron* yang terletak di *hidden layer* (NAND dan OR) akan menjadi input pada *perceptron* AND yang terletak di output layer.

Tabel 4. 8 Training Perceptron NAND

bias	x1	x2	w0	w1	w2	NAND	S(NAND)	h1	t	α	w0 new	w1 new	w2 new
1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.0000	1.0000	0.2	0.1000	0.0000	0.0000
1.0000	0.0000	1.0000	0.1000	0.0000	0.0000	0.1000	0.5250	1.0000	1.0000				
1.0000	1.0000	0.0000	0.1000	0.0000	0.0000	0.1000	0.5250	1.0000	1.0000				
1.0000	1.0000	1.0000	0.1000	0.0000	0.0000	0.1000	0.5250	1.0000	0.0000		-0.0050	-0.1050	-0.1050
1.0000	0.0000	0.0000	-0.0050	-0.1050	-0.1050	-0.0050	0.4988	0.0000	1.0000	0.2	0.1454	-0.1050	-0.1050
1.0000	0.0000	1.0000	0.1454	-0.1050	-0.1050	0.0404	0.5101	1.0000	1.0000				
1.0000	1.0000	0.0000	0.1454	-0.1050	-0.1050	0.0404	0.5101	1.0000	1.0000				
1.0000	1.0000	1.0000	0.1454	-0.1050	-0.1050	-0.0646	0.4839	0.0000	0.0000				
1.0000	0.0000	0.0000	0.1454	-0.1050	-0.1050	0.1454	0.5363	1.0000	1.0000				
1.0000	0.0000	1.0000	0.1454	-0.1050	-0.1050	0.0404	0.5101	1.0000	1.0000				
1.0000	1.0000	0.0000	0.1454	-0.1050	-0.1050	0.0404	0.5101	1.0000	1.0000				
1.0000	1.0000	1.0000	0.1454	-0.1050	-0.1050	-0.0646	0.4839	0.0000	0.0000				

Tabel 4. 9 Training Perceptron OR

bias	x1	x2	w0	w1	w2	OR	S(OR)	h2	t	α	w0 new	w1 new	w2 new
1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.0000	0.0000	0.2			
1.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.0000	1.0000		0.1000	0.0000	0.1000
1.0000	1.0000	0.0000	0.1000	0.0000	0.1000	0.1000	0.5250	1.0000	1.0000				
1.0000	1.0000	1.0000	0.1000	0.0000	0.1000	0.2000	0.5498	1.0000	1.0000				
1.0000	0.0000	0.0000	0.1000	0.0000	0.1000	0.1000	0.5250	1.0000	0.0000	0.2	-0.0050	0.0000	0.1000
1.0000	0.0000	1.0000	-0.0050	0.0000	0.1000	0.0950	0.5237	1.0000	1.0000				
1.0000	1.0000	0.0000	-0.0050	0.0000	0.1000	-0.0050	0.4988	0.0000	1.0000		0.0953	0.1002	0.1000
1.0000	1.0000	1.0000	0.0953	0.1002	0.1000	0.2955	0.5733	1.0000	1.0000				
1.0000	0.0000	0.0000	0.0953	0.1002	0.1000	0.0953	0.5238	1.0000	0.0000	0.2	-0.0095	0.1002	0.1000
1.0000	0.0000	1.0000	-0.0095	0.1002	0.1000	0.0905	0.5226	1.0000	1.0000				
1.0000	1.0000	0.0000	-0.0095	0.1002	0.1000	0.0907	0.5227	1.0000	1.0000				
1.0000	1.0000	1.0000	-0.0095	0.1002	0.1000	0.1907	0.5475	1.0000	1.0000				
1.0000	0.0000	0.0000	-0.0095	0.1002	0.1000	-0.0095	0.4976	0.0000	0.0000				
1.0000	0.0000	1.0000	-0.0095	0.1002	0.1000	0.0905	0.5226	1.0000	1.0000				
1.0000	1.0000	0.0000	-0.0095	0.1002	0.1000	0.0907	0.5227	1.0000	1.0000				
1.0000	1.0000	1.0000	-0.0095	0.1002	0.1000	0.1907	0.5475	1.0000	1.0000				

Tabel 4. 10 Training Perceptron AND

bias	h1	h2	w0	w1	w2	AND	S(AND)	Y	t	α	w0 new	w1 new	w2 new
1.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.0000	0.0000	0.1			
1.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.0000	1.0000		0.0500	0.0500	0.0500
1.0000	1.0000	1.0000	0.0500	0.0500	0.0500	0.1500	0.5374	1.0000	1.0000				
1.0000	0.0000	1.0000	0.0500	0.0500	0.0500	0.1000	0.5250	1.0000	0.0000		-0.0025	0.0500	-0.0025
1.0000	1.0000	0.0000	-0.0025	0.0500	-0.0025	0.0475	0.5119	1.0000	0.0000	0.2	-0.1049	-0.0524	-0.0025
1.0000	1.0000	1.0000	-0.1049	-0.0524	-0.0025	-0.1597	0.4601	0.0000	1.0000		0.0031	0.0556	0.1055
1.0000	1.0000	1.0000	0.0031	0.0556	0.1055	0.1642	0.5409	1.0000	1.0000				
1.0000	0.0000	1.0000	0.0031	0.0556	0.1055	0.1086	0.5271	1.0000	0.0000		-0.1023	0.0556	0.0000
1.0000	1.0000	0.0000	-0.1023	0.0556	0.0000	-0.0467	0.4883	0.0000	0.0000	0.1			
1.0000	1.0000	1.0000	-0.1023	0.0556	0.0000	-0.0467	0.4883	0.0000	1.0000		-0.0512	0.1068	0.0512
1.0000	1.0000	1.0000	-0.0512	0.1068	0.0512	0.1068	0.5267	1.0000	1.0000				
1.0000	0.0000	1.0000	-0.0512	0.1068	0.0512	0.0001	0.5000	0.0000	0.0000				
1.0000	1.0000	0.0000	-0.0512	0.1068	0.0512	0.0556	0.5139	1.0000	0.0000	0.1	-0.1025	0.0554	0.0512
1.0000	1.0000	1.0000	-0.1025	0.0554	0.0512	0.0040	0.5010	1.0000	1.0000				
1.0000	1.0000	1.0000	-0.1025	0.0554	0.0512	0.0040	0.5010	1.0000	1.0000				
1.0000	0.0000	1.0000	-0.1025	0.0554	0.0512	-0.0513	0.4872	0.0000	0.0000				
1.0000	1.0000	0.0000	-0.1025	0.0554	0.0512	-0.0513	0.4872	0.0000	0.0000				
1.0000	1.0000	1.0000	-0.1025	0.0554	0.0512	0.0040	0.5010	1.0000	1.0000				
1.0000	1.0000	1.0000	-0.1025	0.0554	0.0512	0.0040	0.5010	1.0000	1.0000				
1.0000	0.0000	1.0000	-0.1025	0.0554	0.0512	-0.0513	0.4872	0.0000	0.0000				

Berdasarkan hasil *training perceptron* NAND, OR, dan AND, didapatkan tiga model matematis:

1. Model matematis *perceptron* NAND
$$h1 = S(0.1454 * 1 - 0.1050(x1) - 0.1050(x2))$$
2. Model matematis *perceptron* OR
$$h2 = S(-0.0095 * 1 + 0.1002(x1) + 0.1(x2))$$
3. Model matematis *perceptron* AND
$$y = S(-0.1025 * 1 + 0.0554(h1) + 0.0512(h2))$$

Ketiga model matematis di atas merupakan solusi untuk MLP kasus XOR. Ketiga model ini selanjutnya dapat digunakan untuk melakukan klasifikasi data yang masih berkaitan dengan kasus XOR (uji coba data *testing*).

4.4 Ringkasan Perceptron dan Multilayer Perceptron

1. *Perceptron* merupakan sistem yang meniru cara kerja jaringan syaraf makhluk hidup.
2. *Perceptron* berguna untuk mengklasifikasikan data secara linier (menggunakan satu buah garis).
3. Garis klasifikasi *perceptron* disebut juga sebagai *hyperplane*.
4. *Perceptron* dan MLP mampu megklasifikasikan data ke dalam dua kelas yang berbeda (kelas satu dan nol) berdasarkan hasil perhitungan output data menggunakan fungsi aktivasi.
5. *Perceptron* menggunakan fungsi aktivasi ***Treshold*** dalam proses klasifikasi data.
6. Sebelum klasifikasi dilakukan, *perceptron* harus di-*training* terlebih dahulu.
7. *Training* merupakan proses untuk menentukan *hyperplane* yang sesuai dengan mencari nilai $w_0 \dots w_n$ optimal dan membentuk model matematis.
8. Data *training* digunakan dalam proses *training*.
9. Model matematis yang dihasilkan dari proses *training* dapat digunakan untuk mengklasifikasikan data *testing*.
10. MLP merupakan pengembangan lebih lanjut dari *perceptron*.
11. MLP berguna untuk mengklasifikasikan data yang tidak bisa diselesaikan oleh *perceptron*, contohnya adalah data pada kasus XOR.
12. MLP menggunakan dua buah *hyperplane* dalam mengklasifikasikan data.
13. MLP terdiri dari kombinasi *perceptron* NAND, OR, dan AND.
14. MLP memiliki *input layer*, *hidden layer*, dan *output layer* di dalam sistemnya.
15. MLP menggunakan fungsi aktivasi ***Sigmoid*** dalam proses klasifikasi data.
16. MLP dapat di-*training* menggunakan algoritma ***Backpropagation***. Cara lainnya adalah dengan men-*training* MLP per neuron/*perceptron* (memecah MLP terlebih dahulu).

4.5 Latihan Perceptron

Tentukan model matematis dari dataset berikut ini (tabel 4.9)

Tabel 4. 11 Tabel Kebenaran XNOR

x1	x2	T
0	0	1
0	1	0
1	0	0
1	1	1

BAB 5. BACKPROPAGATION

5.1 Pendahuluan

Jaringan Syaraf Tiruan (JST) adalah bentuk penerapan dari cara kerja otak manusia dengan tujuan mensimulasikan proses pembelajaran melalui perubahan sinapsis otak manusia (Andrijasa, 2010), dalam (Y. A. Lesnussa., 2015)

Backpropagation adalah algoritma hitung multilayer dengan tujuan memperkecil error toleransi, dilakukan dengan cara penyesuaian bobot antara output dengan target yang diinginkan. Backpropagation disebut algoritma hitung multilayer karena mempunyai tiga layer yang berhubungan dalam proses perhitungannya. Layer tersebut terbagi menjadi *input layer, hidden layer, dan output layer.* Input layer berfungsi sebagai data awalan yang ingin diproses pada algoritma backpropagation setelah itu dilanjutkan ke tahapan hidden layer. Hidden layer merupakan tempat perhitungan dengan hasil penyesuaian bobot sehingga dapat diteruskan ke output layer dengan target yang diinginkan. Output layer merupakan hasil atau target yang telah ditentukan pada metode backpropagation.

