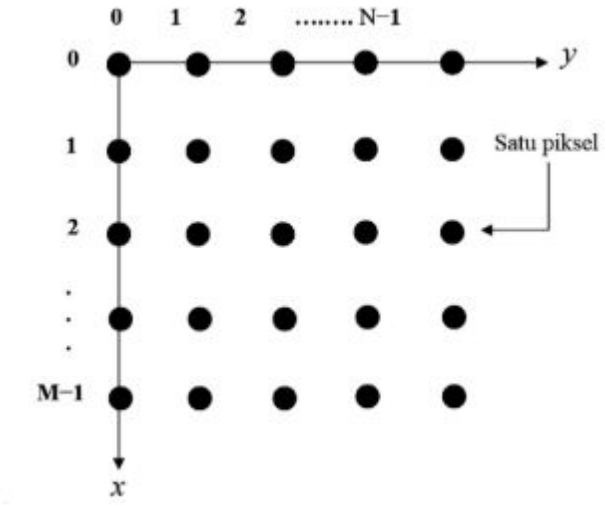
BAB 2
TINJAUN PUSTAKA

2.1 Citra

Citra secara harfiah adalah sebuah gambar pada bidang dwimatra (dua dimensi) [8]. Sebuah citra digital diwakili oleh matriks yang terdiri dari M baris dan N kolom, di mana perpotongan antara baris dan kolom disebut dengan piksel. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel dititik itu [9].



Gambar 2.1 Sistem koordinat pada citra

Artinya, sebuah citra digital dapat ditulis dalam bentuk matriks berikut:

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & \dots & \dots & f(1,M-1) \\ \dots & \dots & \dots & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix} \quad (2.1)$$

Ada beberapa tipe citra yang sering digunakan untuk penelitian, di antaranya adalah :

A. Citra Biner

Di dalam citra biner, tiap-tiap piksel hanya membutuhkan 1 bit memori. Maka dengan demikian, setiap piksel hanya mempunyai 2 buah kemungkinan nilai intensitas, yaitu hitam dan putih.

B. Citra *Grayscale*

Citra *grayscale* adalah matriks data yang nilai-nilainya mewakili intensitas setiap piksel berkisar antara 0 sampai dengan 255. Setiap piksel membutuhkan 8 bit memori.

C. Citra Warna

Citra warna adalah citra yang masing-masing piksel mempunyai 3 (tiga) komponen warna yang spesifik, yaitu komponen merah (red), hijau (green) dan biru (blue). Warna setiap piksel ditentukan oleh kombinasi dari intensitas warna merah, hijau dan biru yang disimpan pada bidang warna di okasi piksel. Format file grafis menyimpan citra warna sebagai citra 24 bit, yang berasal dari komponen merah, hijau dam biru masing-masing 8 bit. Hal ini menyebabkan citra warna mempunyai 24 juta kemungkinan warna.

2.2 *Preprocessing*

Preprocessing adalah proses yang dilakukan untuk meningkatkan kualitas data pada citra sehingga dapat memudahkan dalam pengekstrasian fitur yang nantinya

akan diguunakan pada tahap selanjutnya. Pada tahap awal ini citra akan diubah menjadi *grayscale* dan *resizing*.

2.2.1 *Grayscaling*

Grayscaling adalah proses dimana mengubah jenis citra berwarna menjadi citra *grayscale* (abu-abu). Pengubahan dilakukan dengan mengkalkulasikan tiga *channel* warna merah, hijau dan biru sehingga menjadi satu *channel* yang saja yang mempunyai nilai intensitas dari 0 untuk hitam sampai 255 untuk putih. Pengkalkulasian bisa dilakukan dengan menggunakan perhitungan standar ITU-R 601-2 untuk luminansi. Rumus didefinisikan sebagai berikut:

$$Y' = 0{,}299R' + 0{,}587G' + 0{,}114B' \tag{2.2}$$

Keterangan:

Y’ : Citra *grayscale*

R’ : Komponen merah dari citra RGB

G’ : Komponen hijau dari citra RGB

B’ : Komponen biru dari citra RGB

(a). Citra Sebelum Grayscaling (b). Citra Sesudah Grayscaling

Gambar 2.2 Contoh Citra Grayscaling

2.3 *Skew Correction*

Skew correction adalah proses dimana mengubah citra dengan rotasi untuk memperbaiki jenis citra yang mempunyai kemiringan. Kemiringan citra bisa

mengakibatkan pendeteksian tulisan yang ada pada citra menjadi sulit kenali. Pendeteksian dan pengoreksian tingkat kemiringan citra dilakukan dengan menggunakan metode *horizontal profile projection* dimana pengkoreksian akan dicoba dengan berbagai *angle* dan *angle* dengan jumlah *energy function* maksimal yang akan dijadikan *angle* citra terbaru [10]. Perhitungan *energy function* dilakukan dengan menggunakan persamaan berikut.

$$A(\theta) = \sum_{i=1}^{m} C_i^2\ (\theta) \tag{2.3}$$

Dimana m adalah panjang dari citra dan *ci(θ)* adalah perhitungan histogram pada area pada citra yang sudah dirotasi dengan *angle* θ [11]. Dimana perhitungan *energy function* itu sendiri adalah sebagai berikut [12].

$$h = \left(area_{top} - area_{bottom}\right)^2 \tag{2.4}$$

Setelah semua *energy function* ditemukan, maka *angle* dengan *energy function* terbesarlah yang akan digunakan untuk rotasi terbaik.

(a) Sebelum *Skew Correction* (b) Sesudah *Skew Correction*

Gambar 2.3 Contoh *Skew Correction*

2.4 *Resizing*

Resizing adalah proses yang digunakan untuk mengubah ukuran citra digital dalam pixel. Pengubahan bisa mengubah ukuran citra menjadi lebih kecil dari ukuran sebenarnya ataupun sebaliknya. Salah satu metode yang sering digunakan dalam pengubahan ukuran citra adalah metode *Nearest Neighbor*. *Nearest Neighbor* merupakan metode interpolasi paling sederhana dan cepat dengan memindahkan

ruang yang kosong dengan piksel yang berdekatan (the *nearest neighboring pixel*) pada saat pengecilan atau pembesaran skala gambar [13].

N*earest neighbor* menggunakan nilai piksel terdekat pada gambar awal untuk memberikan nilai piksel pada gambar awal yang akan diperbesar atau diperkecil. Sebagai contoh terdapat sebuah gambar dengan ukuran 4 x 4 dengan jumlah piksel 16 dimana setiap pikselnya diwakilkan dengan nilai A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P. Kemudian gambar akan diperbesar menjadi ukuran 6 x 8 dengan jumlah piksel 48 menggunakan *Nearest neighbor* [14]. Berikut adalah contoh dari *nearest neighbor*:

i / j	1	2	3	4
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P

Gambar 2.4 Nilai Piksel Citra Asli

Adapun contoh proses perhitungan untuk mendapatkan setiap nilai piksel pada gambar dengan ukuran 6 x 8 yaitu:

Perbandingan lebar (*ratio weight)* = 4 : 6 = 2 : 3.

Perbandingan panjang (*ratio height)* = 4 : 8 = 1 : 2.

- Untuk posisi piksel dengan nilai x = 1, y = 1

  $Piksel_x$ = *ceil*(x * *ratio weight*) = *ceil*(1 * 2/3) = 1

  $Piksel_y$ = *ceil*(y * *ratio height*) = *ceil*(1 * ½) = 1

  Nilai piksel pada gambar ukuran 6 x 8 dengan x = 1 yang menghasilkan $Piksel_x$ = 1 dan y = 1 yang menghasilkan $Piksel_y$ = 1 disesuaikan dengan nilai piksel pada gambar awal dengan i = 1 dan j = 1 yaitu A.

- Untuk posisi piksel dengan nilai x = 1, y = 2

  $Piksel_x$ = *ceil*(x * *ratio weight*) = *ceil*(1 * 2/3) = 1

  $Piksel_y$ = *ceil*(y * *ratio height*) = *ceil*(2 * ½) = 1

  Nilai piksel pada gambar ukuran 6 x 8 dengan x = 1 dan y = 2 juga memiliki nilai A.