Tujuan dari Metode backpropagation adalah menentukan target yang ingin diciptakan, target tersebut didapat dari perhitungan dari data yang telah dibuat. Pada penggunaan metode backpropagation biasanya menggunakan fungsi sigmoid binner/sigmoid function dan sigmoid bipolar. Sigmoid binner memiliki range 0 – 1 (Erna Budhiarta Nababan, 2015). Definisi rumus sigmoid binner adalah sebagai berikut :

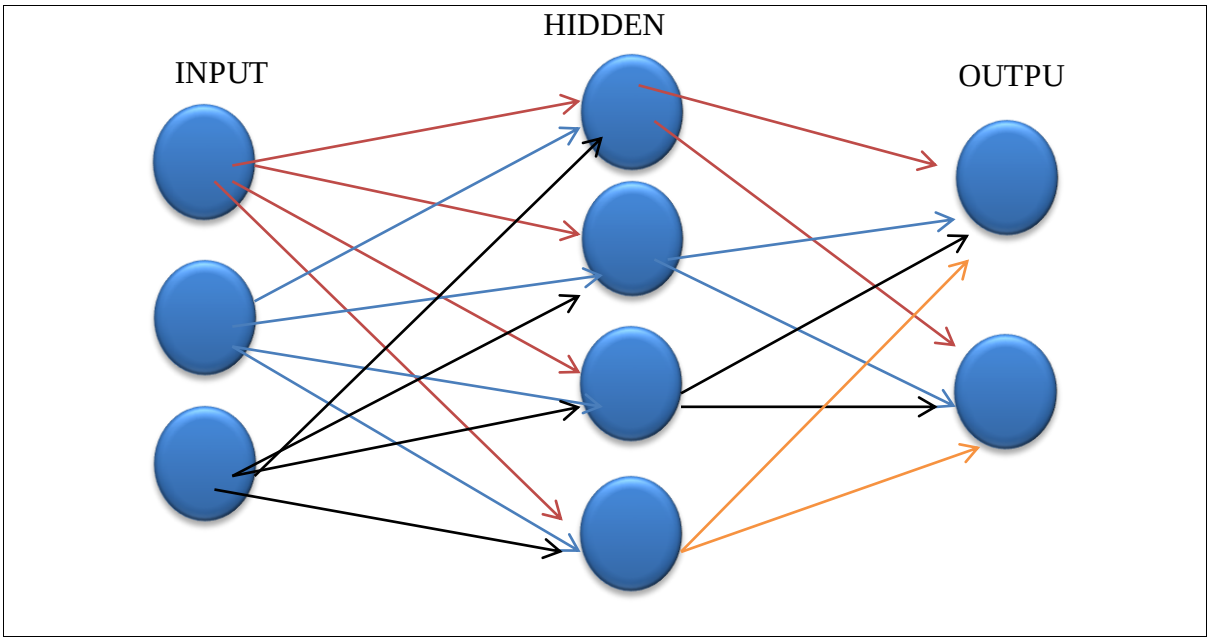
$$y = f(x) = \frac{1}{(1+e^{-x})} \quad (1)$$

Fungsi selanjutnya adalah penggunaan fungsi sigmoid Bipolar dengan range antara 1 sampai -1, yang didefinisikan sebagai berikut :

$$y = f(x) = \frac{1-e^{-x}}{(1+e^{-x})} \quad (2)$$

5.1.1 Arsitektur Backpropagation

Arsitektur dari backpropagation adalah input layer, hidden layer, dan output layer. Yang mana saling berhubungan satu sama lain, pada input layer tidak terjadi perhitungan atau komputasi, pada hidden layer dan output layer terjadi proses perhitungan atau komputasi.



Gambar 5. 1 Arsitektur Backpropagation

Pada backpropagation terdapat juga weight dan bias. Weight adalah bobot yang terhubung antar layer. Bias adalah banyak data yang dihasilkan setelah penentuan weight pada layer. Pada gambar diatas terdapat 12 weight dan 4 bias diantara input layer dan hidden layer, terdapat 8 weight dan 2 bias daiantara hidden layer dan output layer.

5.1.2 Perbedaan Backpropagation dengan Single Layer Network Arsitektur *Backpropagation*

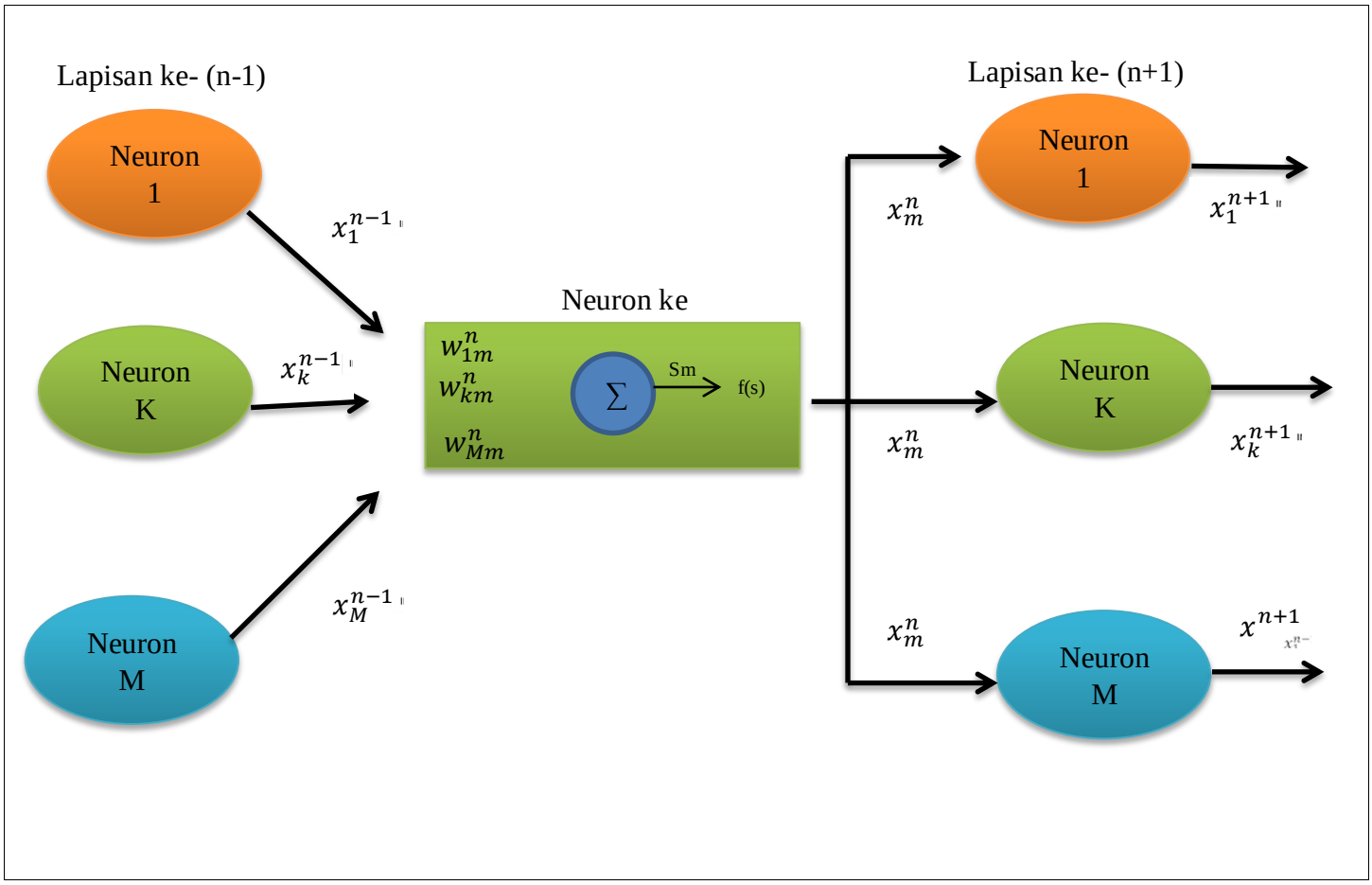
Backpropagation merupakan pengembangan dari metode single layer network. Single layer network merupakan jaringan yang hanya memiliki dua layer, yaitu input layer dan output layer. Pada input layer memiliki data yang ingin diproses, lalu data tersebut langsung diberikan ke output layer tanpa harus melewati hidden layer. Perbedaan mendasar adalah Backpropagation memiliki proses yang lebih baik dalam mengurangi error toleransi karena memiliki hidden layer yang berfungsi sebagai proses komputasi data yang ada, sedangkan single layer network tidak memiliki hidden layer (Ginanto, 2012)

5.1.3 Kelebihan dan Kekurangan Backpropagation

Kelebihan metode Backpropagation adalah baik dalam melakukan penanganan masalah dengan pola kompleks, karena pada input layer, hidden layer, dan output layer memiliki kaitan dan saling terhubung dan kelemahan dalam metode Backpropagation adalah cenderung sangat lama dalam konvergen dan permasalahn dalam local minimum sehingga sering terjadi kesalahan pada local minimum yang ada

5.2 Persamaan Matematis *Backpropagation*

Pada pembahasan Algoritma Backpropagation akan ditentukan persamaan matematis seperti yang akan dijelaskan pada pembahasan kali ini :



Gambar 5. 2 Persamaan Matematis Backpropagation

Berikut ini merupakan persamaan matematis *backpropagation:*

1. Digunakan metode *gradient descent*

$$w_k \longleftarrow w_k - \mu \frac{\partial E}{\partial w_k} \quad (1)$$

$\frac{\partial E}{\partial w_k}$ = turunan parsial terhadap bobot w_k

μ = learning rate

2. Untuk menurunkan aturan dari pembaharuan bobot digunakan rumus atau aturan *chain rule*

$$\frac{\partial E}{\partial w_{km}^n} = \frac{\partial E}{\partial s_m^n} \frac{\partial s_m^n}{\partial w_{km}^n} \quad (2)$$

3. Jumlah input yang telah diberi bobot untuk neuron ke-n sebelum masuk ke fungsi transfer non-linear

$$s_m^n = \sum_{j=1}^{M} w_{jm}^n x_j^{n-1} \quad (3)$$

atau menjadi $\frac{\partial s_m^n}{\partial w_{km}^n} = x_k^{n-1}$

4. Substitusikan persamaan 3 dengan persamaan 2 maka akan menghasilkan persamaan

$$\frac{\partial E}{\partial w_{km}^n} = x_k^{n-1} \frac{\partial E}{\partial w_m^n} \quad (4)$$

5. Setelah digunakan aturan rantai didapat persamaan :

$$\frac{\partial E}{\partial w_{km}^n} = x_k^{n-1} \frac{\partial E}{\partial x_m^n} \frac{\partial x_m^n}{s_m^n} \quad (5)$$

6. Langkah selanjutnya adalah menggunakan fungsi sigmoid menjadi :

$f(x) = \frac{1}{1+e^{-x}}$ menjadi $f'(x) = f(x)[1 - f(x)]$ (6)

7. karena $f(s_m^n) = x_m^n$, persamaan 5 menjadi :

$$\frac{\partial E}{\partial w_{k\,m}^n} = x_k^{n-1} f'(s_m^n) \frac{\partial E}{\partial x_m^n}$$ (derivative suatu bobot terhadap error jaringan) (7)

8. Menemukan ekspresi rekursif dengan menggunakan aturan rantai :

$$\frac{\partial E}{\partial x_m^n} = \sum_j w_{mj}^{n+1} f'(s_j^{n+1}) \frac{\partial E}{\partial x_j^{n+1}}$$ (8)

9. Mendefinisikan $\delta_m^n = f'(s_m^n) \frac{\partial E}{\partial x_m^n}$ (definisi 9) dengan persamaan 8 dihasilkan persamaan : (9)

$$w_{km}^n \longleftarrow w_{km}^n - \mu \delta_m^n x_m^{n-1}$$ (10)

μ = konstanta

10. Neuron ke m dari persamaan definisi 9 :

$$\delta_m^N = -2(y_m^N - x_m^N) f'(s_m^N)$$ (11)

y_m^N = target output

$E = (y_m^N - x_m^N)^2$ = target output

11. Neuron di hidden layer , persamaannya adalah :

$$\delta_k^n = \sum_m w_{km}^{n+1} \delta_m^{n+1}$$ (12)

Kesimpulannya adalah no 9, 10, 11 adalah persamaan kunci di dalam penurunan rumus yang dapat digunakan sebagai pembaharuan bobot jaringan sehingga error bisa didapatkan ke nilai minimum error (Alisabana, 2011)

5.3 Prosedur / Algoritma *Backpropagation*

Algoritma Backpropagation suatu algoritma yang menentukan tingkat error yang sangat kecil dengan bobot berdasarkan target output yang kita inginkan. Backpropagation pada umumnya mempunyai langkah-langkah utama yaitu *input* layer, *hidden* layer dan *output* layer. Dalam algoritma backpropagation mempunyai 3 fase yaitu fase maju, fase mundur dan fase modifikasi.

Prosedur atau langkah-angkah aan dijelasakan pada pembahasan sebagai berikut :

1. Menentukan data atau bobot pertama sampai data ke Sembilan, menentukan banyaknya bias di input layer serta hidden layer, menentukan target yang ingin dicapai.
2. Menghitung nilai V1_1 dengan cara mengalikan nilai bias dengan nilai W01_1, nilai X dengan X1.1, nilai Y dengan X2.1 lalu menjumlahkan semua perkalian tadi untuk mengetahui hasil dari nilai V1_1
3. Setelah melakukan perhitungan nilai V1_1 lalu melanjutkan perhitungan untuk nilai V01 dengan cara mengaktifkan nilai V1_1 menggunakan rumus fungsi Simoid
4. Ulangi perhitungan seperti nomor 2 dan 3 untuk mencari nilai V2_2 sampai V4_4 untuk mengetahui nilai tersebut.
5. Menghitung nilai Z1_1 dengan cara mengalikan nilai V01 dan V01.1 , V02 dan V02.1 , V03 dan V03.1 , V04 dan V04.1 . lalu dijumlahkan keseluruhan perkalian tersebut dan ditambahkan dengan nilai bias (W02.1)
6. Setelah mendapatkan nilai Z1_1 lakukan pengaktifan nilai dengan menggunakan rumus fungsi Sigmoid

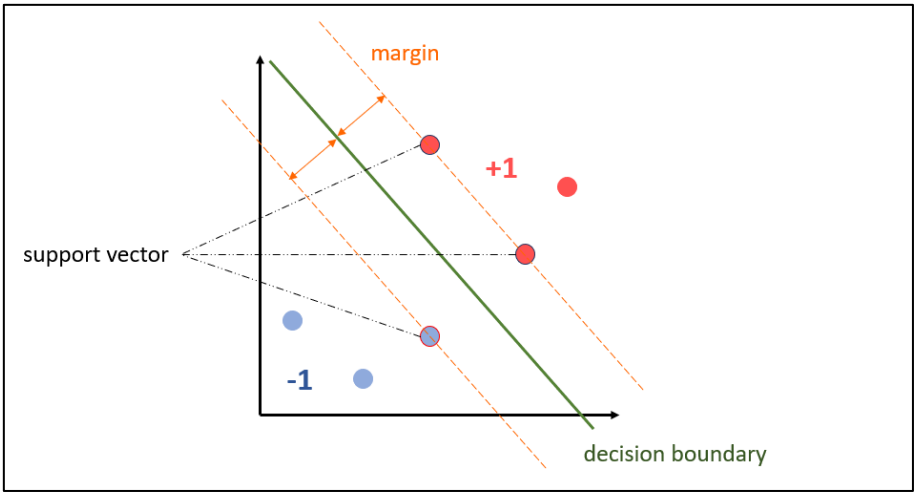
7. Setelah itu lakukan perhitungan nilai Error T1.1 dengan cara nilai target 1 – nilai Z1, nilai Error T1.2 dengan cara nilai target 0 – nilai Z2, nilai Error T1.3 dengan cara nilai target 0 – nilai Z3
8. Selanjutnya mencari kuadrat Error dengan cara mengkuadratkan nilai T1.1, T1.2, T1.3
9. Setelah itu lakukan pencarian nilai perhitungan Delta1_1 dengan cara nilai Error T1.1 x nilai Z1 x (1 - Z1), Delta1_2 dengan cara nilai Error T1.2 x nilai Z2 x (1 – Z2), Delta1_3 dengan cara nilai Error T1.3 x nilai Z3 x (1 - Z3)
10. Hitung Proses Backpropagation ke Hidden Layer Target, menghitun nilai Bias ΔW02_1 dengan cara α * δ1_1 * δ1_2 * δ1_3. Lakukan perhitungan yang sama untuk mencari nilai ΔW02_2 dan ΔW02_3. untuk mencari nilai ΔV01.1 lakukan perhitungan α * δ1_1 * δ1_2 * δ1_3 * V01, nilai ΔV02.1 lakukan perhitungan α * δ1_1 * δ1_2 * δ1_3 * V02 dst… sampai ΔV04.3
11. Untuk penghitungan nilai (δ in Output Layer -> Hiden Layer) lakukan perhitungan δ IN 1_T1 dengan cara (V01.1+V01.2+V01.3 x δ1_1 + δ1_2 + δ1_3) lakukan perhitungan sampai δ IN 4_T1
12. Selanjutnya lakukan perhitungan (AKTIVASI δ) δ1_T1 dengan cara (δ IN 1_T1 * V01 * (1 – V01)) lakukan perhitungan sampai δ4_T1
13. Setelah melakukan perhitungan diatas lakukan perhitungan baru yaitu PROSES BACKPROPAGATION HIDEN LAYER -> INPUT LAYER dengan cara (α * V1_1 * δ1_T1) lakukan perhitungan sampai ΔX2.4.
14. Lakukan proses perhitungan PROSES BACKPROPAGATION HIDEN LAYER -> INPUT LAYER denan cara ΔW01.1 = α * δ1_T1. Lakukan perhitungan sampai ΔW01.4
15. Setelah melakukan proses backpropagation lakukan penjumlahan akan menghasilkan nilai baru untuk proses ke iterasi selanjutnya, untuk nilai baru X, lakukan penjumlahan anatara nilai awal X + nilai delta X. cth: X1.1 + ΔX1.1 = nilai X1.1 baru. Lakukan hingga terakhir nilai X2.4 baru (X pada Input Layer ke Hiden Layer).
16. Sama dengan mencari nilai W baru, nilai W baru didapatkan dengan penjumlahan antara nilai W lama + delta W. cth : W01.1 + ΔW01.1 = nilai W01.1 baru. Lakukan hingga nilai W01.4 (W pada Input Layer ke Hiden Layer).
17. Kemudian lakukan penjumlahan untuk mendapatkan nilai V baru, yaitu nilai V lama + nilai delta V. cth: V01.1 + ΔV01.1 = nilai V01.1 baru. Lakukan hingga semua nilai V terupdate (V Hiden Layer ke Output Layer).
18. Dan yang terakhir update nilai W. untuk mendapatkan nilai W baru lakukan penjumlahan nilai W lama + delta W baru. Cth: W02.1 + ΔW02.1 = nilai W02.1 baru. Lakukan hingga nilai W02.3. (W Hiden Layer ke Output Layer).
19. Lakukan kembali iterasi untuk mendapatkan nilai error tolenrasi yang baru hingga error target terpenuhi.

BAB 6. SUPPORT VECTOR MACHINE

6.1 Pendahuluan

Support Vector Machine (SVM) merupakan salah satu algoritma klasifikasi yang mampu mengklasifikasikan data ke dalam dua kelas berbeda, yakni kelas +1 dan kelas -1. Walaupun sama-sama membagi data ke dalam dua kelas berbeda, SVM tidaklah sama dengan *perceptron*. SVM memanfaatkan data yang paling dekat dengan kelas lainnya sebagai patokan pembuatan *decision boundary* (garis pemisah antar kelas). Data yang dijadikan sebagai patokan tersebut dinamakan sebagai *support vector*.

Pada SVM, jarak *decision boundary* terhadap tiap kelas merupakan hasil bagi dua jarak *support vector* suatu kelas terhadap *support vector* kelas lainnya. Jarak *decision boundary* ini umumnya dikenal dengan nama *margin*. *Margin* pada SVM sengaja dibuat sebesar mungkin mungkin agar tidak terjadi kesalahan pada proses klasifikasi data.



Gambar 6. 1 Support Vector Machine

6.2 SVM Linear

6.2.1 Pendahuluan

SVM linear adalah penerapan SVM pada data yang sifatnya linear separable atau data yang dapat diklasifikasikan langsung secara linier (menggunakan garis).

6.2.2 Persamaan Matematis SVM Linear

Model (persamaan matematis) regresi linear secara umum dapat dituliskan sebagai berikut:

$$w_n\overline{S_nS_m} = y \quad n = 1,2,3;\ m = 1,2,3 \quad \text{.....(6.1)}$$

dimana:

- w = Bobot *support vector*
- S = *Support vector*
- y = Kelas *support vector* (+1 atau -1)

Bobot *decision boundary* dapat dicari menggunakan persamaan berikut:

$$\bar{w} = \sum w_n S_n \quad \text{.....(6.2)}$$

dimana $\bar{w}$ merupakan bobot *decision boundary*. Selanjutnya, klasifikasi *data testing* (prediksi) dapat dilakukan menggunakan persamaan berikut:

$$class \begin{cases} +1 & if \quad \bar{w} \cdot (x_1, \dots, x_n) > b \\ -1 & otherwise \end{cases} \quad \text{.....(6.3)}$$

dimana:

- x_n= nilai fitur data testing yang diklasifikasi.
- b = bias *decision boundary*

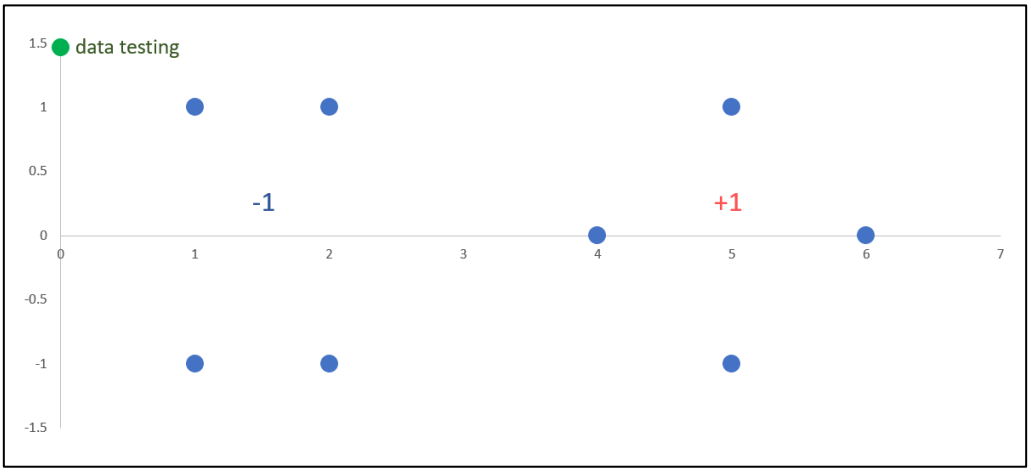
6.2.3 Prosedur / Algoritma SVM Linear

Prosedur atau algoritma SVM linear adalah sebagai berikut:

1. Tentukan data yang menjadi *support vector* dalam *data training*.
2. Tentukan nilai *support vector* dan nilai bias-nya.
3. Gunakan persamaan (6.1) untuk membentuk model persamaan *support vector*.
4. Gunakan metode Eliminasi Gauss Jordan terhadap model persamaan *support vector* untuk mendapatkan bobot tiap *support vector*.
5. Tentukan bobot dan bias *decion boundary* dengan menggunakan persamaan (6.2).
6. Buat model klasifikasi data menggunakan persamaan (6.3).
7. Klasifikasikan *data testing* menggunakan model klasifikasi yang telah dibuat.