Ceil (*ceiling*) merupakan proses pembulatan sebuah bilangan ke atas. Sehingga akan menghasilkan citra matrik berikut:

x / y	1	2	3	4	5	6
1	A	B	B	C	D	D
2	A	B	B	C	D	D
3	E	F	F	G	H	H
4	E	F	F	G	H	H
5	I	J	J	K	L	L
6	I	J	J	K	L	L
7	M	N	N	O	P	P
8	M	N	N	O	P	P

Gambar 2.5 Nilai Piksel Hasil *Resize*

2.5 Segmentasi

Segmentasi adalah sebuah proses untuk memisahkan antara objek dan background. Proses segmentasi yang akan dilakukan pada tahap ini adalah binarisasi dengan menggunakan *Bradley-Roth Adaptive Threshold* dan *Profile Projection* secara horizontal dan vertikal untuk memisahkan tulisan perbaris dan perkarakter.

2.5.1 *Bradley-Roth Adaptive Threshold*

Bradley-Roth Adaptive Threshold merupakan sebuah metode untuk melakukan proses binarisasi pada sebuah citra dokumen. Metode *Bradley-Roth* merupakan sebuah metode hasil pengembangan dari metode *Pierre D. Wellner* [15], yang mengemukakan sebuah pengambangan adaptif cepat yang mana mengkalkulasikan hasil pergerakan rata-rata dari piksel terakhir yang dilihat dengan ambang lokal. Metode *Bradley-Roth* menggunakan integral image sebagai citra masukan. Integral image adalah hasil penjumlahan dari area yang ditentukan, teknik ini mampu mempercepat dalam mendapatkan hasil perhitungan ambang. Pembuatan *integral images* didefinisikan sebagai berikut:

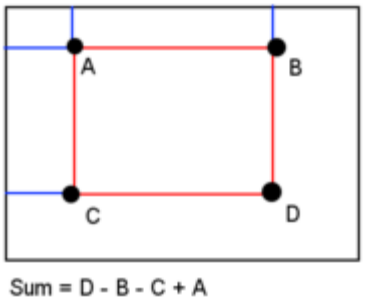
$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',j') \tag{2.5}$$

Keterangan:

i : Matriks citra asli

x',y' : Nilai piksel dari posisi x,y

Ketika tabel *integral images* sudah dibuat, maka untuk mendapatkan jumlah dari suatu area bisa dilakukan dengan persamaan sederhana seperti berikut:



Gambar 2.6 Jumlah dari suatu area

$$i(x,y) = I(D) - I(B) - I(C) + I(A) \tag{2.6}$$

Keterangan:

I : *Integral images*

D : Titik bawah/titik pilihan

A : Titik pojok kiri atas dari area yang dipilih

B : Titik pojok kanan atas dari area yang dipilih

C : Titik kiri bawah dari area yang dipilih

Kunci dari metode *Bradley-Roth* adalah setiap piksel pada citra akan ditetapkan sebagai warna hitam jika tingkat kecerahan dari piksel dibawah sekian persen dari rata-rata kecerahan disekelilingnya, jika sebaliknya maka nilai piksel akan ditentukan sebagai warna putih [16]. Berikut adalah *pseudo code* yang digunakan pada perhitungan *Bradley-Roth.*

$$\left(nilai_{piksel} * jumlah_{pikselarea}\right) < \left(sum * \frac{100-15}{100}\right) \tag{2.7}$$

Gambar 2.7 Hasil binarisasi *Bradley-Roth Adaptive Thresholding*

2.5.2 *Profile Projection*

Metode *Profile Projection* digunakan untuk memisahkan tulisan perbaris dan perkarakter. Metode *Profile Projection* akan menghitung jumlah piksel non-background secara vertikal dan horizontal dan nilai tersebut akan dibandingkan dengan nilai ambang tertentu untuk memisahkan tulisan perbaris, perkata dan perkarakter.

1. *Profile Projection* secara horizontal

   Profile Projection secara horizontal adalah jumlah banyaknya piksel hitam yang tegak lurus dengan sumbu x. *Profile Projection* secara horizontal direpresentasikan dengan suatu vektor Ph berukuran M. *Profile Projection secara* horizontal pada kolom ke-j, yaitu Ph[j], didefinisikan sebagai berikut [17]:

$$P_h[j] = N - \sum_{j=1}^{N} S[i,j] \tag{2.8}$$

2. *Profile Projection* secara vertical

   Profile Projection secara vertikal adalah jumlah banyaknya piksel hitam yang tegak lurus dengan sumbu y. *Profile Projection* secara vertikal direpresentasikan dengan suatu vektor Pv berukuran N. *Profile Projection* secara vertikal pada baris ke-i, yaitu Pv[i], didefinisikan sebagai berikut [17]:

$$P_v[i] = M - \sum_{i=1}^{M} S[i,j] \tag{2.9}$$

   keterangan:

   S : Biner citra

   M : Banyak kolom pada citra

   N : Banyak baris pada citra

2.6 *Support Vector Machine*

Support Vector Machine (SVM) adalah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi – fungsi linier dalam sebuah fitur yang berdimensi tinggi [18] dan dilatih dengan menggunakan algoritma pembelajaran yang didasarkan pada teori optimasi. SVM pertama kali diperkenalkan pada tahun 1992 oleh Boser, Guyon, Vapnik, di *Workshop on Computational Learning Theory* sebagai

rangkaian dari beberapa konsep – konsep unggulan dalam bidang *pattern recognition* [19]. SVM merupakan suatu teknik untuk menemukan *hyperplane* yang bisa memisahkan dua set data dari dua kelas yang berbeda. *Hyperplane* adalah garis batas pemisah data antar kelas, sedangkan margin adalah jarak antara *hyperplane* dengan data terdekat pada masing-masing kelas. Adapun data terdekat dengan *hyperplane* pada masing-masing kelas inilah yang disebut *support vector* [20]. Pada awalnya, SVM digunakan untuk mengklasifikasi data biner saja, sehingga pembagian label biasanya dibagi menjadi dua kelas, yaitu positif dan negatif atau +1 dan -1. Jika diasumsukan data bisa dipisahkan secara linear, maka persamaan untuk pemisah *hyperplane* adalah sebagai berikut:

$$w.x_i + b = 0 \tag{2.10}$$

Data x_i yang tergabung dalam kelas negatif adalah data yang memenuhi pertidaksamaan sebagai berikut:

$$w.x_i + b < 0 \tag{2.11}$$

Adapun data yang tergolong dalam kelas positif adalah data yang memenuhi pertidaksamaan sebagai berikut:

$$w.x_i + b > 0 \tag{2.12}$$

Keterangan:

X_i : data masukan

w : nilai dari bidang normal

b : nilai bias

Parameter w dan b adalah parameter yang nantinya akan dicari nilainya. Meskipun awalnya SVM digunakan untuk mengklasifikasi kelas biner saja, ada beberapa metode yang bisa digunakan untuk mengklasifikasi beberapa kelas sekaligus (*multiclass*), salah satunya adalah dengan pendekatan *One vs One (OVO)*. Pada pendekatan OVO, misalkan permasalahan yang ditemui terdiri dari N kelas. Sehingga

akan dibuat N(N-1)/2 *decision boundary*. *Decision boundary* yang dihasilkan merupakan hasil dari pencarian *hyperplane* dari setiap kelas dengan setiap satu kelas yang lainnya [21]. Tingkat keakurasian SVM juga bergantung terhadap jenis *kernel* dan beberapa parameter lainnya. Berdasarkan dari karakteristiknya, metode SVM dibagi menjadi dua, yaitu SVM Linier dan SVM Non-Linier. SVM linier digunakan pada data yang dipisahkan secara linier, yaitu memisahkan kedua class pada *hyperplane* dengan soft margin. Sedangakan SVM Non-Linier yaitu menerapkan fungsi dari kernel trick terhadap ruang yang berdimensi tinggi [19].

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antar bidang pembatas kedua kelas dan titik terdekatnya, yaitu 2/|w|. Hal ini dirumuskan sebagai permasalahan *quadratic programming* (QP) problem dengan melihat kondisi Karush-Kuhn-Tucker (KKT) yaitu mencari titik minimal persamaan (2.10) dengan memperhatikan pertidaksamaan (2.11) berikut:

$$\min \frac{1}{2} ||\mathrm{w}||^2 \tag{2.13}$$

$$y_i \, (\mathrm{w} * x_i + \mathrm{b}) - 1 \geq 0 \tag{2.14}$$

Persoalan ini dapat dengan mudah diselesaikan dengan mudah ke dalam formula *lagrangian* yang menggunakan *lagrangian multiplier* [22]. Dengan demikian persamaan bisa diubah menjadi persamaan berikut:

$$\text{Min L}(\mathrm{w}, \mathrm{b}, \mathrm{a}) = \frac{1}{2}\left|\left|\mathrm{w}\right|\right|^2 - \sum_{i=1}^{n} a_i y_i \, (w^T x_i + \mathrm{b}) + \sum_{i=1}^{n} a_i \tag{2.15}$$