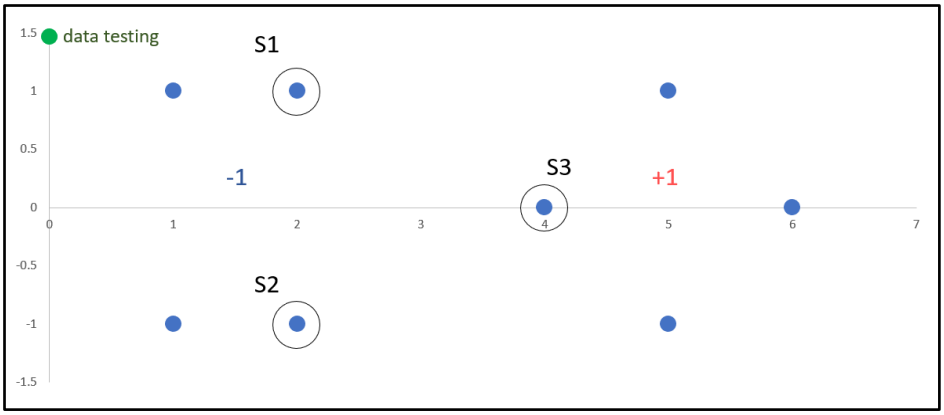
6.2.4 Sample Case (homevideotutor, Support Vector Machines (SVM) - Part 1 - Linear Support Vector Machines, 2014)

Klasifikasikan *data testing* pada gambar 6.1 menggunakan metode SVM.



Gambar 6. 2 Scatter Dataset

Terdapat tiga buah *support vector* pada gambar 6.2, yaitu *support vector* 1, 2, dan 3 (S1, S2, dan S3). Ketiga *support vector* tersebut dapat dilihat pada gambar 6.3.



Gambar 6. 3 Tiga Buah Support Vector

Susun fitur (x_1, x_2) ketiga *support vector* tadi ke dalam bentuk *vector*, kemudian tambahkan nilai bias (bias = 1) pada *vector* bentukan tersebut.

$$S_n = \begin{bmatrix} x_1 \\ x_2 \\ bias \end{bmatrix}, \qquad S_1 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \qquad S_3 = \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}$$

Input *vector* bentukan S1, S2, dan S3 ke dalam persamaan (6.1).

$$w_1 \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + w_2 \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} + w_3 \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} = -1$$

$$w1 \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + w2 \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} + w_3 \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}\begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} = -1$$

$$w1 \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + w2 \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} + w_3 \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} = 1$$

Ubah persamaan *vector* bentukan (*support vector*) ke dalam persamaan skalar.

$$6\,w_1 + 4\,w_2 + 9\,w_3 = -1$$
$$4\,w_1 + 6\,w_2 + 9\,w_3 = -1$$
$$9\,w_1 + 9\,w_2 + 17\,w_3 = 1$$

Selesaikan ketiga persamaan skalar tersebut menggunakan Eliminasi Gauss Jordan untuk mendapatkan nilai bobot tiap *support vector* (w_1, w_2, w_3).

$$\begin{bmatrix} 6 & 4 & 9 \\ 4 & 6 & 9 \\ 9 & 9 & 17 \end{bmatrix}^{-1} X \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

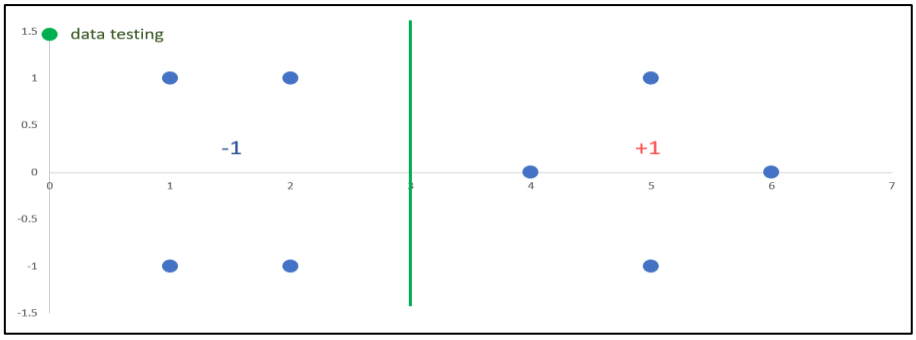
$$\begin{bmatrix} 1.3125 & 0.8125 & -1.125 \\ 0.8125 & 1.3125 & -1.125 \\ -1.125 & -1.125 & 1.25 \end{bmatrix} X \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -3.25 \\ -3.25 \\ 3.5 \end{bmatrix}$$

Berdasarkan hasil Eliminasi Gauss Jordan didapatkan nilai $w_1 = w_2 = -3.25$ dan $w_3 = 3.5$. Input ketiga nilai *w* tersebut ke dalam persamaan (6.2) untuk mendapatkan nilai bobot dan bias *decision boundary*.

$$-3.25 \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + -3.25 \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} + 3.5 \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} w \\ w \\ b \end{bmatrix}$$

$$-3.25 \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + -3.25 \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} + 3.5 \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -3 \end{bmatrix}$$

Gambar *decision boundary* (garis pemisah antarkelas) dalam *scatter* data berdasarkan nilai bobot (1,0) dan biasnya (-3). *Decision boundary* yang didapat dari hasil perhitungan dapat dilihat pada gambar 6.4.



Gambar 6. 4 *Decision Boundary* yang Didapat Berbentuk Garis Vertikal

Data testing dapat diklasifikasikan menggunakan persamaan (6.3) yang dibentuk menggunakan nilai bobot dan bias *decision boundary*.

$$class \begin{cases} +1 & if \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot (x_1, \dots, x_n) > 3 \\ -1 & otherwise \end{cases}$$

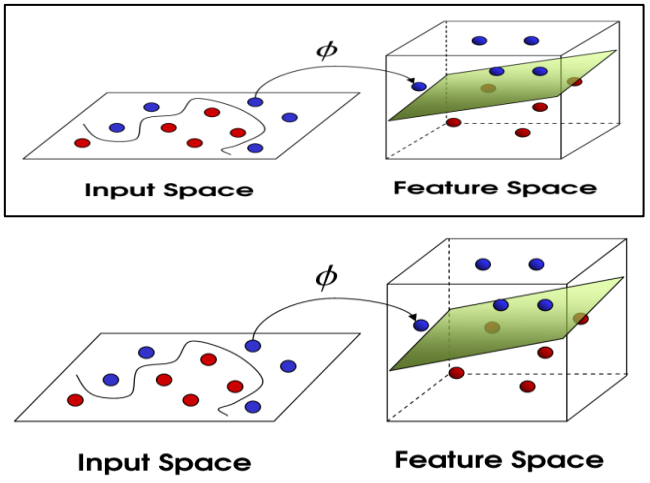
$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

$0 < 3$ maka *data testing* yang diuji terletak/terdapat pada kelas -1.

6.3 SVM Non-Linear

6.3.1 Pendahuluan

Pada dasarnya SVM merupakan metode/algoritma klasifikasi linier. Sebelum melakukan klasifikasi data non-linear menggunakan SVM perlu dilakukan perubahan dimensi data terlebih dahulu. *Mapping function* umumnya sering digunakan pada proses ini. Contoh perubahan dimensi data adalah mengubah data dua dimensi menjadi data tiga dimensi. Selain meningkatkan dimesi data, mengubah representasi data juga termasuk ke dalam proses perubahan dimensi data.



Gambar 6. 5 Contoh Perubahan Dimensi Data Dua Dimensi Menjadi Tiga Dimensi (Jordan, 2017)

6.3.2 Persamaan Matematis SVM Non-Linear

SVM non-linear memiliki model (persamaan) matematis yang hamper sama dengan persamaan SVM linear. Hal yang membedakan keduanya hanyalah penggunaan *mapping function*. SVM non-linear menggunakan *mapping function* untuk mengubah dimensi data sebelum proses klasifikasi dilakukan. Contoh *mapping function* dapat dilihat dalam persamaan (6.4).

$$\phi\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{cases}6 - x_1 + (x_1 - x_2)^2 & if \quad \sqrt{{x_1}^2 + {x_2}^2} \geq 2\\ 6 - x_2 + (x_1 - x_2)^2 & \\ \begin{pmatrix}x_1\\x_2\end{pmatrix} & otherwise\end{cases} \quad \text{.....(6.4)}$$

dimana:

- ϕ = *Mapping function*
- x_n = Fitur data ke-n

Selain itu, pada SVM non-linear juga terdapat penambahan *mapping function* pada persamaan (6.3) sehingga menjadi:

$$class \begin{cases}+1 & if \quad \bar{w} \cdot \phi(x_1, \dots, x_n) > b\\ -1 & otherwise\end{cases} \quad \text{.....(6.5)}$$

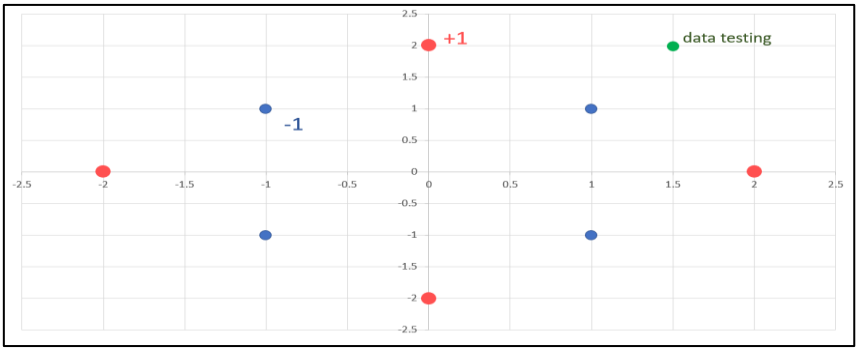
6.3.3 Prosedur / Algoritma SVM Non-Linear

Prosedur atau algoritma SVM non-linear adalah sebagai berikut:

1. Tentukan *mapping function* yang akan digunakan untuk mengubah dimensi data, contoh *mapping function* dapat dilihat dalam persamaan (6.4).
2. Gunakan *mapping function* pada tiap *data training* untuk membentuk dimensi data yang baru.
3. Tentukan data yang menjadi *support vector* dalam *data training*.
4. Tentukan nilai *support vector* dan nilai bias-nya.
5. Gunakan persamaan (6.1) untuk membentuk model persamaan *support vector*.
6. Gunakan metode Eliminasi Gauss Jordan terhadap model persamaan *support vector* untuk mendapatkan bobot tiap *support vector*.
7. Tentukan bobot dan bias *decion boundary* dengan menggunakan persamaan (6.2).
8. Buat model klasifikasi data menggunakan persamaan (6.3).
9. Klasifikasikan *data testing* menggunakan model klasifikasi yang telah dibuat.

6.3.4 Sample Case (homevideotutor, Non Linear Support Vector Machines (Non Linear SVM), 2014)

Klasifikasikan *data testing* pada gambar 6.6 menggunakan metode SVM non-linear.



Gambar 6. 6 Scatter Data Sample Case SVM Non-Linear

Ubah data-data di dalam scatter data pada gambar 6.6 ke dalam bentuk tabel. Tabel hasil perubahan dapat dilihat pada tabel 6.1.

Tabel 6. 1 Tabel Dataset Sample Case SVM Non-Linear

Data	x_1	x_2	Kelas
1	-2	0	+1
2	-1	-1	-1
3	-1	1	-1
4	0	-2	+1
5	0	2	+1
6	1	-1	-1
7	1	1	-1
8	2	0	+1

Tentukan dimensi data yang baru dengan meng-input setiap fitur data (x_1, x_2) ke dalam persamaan (6.4) (*mapping function*).

Tabel 6. 2 Menentukan Fitur Baru Berdasarkan Perhitungan *Mapping Function*

Data	x_1	x_2	Kelas	$\sqrt{{x_1}^2 + {x_2}^2}$	$\sqrt{{x_1}^2 + {x_2}^2} \geq 2$
1	-2	0	+1	2	Yes
2	-1	-1	-1	1	-
3	-1	1	-1	1	-
4	0	-2	+1	2	Yes
5	0	2	+1	2	Yes
6	1	-1	-1	1	-
7	1	1	-1	1	-
8	2	0	+1	2	Yes

Data 1:

$$\begin{pmatrix} {x_1}' \\ {x_2}' \end{pmatrix} = \begin{pmatrix} 6 - (-2) + (-2 - 0)^2 \\ 6 - 0 + (-2 - 0)^2 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \end{pmatrix}$$

Data 4:

$$\begin{pmatrix} {x_1}' \\ {x_2}' \end{pmatrix} = \begin{pmatrix} 6 - 0 + (0 - (-2))^2 \\ 6 - (-2) + (0 - (-2))^2 \end{pmatrix} = \begin{pmatrix} 10 \\ 12 \end{pmatrix}$$

Data 5:

$$\begin{pmatrix} {x_1}' \\ {x_2}' \end{pmatrix} = \begin{pmatrix} 6 - 0 + (0 - 2)^2 \\ 6 - 2 + (0 - 2)^2 \end{pmatrix} = \begin{pmatrix} 10 \\ 8 \end{pmatrix}$$

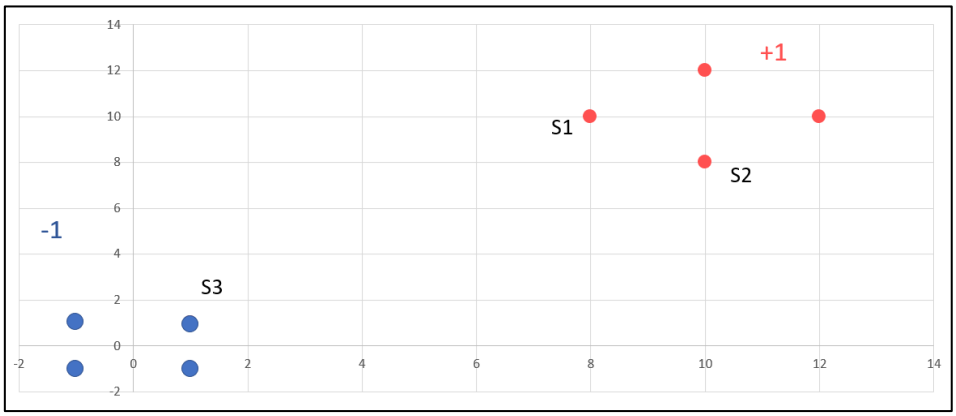
Data 8:

$$\begin{pmatrix} {x_1}' \\ {x_2}' \end{pmatrix} = \begin{pmatrix} 6 - 2 + (2 - 0)^2 \\ 6 - 0 + (2 - 0)^2 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \end{pmatrix}$$

Proses perhitungan menggunakan *mapping function* akan menghasilkan fitur data baru $({x_1}'$dan ${x_2}')$. Fitur data ini merupakan fitur yang mendeskripsikan dimensi data yang baru.

Tabel 6. 3 Fitur Data Baru (x_1'dan x_2') yang Didapatkan

Data	x_1	x_2	x_1'	x_2'	Kelas
1	-2	0	12	10	+1
2	-1	-1	-1	-1	-1
3	-1	1	-1	1	-1
4	0	-2	10	12	+1
5	0	2	10	8	+1
6	1	-1	1	-1	-1
7	1	1	1	1	-1
8	2	0	8	10	+1



Gambar 6. 7 Dimensi Data yang Baru

Setelah melalui proses perubahan dimensi data menggunakan *mapping function*, dataset *sample case* menjadi lebih mudah untuk diklasifikasikan (lihat gambar 6.7). Langkah selanjutnya adalah menentukan *support vector* pada dimensi data yang baru tersebut. Berdasarkan gambar 6.6 didapatkan tiga titik data yang dapat dijadikan sebagai *support vector*, yaitu titik S1, S2, dan S3.

Susun fitur (x_1', x_2') ketiga *support vector* tadi ke dalam bentuk *vector*, kemudian tambahkan nilai bias (bias = 1) pada *vector* bentukan tersebut.

$$S_n = \begin{bmatrix} x_1' \\ x_2' \\ bias \end{bmatrix}, \qquad S_1 = \begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix} \quad S_2 = \begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix} \quad S_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Input *vector* bentukan S1, S2, dan S3 ke dalam persamaan (6.1).

$$w_1 \begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix}\begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix} + w_2 \begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix}\begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix} + w_3 \begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1$$

$$w1 \begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix}\begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix} + w2 \begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix}\begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix} + w_3 \begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1$$

$$w1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix} + w2 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix} + w_3 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = -1$$

Ubah persamaan *vector* bentukan (*support vector*) ke dalam persamaan skalar.

$$165\ w_1 + 161\ w_2 + 19\ w_3 = 1$$
$$161\ w_1 + 165\ w_2 + 19\ w_3 = 1$$
$$19\ w_1 + 19\ w_2 + 3\ w_3 = -1$$

Selesaikan ketiga persamaan skalar tersebut menggunakan Eliminasi Gauss Jordan untuk mendapatkan nilai bobot tiap *support vector* (w_1, w_2, w_3).

$$\begin{bmatrix} 165 & 161 & 19 \\ 161 & 165 & 19 \\ 19 & 19 & 3 \end{bmatrix}^{-1} X \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$\begin{bmatrix} 0.1309 & -0.1191 & -0.0742 \\ -0.1191 & 0.1309 & -0.0742 \\ -0.0742 & -0.0742 & 1.2734 \end{bmatrix} X \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.0859 \\ 0.0859 \\ -1.4219 \end{bmatrix}$$

Berdasarkan hasil Eliminasi Gauss Jordan, didapatkan nilai $w_1 = w_2 = 0.0859$ dan $w_3 = -1.4219$. Input ketiga nilai w tersebut ke dalam persamaan (6.2) untuk mendapatkan nilai bobot dan bias *decision boundary*.

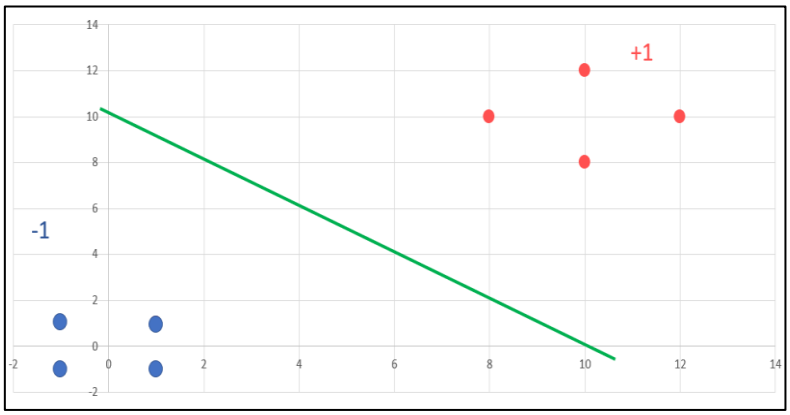
$$0.0859 \begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix} + 0.0859 \begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix} + -1.4219 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} w \\ w \\ b \end{bmatrix}$$

$$0.0859 \begin{bmatrix} 8 \\ 10 \\ 1 \end{bmatrix} + 0.0859 \begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix} + -1.4219 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.125 \\ -1.25 \end{bmatrix}$$

Lakukan normalisasi terhadap hasil perhitungan bobot dan bias *decision boundary* untuk lebih menyederhanakan nilai keduanya.

$$\begin{bmatrix} 0.125 & : & 0.125 \\ 0.125 & : & 0.125 \\ -1.25 & : & 0.125 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -10 \end{bmatrix}$$

Gambar *decision boundary* (garis pemisah antarkelas) dalam dimensi data yang baru berdasarkan nilai bobot (1,1) dan biasnya (-10). *Decision boundary* yang didapat dari hasil perhitungan dapat dilihat pada gambar 6.8.



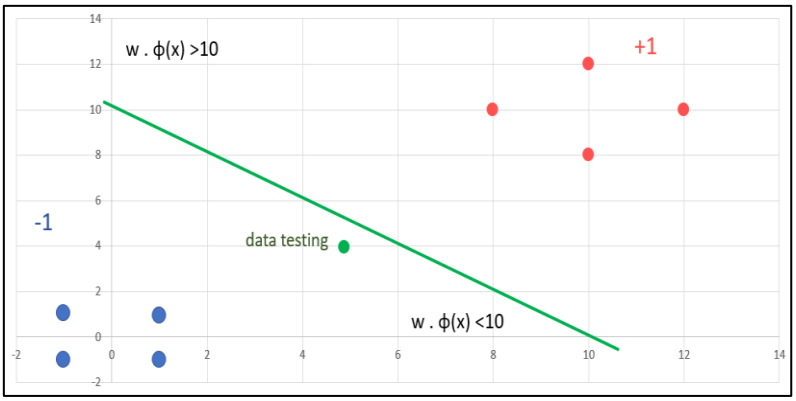
Gambar 6. 8 *Decision Boundary* pada Dimensi Data yang Baru

Data testing dapat diklasifikasikan menggunakan persamaan (6.5) yang dibentuk dari nilai bobot dan bias *decision boundary*.

$$class \begin{cases} +1 & if \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \phi(x_1, \dots, x_n) > 10 \\ -1 & otherwise \end{cases}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \phi \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 4.75 \\ 4.25 \end{bmatrix} = 9$$

$9 < 10$ maka *data testing* yang diuji terletak/terdapat pada kelas -1.