Dengan memperhatikan persamaan berikut:

$$\mathrm{w} = \sum_{i=1}^{n} a_i y_i x_i \tag{2.16}$$

$$\sum_{i=1}^{n} a_i y_i = 0 \tag{2.17}$$

Model persamaan (2.12) diatas merupakan model *primal Lagrange*. Sedangkan dengan memaksimalkan L terhadap a_i, persamannya menjadi persamaan berikut:

$$\text{Max Ld}(a) \sum_{i=1}^{n} a_{i-\frac{1}{2}} \sum_{i=1,j=i}^{n} a_i a_j y_i y_j^T x_i x_j^T \tag{2.18}$$

$$\sum_{i=1}^{n} a_i y_i = 0, a_i \geq 0 \ (i,j = 1,..,n)$$

Dengan demikian, menggunakan persamaan pada *dual form* dapat diperoleh nilai a_i yang nantinya digunakan untuk menemukan w. Terdapat nilai a_i untuk setiap data pelatihan. Data pelatihan yang memiliki nilai $a_i > 0$ adalah *support vector* sedangkan sisanya memiliki nilai $a_i = 0$. Dengan demikian fungsi keputusan yang dihasilkan hanya dipengaruhi oleh *support vector.* Salah satu algoritma untuk menyelesaikan permasalahan *Quadratic Programming* ini salah satunya adalah algoritma *Sequential Minimal Optimization* (SMO) [23]. SMO dapat menyelesaikan permasalahan dalam mencari nilai *lagrange multiplier* dengan kondisi KKT berikut:

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y_i u_i \geq 1, \\ 0 < \alpha_i < C &\Leftrightarrow y_i u_i = 1, \\ \alpha_i = C &\Leftrightarrow y_i u_i \leq 1. \end{aligned} \tag{2.19}$$

Pengupdate-an nilai dari *alpha* dibutuhkan nilai dari L dan H untuk batas atas dan bawah. Nilai L dan H bisa didapatkan dengan persamaan berikut jika y1 != y2.

$$L = \max(0, a_2 - a_1) \qquad H = \min(C, C + a_2 - a_1) \tag{2.20}$$

Dan jika y1 = y2, maka persamaan berikut digunakan.

$$L = \max(0, \alpha_2 + \alpha_1 - C), \qquad H = \min(C, \alpha_2 + \alpha_1). \tag{2.21}$$

Sehingga nilai dari *alpha* baru bisa didapatkan dari persamaan berikut.

$$a_2^{new} = a_2 + \frac{y_2(E_1 - E_2)}{n} \tag{2.22}$$

Dimana nilai dari n didapatkan dari persamaan berikut.

(2.23)

$$n = K(x1, x1) + K(x2, x2) - 2K(x1, x2)$$

Setelah nilai *alpha* baru ditemukan, maka langkah selanjutnya adalah mengatur nilai *alpha* agar tidak keluar dari *boundary.* Pembatasan tersebut dilakukan dengan menggunakan persamaan berikut.

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H & \text{if} \quad \alpha_2^{\text{new}} \geq H; \\ \alpha_2^{\text{new}} & \text{if} \quad L < \alpha_2^{\text{new}} < H; \\ \text{L} & \text{if} \quad \alpha_2^{\text{new}} \leq L. \end{cases} \tag{2.24}$$

Sehingga nilai *alpha* yang satunya lagi bisa didapatkan dengan persamaan berikut.

$$a_1^{new} = \mathrm{a}_1 + s(a_2 - a^{new\,clipped}{}_2) \tag{2.25}$$

Langkah tersebut akan dilakukan pada setiap nilai *alpha* sampai iterasi maksimal yang ditentukan atau nilai *alpha* sudah teroptimisasi.

Untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier formula SVM dimodifikasi karena tidak akan ada solusi yang ditemukan. Oleh karena itu, kedua bidang pembatas harus diubah sehingga lebih fleksibel (untuk kondisi tertentu) dengan penambahan variable *slack*, teknik tersebut dinamakan teknik *softmargin*. Dengan demikian formula pencarian bidang pemisah terbaik berubah menjadi:

$$\begin{aligned} &\min \frac{1}{2}|w|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \\ &s.t.\, y_i(w.x_i + b) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \tag{2.26}$$

Parameter C adalah parameter yang menentukan besar penalti akibat kesalahan dalam klasifikasi data dan nilainya ditentukan oleh pengguna. Metode lain untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier adalah dengan mentransformasikan data ke dalam dimensi ruang fitur (*feature space*) sehingga dapat dipisahkan secara linier pada *feature space* (fungsi kernel). Pemetaan kedalam dimensi ruang fitur mengakibatkan komputasi yang sangat besar dikarenakan ada

kemukingkan dibutuhkan untuk ditransformasikan pada ruang yang tidak terhingga, sehingga untuk mengatasi permasalahan ini, digunakan lah *kernel trick*. Jika terdapat sebuah fungsi kernel K, maka fungsi transformasi φ (x_k) tidak perlu diketahui secara persis. Dengan menggunakan fungsi kernel, maka persamaan berubah sebagai berikut. (2.27)

$$L_d = \sum a_i - \frac{1}{2}\sum a_i a_j y_i y_j K(x_i, x_j)$$

Dengan demikian fungsi yang dihasilkan dari pelatihan adalah sebagai berikut. (2.28)

$$f(x_d) = \sum_{i=1}^{ns} \alpha_i y_i K(x_i, x_d) + b \;\; (x_i = support \;\; vector).$$

Dalam non linear SVM, pertama-tama data x dipetakan oleh fungsi *Φ (x)* ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini, *hyperplane* yang memisahkan kedua *class* tersebut dapat dikonstruksikan. Hal ini sejalan dengan teori *Cover* yang menyatakan "Jika suatu transformasi bersifat non linear dan dimensi dari *feature space* cukup tinggi, maka data pada *input space* dapat dipetakan ke *feature space* yang baru, dimana *pattern-pattern* tersebut pada probabilitas tinggi dapat dipisahkan secara linear".

Pemetaan ini dilakukan dengan menjaga topologi data, dalam artian dua data yang berjarak dekat pada *input space* akan berjarak dekat juga pada *feature space*, sebaliknya dua data yang berjarak jauh pada *input space* akan juga berjarak jauh pada *feature space*. Selanjutnya proses pembelajaran pada SVM dalam menemukan titik-titik *support vector*, hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi.

Karena umumnya transformasi *Φ* ini tidak diketahui, dan sangat sulit untuk difahami secara mudah, maka perhitungan *dot product* tersebut sesuai teori Mercer dapat digantikan dengan fungsi kernel yang terlihat pada persamaan berikut:

$$K(x_i, x) = \Phi(x_i).\Phi(x) \tag{2.29}$$

Beberapa kernel yang terdapat pada svm meliputi:

a. Kernel linier

$$K(x_i, x) = x_i^T x \tag{2.30}$$

b. Polynomial

$$K(x_i, x) = (\Upsilon\, x_i^T x + r)^p, \Upsilon > 0 \tag{2.31}$$

c. Radial basis function (RBF)

$$K(x_i, x) = \exp(-\Upsilon|x_i - x|^2), \Upsilon > 0 \tag{2.32}$$

d. Sigmoid kernel

$$K(x_i, x) = tanh(\Upsilon\, x_i^T x + r) \tag{2.33}$$

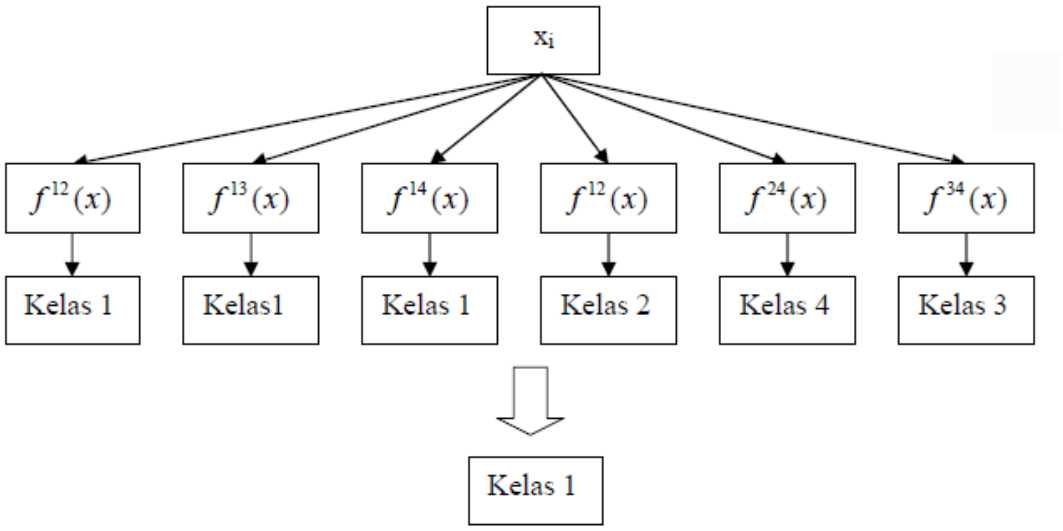
Untuk mengklasifikasikan banyak kelas (*multi class),* Ada dua pilihan untuk mengimplementasikan *multi class* SVM yaitu dengan menggabungkan beberapa SVM biner atau menggabungkan semua data yang terdiri dari beberapa kelas ke dalam sebuah bentuk permasalahan optimasi. Namun, pada pendekatan yang kedua, permasalahan optimasi yang harus diselesaikan jauh lebih rumit. Berikut ini adalah salah satu metode yang umum digunakan untuk mengimplementasikan *multi class* SVM yaitu dengan pendekatan "*one-against-one"* [24].