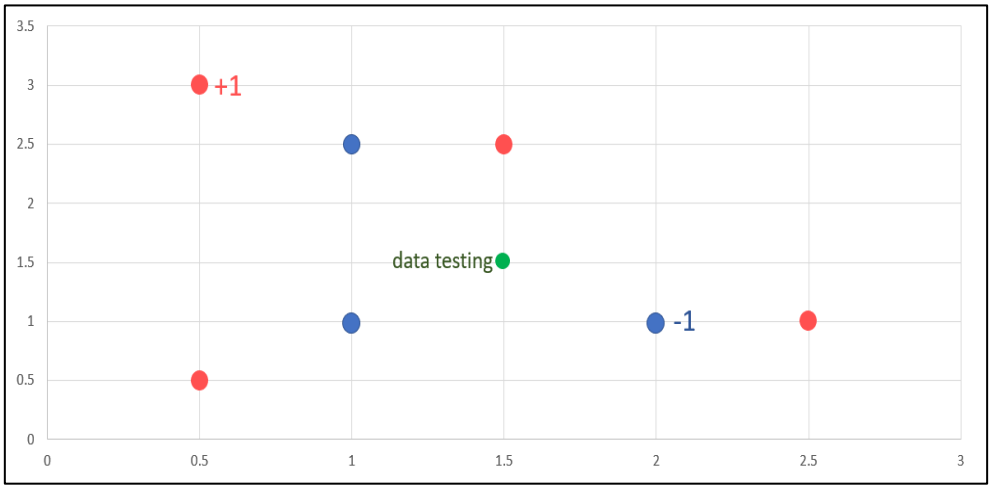
Gambar 6. 9 Hasil Klasifikasi Data Testing Pada Dimensi Data yang Baru

6.4 Ringkasan SVM

1. Support Vector Machine (SVM) merupakan sebuah algoritma klasifikasi data.
2. SVM mampu mengklasifikasikan data ke dalam du akelas berbeda, yakni +1 dan -1.
3. SVM bersifat linier.
4. Untuk mengklasifikasikan data non-linier menggunakan SVM, dibutuhkan tambahan sebuah fungsi yang bernama mapping function.
5. Mapping function berguna untuk mengubah dimensi data (mendapatkan fitur data baru).
6. Perubahan dimensi data dapat berupa peningkatan dimensi ataupun perubahan representasi.

6.5 Latihan SVM

Klasifikasikan *data testing* pada gambar 6.10 menggunakan metode SVM non-linear.



Gambar 6. 10 Scatter Data Latihan SVM

BAB 7. SUPPORT VECTOR MACHINE BERBASIS KERNEL

7.1 Pendahuluan

SVM (*Support Vector Machine*) berbasis kernel murupakan pengembangan lebih lanjut dari SVM biasa yang telah dibahas pada bab sebelumnya. Pada SVM biasa, pengklasifikasian data dilakukan dengan berpedoman pada *support vector* dalam dataset. Jika model klasifikasi tidak ditemukan, maka perlu dilakukan perubahan dimensi data menggunakan *mapping function*. *Mapping function* harus dibuat sesuai dengan *dataset* agar hasil klasifikasi sesuai dengan keiinginan.

Berbeda dengan SVM biasa, SVM berbasis kernel mampu menggabungkan proses perhitungan *feature space* dan proses perubahan dimensi data sehingga lebih menyederhanakan proses klasifikasi. Dikarenakan kemudahan yang ditawarkan, SVM berbasis kernel umumnya juga dikenal dengan sebutan "*kernel trick*". SVM ini menggunakan fungsi kernel dalam mengklasifikasikan data (tidak lagi menggunakan *support vector* dan *mapping function*).

Fungsi kernel merupakan sebuah model klasifikasi atau pengenalan pola. Terdapat banyak sekali jenis fungsi kernel yang dapat dipilih sesuai kebutuhan (sesuai data yang akan diklasifikasikan). Berbeda dengan *mapping function* yang harus dibuat menyesuaikan *dataset*, fungsi kernel telah dibuat secara standar sehingga tinggal diaplikasikan ke dalam *dataset*. Pada bab ini hanya akan dibahas empat fungsi kernel yang umum digunakan ketika buku ini disusun, yakni: *Linear*, *Polynomial*, *Radial Basis Function* (RBF), dan *Sigmoid*.

7.2 Persamaan Matematis Kernel (Raudlatul Munawarah, 2016)

Model (persamaan matematis) kernel secara umum dapat dituliskan sebagai berikut:

$K(x_1,\ x_2, \dots, x_n)$ ..... (7.1)

dimana:

- $K(\dots)$ = Fungsi kernel
- x_n = Fitur data (*feature space*) ke-n

Terdapat empat model/fungsi kernel yang umum digunakan saat ini, yakni antara lain:

1. Linear Kernel

   $K(x_1, x_2) = x_1 \,.\, x_2$ ..... (7.2)

2. Polynimial Kernel

   $K(x_1, x_2) = (x_1 \,.\, x_2 \ + \ c)^d$ ..... (7.3)

   dimana:

   - c = Konstanta
   - d = Derajat polinomial

3. Radial Basis Function (RBF)

   $K(x_1, x_2) = \exp\left(-\frac{\|\overline{x_1} - \overline{x_2}\|}{2\sigma^2}\right)$ ..... (7.4)

   dimana:

   σ = Sigma

4. Sigmoid

$K(x_1, x_2) = \tanh(K\langle \overline{x_1} \,.\, \overline{x_2} \rangle + \vartheta)$ .....(7.5)

dimana:

ϑ = Theta

Selain model (persamaan matematis) di atas, masih terdapat model (persamaan matematis) lain yang digunakan dalam proses pencarian fungsi keputusan, diantaranya:

1. Persamaan untuk menghitung nilai tiap elemen matriks *Hessian*

$D_{ij} = y_i y_j \left(K\left(\overline{x_i} \cdot \overline{x_j}\right)\right) + \lambda^2$ .....(7.6)

dimana:

- D_{ij} = Nilai elemen matriks *Hessian* baris ke-i kolom ke-j
- y_i = Kelas *data training* ke-i (+1 atau -1)
- y_j = Kelas *data training* ke-j (+1 atau -1)
- λ = Lambda

2. Persamaan untuk menghitung nilai *error*

$E_i = \sum \alpha_i \, D_{ij}$ .....(7.7)

dimana:

- E_i = Nilai *error* baris ke-i
- α_i = Alpha/lagrange multiplier baris ke-i

3. Persamaan untuk menghitung nilai delta alpha

$\delta\alpha_i = min\{\, max[\gamma(1 - E_i), -\alpha_i] \,,\, C - \alpha_i \,\}$ .....(7.8)

dimana:

- $\delta\alpha_i$= Nilai delta alpha baris ke-i
- γ = Gamma
- C = Nilai parameter

4. Persamaan untuk menghitung nilai alpha baru

$\alpha_i \, new = \alpha_i \, old + \delta\alpha_i$ .....(7.9)

5. Persamaan untuk menghitung nilai bobot positif dan negatif

$w_{i+} = \alpha_i \, y_i \, K\left(\overline{x_i} \cdot \overline{x_j}\right)$ .....(7.10)

$w_{i-} = \alpha_i \, y_i \, K\left(\overline{x_i} \cdot \overline{x_j}\right)$ .....(7.11)

dimana:

- w_{i+} = Bobot positif (dengan nilai α_i terbesar pada kelas +1)
- w_{i-} = Bobot negatif (dengan nilai α_i terbesar pada kelas -1)

6. Persamaan untuk menghitung nilai bias

$b = -\frac{1}{2} \, (w_{i+} + w_{i-})$ .....(7.12)

dimana:

b = Bias

7. Model umum dalam membentuk fungsi keputusan

$f(x) = \sum \alpha_i \, y_i \, K\left(x_i, x_j\right) + b$ .....(7.13)

dimana:

$f(x)$ = Fungsi keputusan (fungsi klasifikasi)

8. Persamaan untuk mengklasifikasikan *data testing*

$class \begin{cases} +1 & if \quad f(x) \geq +1 \\ -1 & if \quad f(x) \leq -1 \end{cases}$ .....(7.14)

7.3 Prosedur / Algoritma SVM Kernel

Prosedur atau algoritma SVM linear adalah sebagai berikut:

1. Tentukan nilai alpha, gamma, lambda, dan C.
2. Buat matriks $K(x_1, x_2)$ dengan jumlah ordo = jumlah data * jumlah data. Nilai tiap elemen matriks $K(x_1, x_2)$ ditentukan oleh fungsi kernel yang digunakan.
3. Buat matriks *Hessian* dengan ordo = ordo matriks $K(x_1, x_2)$, kemudian hitung nilai tiap elemennya menggunakan persamaan (7.6)
4. Hitung nilai *error* menggunakan persamaan (7.7)
5. Hitung nilai delta alpha menggunakan persamaan (7.8)
6. Hitung nilai alpha baru menggunakan persamaan (7.9)
7. Ulangi langkah empat, lima, dan enam hingga didapatkan nilai alpha yang konvergen
8. Hitung nilai bobot positif dan negatif menggunakan persamaan (7.10) dan (7.11)
9. Hitung nilai bias menggunakan persamaan (7.12).
10. Bentuk fungsi keputusan (fungsi klasifikasi) menggunakan persamaan (7.13).
11. Input fitur data pada *data testing* ke dalam fungsi keputusan (lakukan normalisasi terhadap hasil perhitungan jika diperlukan), kemudian tentukan kelas *data testing* menggunakan persamaan (7.14).

7.4 Sample Case

Tentukan kelas data uji pada tabel 1.1. Gunakan fungsi kernel linear dalam proses perhitungan!

Tabel 7. 1 Sample Case Dataset

Data	x_1	x_2	Kelas (y)
1	0	7	1
2	0	3	-1
3	1	1	-1
4	1	3	-1
5	1	8	1
6	2	4	-1
7	2	7	1
8	3	2	-1
9	3	5	1
10	4	2	1
Uji	2	3	**?**

Tentukan nilai awal:

- α = 0.5;
- γ = 0.5;
- λ = 0.5; dan
- C = 1

Kemudian buat matriks $K(x_1, x_2)$ dengan ordo: jumlah data * jumlah data = 10*10 (10 baris dan 10 kolom). Selanjutnya hitung nilai tiap elemen matriks $K(x_1, x_2)$ persamaan (7.2). Matriks $K(x_1, x_2)$ yang telah dibuat dapat dilihat pada tabel 7.2.

Tabel 7. 2 Matriks K(x_1,x_2)

$K(i,j)$	1	2	3	4	5	6	7	8	9	10
1	49.0000	21.0000	7.0000	21.0000	56.0000	28.0000	s49.0000	14.0000	35.0000	14.0000
2	21.0000	9.0000	3.0000	9.0000	24.0000	12.0000	21.0000	6.0000	15.0000	6.0000
3	7.0000	3.0000	2.0000	4.0000	9.0000	6.0000	9.0000	5.0000	8.0000	6.0000
4	21.0000	9.0000	4.0000	10.0000	25.0000	14.0000	23.0000	9.0000	18.0000	10.0000
5	56.0000	24.0000	9.0000	25.0000	65.0000	34.0000	58.0000	19.0000	43.0000	20.0000
6	28.0000	12.0000	6.0000	14.0000	34.0000	20.0000	32.0000	14.0000	26.0000	16.0000
7	49.0000	21.0000	9.0000	23.0000	58.0000	32.0000	53.0000	20.0000	41.0000	22.0000
8	14.0000	6.0000	5.0000	9.0000	19.0000	14.0000	20.0000	13.0000	19.0000	16.0000
9	35.0000	15.0000	8.0000	18.0000	43.0000	26.0000	41.0000	19.0000	34.0000	22.0000
10	14.0000	6.0000	6.0000	10.0000	20.0000	16.0000	22.0000	16.0000	22.0000	20.0000

Keterangan:

- $K(x_1, x_2) = K(i,j) = x_1 i \,.\, x_1 j + x_2 i \,.\, x_2 j$
- i = baris ke-i
- j = kolom ke-j
- $x_n i$ = feature space x_n data ke-i
- $x_n j$ = feature space x_n data ke-j

Buat matriks *Hessian* dengan ordo = ordo matriks $K(x_1, x_2)$. Nilai tiap elemen matriks *Hessian* dapat dicari menggunakan persamaan (7.6). Matriks *Hessian* yang telah dibuat dapat dilihat pada tabel 7.3.

Tabel 7. 3 Matriks Hessian

12.2500	-5.2500	-1.7500	-5.2500	14.0000	-7.0000	12.2500	-3.5000	8.7500	3.5000
-5.2500	2.2500	0.7500	2.2500	-6.0000	3.0000	-5.2500	1.5000	-3.7500	-1.5000
-1.7500	0.7500	0.5000	1.0000	-2.2500	1.5000	-2.2500	1.2500	-2.0000	-1.5000
-5.2500	2.2500	1.0000	2.5000	-6.2500	3.5000	-5.7500	2.2500	-4.5000	-2.5000
14.0000	-6.0000	-2.2500	-6.2500	16.2500	-8.5000	14.5000	-4.7500	10.7500	5.0000
-7.0000	3.0000	1.5000	3.5000	-8.5000	5.0000	-8.0000	3.5000	-6.5000	-4.0000
12.2500	-5.2500	-2.2500	-5.7500	14.5000	-8.0000	13.2500	-5.0000	10.2500	5.5000
-3.5000	1.5000	1.2500	2.2500	-4.7500	3.5000	-5.0000	3.2500	-4.7500	-4.0000
8.7500	-3.7500	-2.0000	-4.5000	10.7500	-6.5000	10.2500	-4.7500	8.5000	5.5000
3.5000	-1.5000	-1.5000	-2.5000	5.0000	-4.0000	5.5000	-4.0000	5.5000	5.0000

Keterangan:

- $D_{ij} = y_i y_j \left(K(\overline{x_i} \cdot \overline{x_j})\right) + \lambda^2$
- y_i atau y_j = Kelas data training ke-i atau ke-j
- $K(\overline{x_i} \cdot \overline{x_j})$ = Nilai elemen matriks $K(x_1, x_2)$ baris ke-i dan kolom ke-j
- λ = 0.5

Hitung nilai *error* menggunakan persamaan (7.7). Jumlah *error* akan mengikuti jumlah baris matriks $K(x_1, x_2)$.

Tabel 7. 4 Nilai Error Epoch ke-0

E_1	14.0000
E_2	-6.0000
E_3	-2.3750
E_4	-6.3750
E_5	16.3750
E_6	-8.7500
E_7	14.7500
E_8	-5.1250
E_9	11.1250
E_{10}	5.5000

Hitung nilai delta alpha ($\delta\alpha_i$) menggunakan persamaan (7.8). Jumlah $\delta\alpha_i$ akan mengikuti jumlah *Error*.

Tabel 7. 5 Nilai δα_i Epoch ke-0

$\gamma(1 - E_i)$	$-\alpha_i$	$\max[\gamma(1-E_i), -\alpha_i]$	$C - \alpha_i$	$\min\{\max[\gamma(1 - E_i), -\alpha_i], C - \alpha_i\}$
-6.5000	-0.5000	-0.5000	0.5000	-0.5000
3.5000	-0.5000	3.5000	0.5000	0.5000
1.6875	-0.5000	1.6875	0.5000	0.5000
3.6875	-0.5000	3.6875	0.5000	0.5000
-7.6875	-0.5000	-0.5000	0.5000	-0.5000
4.8750	-0.5000	4.8750	0.5000	0.5000
-6.8750	-0.5000	-0.5000	0.5000	-0.5000
3.0625	-0.5000	3.0625	0.5000	0.5000
-5.0625	-0.5000	-0.5000	0.5000	-0.5000
-2.2500	-0.5000	-0.5000	0.5000	-0.5000

Hitung nilai alpha baru ($\alpha_i\ new$) menggunakan persamaan (7.9)

Tabel 7. 6 Nilai Alpha Baru Epoch ke-0

$\alpha_1\ new$	0.0000
$\alpha_2\ new$	1.0000
$\alpha_3\ new$	1.0000
$\alpha_4\ new$	1.0000
$\alpha_5\ new$	0.0000

$\alpha_6\ new$	1.0000
$\alpha_7\ new$	0.0000
$\alpha_8\ new$	1.0000
$\alpha_9\ new$	0.0000
$\alpha_{10}\ new$	0.0000

Untuk melihat apakah nilai alpha telah konvergen perlu dilakukan perbandingan nilai alpha baru pada *epoch* ke-n (*epoch* saat ini) dengan *epoch* ke-$n+1$ (*epoch* berikutnya). *Epoch* ke-$n+1$ diawali dengan menghitung kembali nilai *error* menggunakan persamaan (7.7), namun kali ini nilai alpha yang digunakan adalah nilai alpha baru pada *epoch* ke-n.

Tabel 7. 7 Nilai Error Epoch ke-1

E_1	-22.7500
E_2	9.7500
E_3	5.0000
E_4	11.5000
E_5	-27.7500
E_6	16.5000
E_7	-26.2500
E_8	11.7500
E_9	-21.5000
E_{10}	-13.5000

Selanjutnya hitung kembali nilai delta alpha ($\delta\alpha_i$) menggunakan persamaan (7.8)

Tabel 7. 8 Nilai δα_i Epoch ke-1

$\gamma(1-E_i)$	$-\alpha_i$	$\max[\gamma(1-E_i), -\alpha_i]$	$C-\alpha_i$	$\min\{\max[\gamma(1-E_i), -\alpha_i], C-\alpha_i\}$
11.8750	0.0000	11.8750	1.0000	1.0000
-4.3750	-1.0000	-1.0000	0.0000	-1.0000
-2.0000	-1.0000	-1.0000	0.0000	-1.0000
-5.2500	-1.0000	-1.0000	0.0000	-1.0000
14.3750	0.0000	14.3750	1.0000	1.0000
-7.7500	-1.0000	-1.0000	0.0000	-1.0000
13.6250	0.0000	13.6250	1.0000	1.0000
-5.3750	-1.0000	-1.0000	0.0000	-1.0000
11.2500	0.0000	11.2500	1.0000	1.0000
7.2500	0.0000	7.2500	1.0000	1.0000

Kemudian hitung kembali nilai alpha baru ($\alpha_i\ new$) menggunakan persamaan (7.9)

Tabel 7. 9 Nilai Alpha Baru Epoch ke-1

$\alpha_1\ new$	1.0000
$\alpha_2\ new$	0.0000
$\alpha_3\ new$	0.0000

$\alpha_4\ new$	0.0000
$\alpha_5\ new$	1.0000
$\alpha_6\ new$	0.0000
$\alpha_7\ new$	1.0000
$\alpha_8\ new$	0.0000
$\alpha_9\ new$	1.0000
$\alpha_{10}\ new$	1.0000

Setelah nilai alpha baru pada *epoch* ke-$n+1$ didapat, bandingkan nilai alpha baru pada epoch ke-n dan *epoch* ke-$n+1$.

Tabel 7. 10 Perbandingan Nilai Alpha Baru Pada Epoch 0 dan Epoch 1

	Epoch 0	Epoch 1
$\alpha_1\ new$	0.0000	1.0000
$\alpha_2\ new$	1.0000	0.0000
$\alpha_3\ new$	1.0000	0.0000
$\alpha_4\ new$	1.0000	0.0000
$\alpha_5\ new$	0.0000	1.0000
$\alpha_6\ new$	1.0000	0.0000
$\alpha_7\ new$	0.0000	1.0000
$\alpha_8\ new$	1.0000	0.0000
$\alpha_9\ new$	0.0000	1.0000
$\alpha_{10}\ new$	0.0000	1.0000

Berdasarkan tabel 7.10, nilai alpha baru pada epoch ke-n (*epoch* 0) dan *epoch* ke-$n+1$ (*epoch* 1) tidaklah sama atau belum konvergen. Lakukan kembali perhitungan nilai *error*, delta alpha, dan alpha baru pada *epoch* berikutnya hingga nilai alpha baru konvergen (bernilai sama). Jika nilai alpha baru belum juga konvergen, namun *epoch* dirasa telah cukup banyak, tentukan batas *epoch* untuk memberhentikan proses perhitungan. Contohnya seperti pada *sample case* ini, perhitungan hanya dilakukan sampai *epoch* ke-6 dikarenakan nilai alpha baru belum juga konvergen.

Tabel 7. 11 Nilai Alpha Baru Epoch ke-6

$\alpha_1\ new$	1.0000
$\alpha_2\ new$	0.0000
$\alpha_3\ new$	0.0000
$\alpha_4\ new$	0.0000
$\alpha_5\ new$	1.0000
$\alpha_6\ new$	0.0000
$\alpha_7\ new$	1.0000
$\alpha_8\ new$	0.0000
$\alpha_9\ new$	1.0000
$\alpha_{10}\ new$	1.0000

Input nilai alpha baru pada tabel 7.9 ke dalam persamaan (7.10) dan (7.11) untuk mencari nilai bobot positif dan negatif.

Tabel 7. 12 Mencari Nilai Bobot Positif (w_(i+)) dan Negatif (w_(i-))

$K(i,j)$	1	2	3	4	5	6	7	8	9	10
1	49.0000	21.0000	7.0000	21.0000	56.0000	28.0000	49.0000	14.0000	35.0000	14.0000
2	21.0000	9.0000	3.0000	9.0000	24.0000	12.0000	21.0000	6.0000	15.0000	6.0000
3	7.0000	3.0000	2.0000	4.0000	9.0000	6.0000	9.0000	5.0000	8.0000	6.0000
4	21.0000	9.0000	4.0000	10.0000	25.0000	14.0000	23.0000	9.0000	18.0000	10.0000
5	56.0000	24.0000	9.0000	25.0000	65.0000	34.0000	58.0000	19.0000	43.0000	20.0000
6	28.0000	12.0000	6.0000	14.0000	34.0000	20.0000	32.0000	14.0000	26.0000	16.0000
7	49.0000	21.0000	9.0000	23.0000	58.0000	32.0000	53.0000	20.0000	41.0000	22.0000
8	14.0000	6.0000	5.0000	9.0000	19.0000	14.0000	20.0000	13.0000	19.0000	16.0000
9	35.0000	15.0000	8.0000	18.0000	43.0000	26.0000	41.0000	19.0000	34.0000	22.0000
10	14.0000	6.0000	6.0000	10.0000	20.0000	16.0000	22.0000	16.0000	22.0000	20.0000
α_i	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	1.0000
kelas (y_i)	1	-1	-1	-1	1	-1	1	-1	1	1
bobot (w_i)	203.0000	87.0000	39.0000	97.0000	242.0000	136.0000	223.0000	88.0000	175.0000	98.0000
w_{i+}	203.0000	-	-	-	242.0000	-	223.0000	-	175.0000	98.0000
w_{i-}	-	87.0000	39.0000	97.0000	-	136.0000	-	88.0000	-	-

Dikarenakan tidak ada nilai a_i terbesar baik pada kelas positif (+1) maupun kelas negatif (-1), maka nilai w_{i+} dan w_{i-} dapat dipilih secara acak dari daftar yang tersedia pada tabel 7.12. Nilai yang dipilih pada *sample case* ini ialah:

- w_{i+} = 98
- w_{i-} = 39

Hitung nilai bias menggunakan persamaan (7.12).

$$b = -\frac{1}{2}\,(98 + 39) = -68.5$$

Bentuk fungsi keputusan (fungsi klasifikasi) menggunakan persamaan (7.13).

$$f(x) = \sum \alpha_i\, y_i\, K(x_i, x_j) + -68.5$$

Tentukan kelas data uji dengan meng-input *feature space* (x_1 dan x_2) data uji ke dalam fungsi keputusan yang telah dibentuk. Lakukan normalisasi hasil perhitungan jika diperlukan.

Tabel 7. 13 Klasifikasi Data Uji (Data Testing)

α_i	y_i	$K(x_i, x_j)$	b	$f(x)$
1.0000	1.0000	42.0000	-68.5	-26.5000
0.0000	-1.0000	18.0000	-68.5	-68.5000
0.0000	-1.0000	10.0000	-68.5	-68.5000
0.0000	-1.0000	22.0000	-68.5	-68.5000
1.0000	1.0000	52.0000	-68.5	-16.5000
0.0000	-1.0000	32.0000	-68.5	-68.5000
1.0000	1.0000	50.0000	-68.5	-18.5000
0.0000	-1.0000	24.0000	-68.5	-68.5000
1.0000	1.0000	42.0000	-68.5	-26.5000
1.0000	1.0000	28.0000	-68.5	-40.5000
				-471.0000
				SIGN(-471)
				-1

Keterangan:

- $K(x_1, x_2)$ = nilai elemen data uji dalam matriks $K(x_1, x_2)$
- $K(x_1, x_2) = K(i, j) = x_1 i \,.\, x_1 j + x_2 i \,.\, x_2 j$
- i = baris ke-i
- j = kolom ke-j
- $x_n i$ = feature space x_n data ke-i
- $x_n j$ = feature space x_n data ke-j
- Ketika mengklasifikasikan data uji, data tersebut juga diikutsertakan dalam matriks $K(x_1, x_2)$ selama proses perhitungan berlangsung, sehingga ordo matriks $K(x_1, x_2)$ menjadi sebelas.

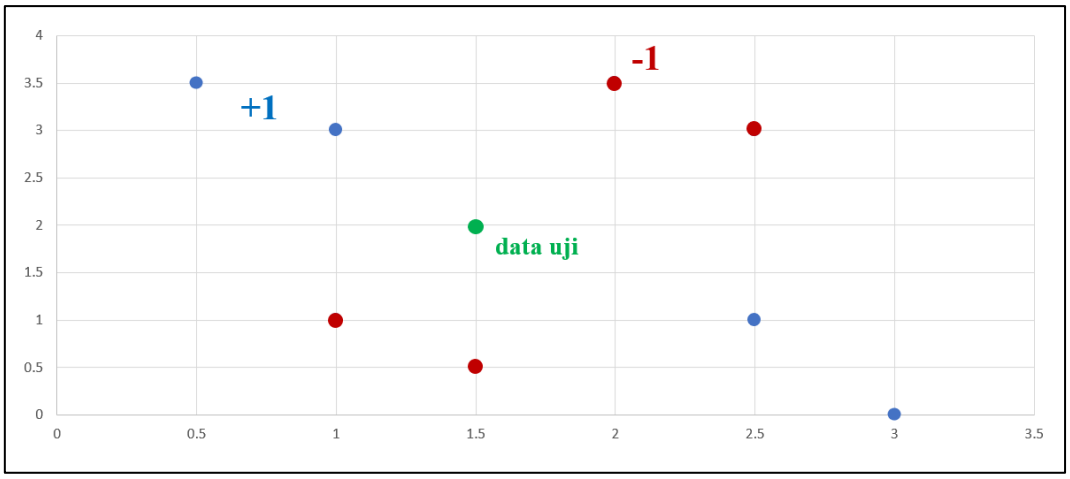
Berdasarkan hasil perhitungan pada tabel 7.13, setelah dilakukan normalisasi, nilai $f(x)$ data uji bernilai -1. Sesuai dengan persamaan (7.14), nilai $f(x)$ = -1 berada pada kelas -1.

7.5 Ringkasan SVM Kernel

1. SVM berbasis kernel merupakan pengembangan lebih lanjut dari SVM biasa.
2. SVM berbasis kernel juga dikenal dengan sebutan "*kernel trick*".
3. SVM berbasis kernel bekerja dengan mengkombinasikan proses perhitungan *feature space* dan proses perubahan dimensi data.
4. SVM berbasis kernel menggunakan fungsi kernel dalam proses klasifikasi data.
5. Fungsi kernel adalah model klasifikasi atau model pengenalan pola yang dibuat secara standar.
6. Terdapat banyak sekali jenis fungsi kernel, namun saat ini hanya terdapat empat fungsi saja yang umum digunakan, yakni: *Linear*, *Polynomial*, *RBF*, dan *Sigmoid*.
7. Fungsi kernel dapat langsung diaplikasikan dalam dataset.

7.6 Latihan SVM Kernel

Tentukan kelas data uji pada gambar 7.1. Gunakan fungsi kernel yang paling sesuai dalam proses perhitungan!



Gambar 7. 1 Scatter Data Latihan SVM Kernel

DAFTAR PUSTAKA

Ahmadi, A. dan Hartati, S. . (2013). Penerapan Fuzzy C-Means dalam Sistem Pendukung Keputusan untuk Penentuan Penerima Bantuan Langsung Masyarakat (BLM) PNPMMPd (Studi Kasus PNPM-MPd Kec. Ngadirojo Kab. Pacitan). 267-268.

Alisabana, H. P. (2011, January 23). *Seputar Jaringan Backpropagation (BPNN)*. Retrieved from matematiku.wordpress.com: http://matematiku.wordpress.com/2011/01/23/backpropagation-learning-algorithm/

Andrijasa, M. (2010). Penerapan Jaringan Syaraf Tiruan Untuk Memprediksi jumlah pengangguran di Provinsi Kalimantan Timur Dengan Menggunakan Algoritma Pembelajaran Backpropagation. *Jurnal Informatika Mulawarman*, 50.

Bezdek, J. C. (1981). Pattern Recognition with Fuzzy Objective Function Algorithms.

cs231n. (2015). *CS231n Convolutional Neural Networks for Visual Recognition*. Retrieved from GitHub: http://cs231n.github.io/neural-networks-1/

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 32-57.

Erna Budhiarta Nababan, M. Z. (2015). Analisis fungsi aktifasi sigmoid binner dan sigmoid bipolar dalam algroitma Backpropagation pada prediksi kemampuan mahasiswa. 105.

Fakhriansyah, K. (Sutradara). (2018). *KELOMPOK 6 | FTSI UNIPDU | Data Meaning & Analitical Business (Clustering "Fuzzy C-Means")* [Gambar Hidup].

Ginanto, N. (2012, November 14). *BACKPROPAGATION*. Retrieved from novikaginanto.wordpress.com: https://novikaginanto.wordpress.com/2012/11/14/backpropagation/

homevideotutor. (2014, 6 3). *Non Linear Support Vector Machines (Non Linear SVM)*. Retrieved from Youtube: https://www.youtube.com/watch?v=6cJoCCn4wuU

homevideotutor. (2014, 1 28). *Support Vector Machines (SVM) - Part 1 - Linear Support Vector Machines*. Retrieved from Youtube: https://www.youtube.com/watch?v=LXGaYVXkGtg

Jordan, J. (2017, 6 20). *Support vector machines.* Retrieved from Jeremy Jordan: https://www.jeremyjordan.me/support-vector-machines/

Kashyap, A. (2018, 6 5). *Quora*. Retrieved from What does an x axis represent in a logistic regression sigmoid function?: https://www.quora.com/What-does-an-x-axis-represent-in-a-logistic-regression-sigmoid-function

Kusumadewi, S. d. (2010). Aplikasi Logika Fuzzy untuk Pendukung Keputusan.

Muhammad Suhaili, Dana Indra Sensuse. (2015). Komparasi Metode Learning Vector Quantization Dengan Metode Fuzzy C-Means. 6.

Munir, R. (2005). *Matematika Diskrit*. Bandung: Informatika.

Nur Indah Selviana, Mustakim. (2016). Analisis Perbandingan K-Means dan Fuzzy C-Means untuk Pemetaan Motivasi Balajar Mahasiswa. *Seminar Nasional Teknologi Informasi, Komunikasi dan Industri (SNTIKI) 8*, 95-104.

Raudlatul Munawarah, O. S. (2016). PENERAPAN METODE SUPPORT VECTOR MACHINE PADA DIAGNOSA HEPATITIS. *Kumpulan Jurnal Ilmu Komputer (KLIK)*, 103-113.

Robot, P. (2006). *Artificial Neural Networks & Robotics*. Retrieved from Pi Robot: http://www.pirobot.org

Sucahyono, H. (2015, 7 10). *Regresi LInear Dengan Excel*. Retrieved from Youtube: https://www.youtube.com/watch?v=pg8lDoWkEFU

Syafrudin. (2015). *Contoh Perhitungan Manual Penerapan Metode K Means Klastering Datamining*. Retrived from https://syafrudinmtop.blogspot.com/2015/10/contoh-perhitungan-manual-kmeans-klastering.html.

Tanjung, P. A. (2016). Penerapan Fuzzy C-Means Clustering Pada Data Nasabah Bank.

Wardhani, A. K. (2016). K-MEANS ALGORITHM IMPLEMENTATION FOR CLUSTERING OF PATIENTS DISEASE IN KAJEN CLINIC OF PEKALONGAN. *JURNAL TRANSFORMATIKA*, 30-37.

Y. A. Lesnussa., S. L. (2015). Aplikasi Jaringan Saraf Tiruan Backpropagation untuk Memprediksi Prestasi Siswa SMA.(Studi kasus: Prediksi Prestasi Siswa SMAN 4 Ambon). 150.

Zimmermann. (2001). *Fuzzy Set Theory and Its Applications, Fourth Edition*.