Dengan menggunakan metode "*one-against-one*", dibangun $n(n-1)/2$ buah model klasifikasi biner (n adalah jumlah kelas). Setiap model klasifikasi dilatih pada data dari dua kelas. Untuk data pelatihan dari kelas ke-I dan kelas ke-j, dilakukan pencarian solusi untuk persoalan optimasi konstrain. Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan model klasifikasi selesai dibangun. Salah satunya adalah metode voting. Contoh penggunaanya dalam mengklasifikasi data baru dapat dilihat sebagai berikut:

Tabel 2.1 Contoh SVM biner dengan metode *One-against-one*

$Y_i = 1$	$Y_i = -1$	Hipotesis

Kelas 1	Kelas 2	$f^{12}(x) = (w^{12})x + b^{12}$
Kelas 1	Kelas 3	$f^{13}(x) = (w^{13})x + b^{13}$
Kelas 1	Kelas 4	$f^{14}(x) = (w^{14})x + b^{14}$
Kelas 2	Kelas 3	$f^{23}(x) = (w^{23})x + b^{23}$
Kelas 2	Kelas 4	$f^{24}(x) = (w^{24})x + b^{24}$
Kelas 3	Kelas 4	$f^{34}(x) = (w^{34})x + b^{34}$



Gambar 2.8 Diagram keputusan *One-Againts-One*

Jika data x dimasukan ke dalam fungsi hasil pelatihan dan hasilnya menyatakan data x adlah kelas I, maka suara untuk kelas I ditambah satu. Kelas data x akan ditentukan dari jumlah suara terbanyak. Jika terdapat dua buah kelas yang jumlah suaranya sama, maka kelas yang indeksnya lebih kecil dinyatakan sebagai kelas dari data. Jadi pada pendekatan ini terdapat $n(n-1)/2$ buah permasalahan *quadratic programming* yang masing-masing memiliki 2k / *n* variabel (k adalah jumlah data pelatihan). Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas. Oleh karena itu, digunakan 6 buah SVM biner seperti contoh diatas.

2.7 *Rule Based System*

Sistem berbasis aturan (*Rule Based System*) adalah suatu program komputer yang memproses informasi yang terdapat di dalam *working memory* dengan sekumpulan aturan yang terdapat di dalam basis pengetahuan menggunakan mesin inferensi untuk menghasilkan informasi baru. Metode rule based dapat digunakan jika suatu dokumen tersebut merupakan dokumen yang terstruktur. Informasi dokumen yang dapat diperoleh dengan cara mencari struktur suatu dokumen salah satunya adalah abstrak skripsi. Isi struktur yang diambil dari skripsi yaitu judul skripsi, nama penulis, nim, isi abstrak, dan kata kunci pada abstrak. Pengertian informasi terstuktur ialah suatu kalimat atau teks yang dapat dibagi ke dalam beberapa kategori seperti topik, fokus, komentar, latar belakang, dan membandingkan informasi lama atau baru [2].

2.8 Abstrak Skripsi

Abstrak adalah representasi dari isi dokumen yang singkat dan tepat. Abstrak merupakan ringkasan bagian-bagian terpenting dari suatu tulisan dan mendiskripsikan isi dan cakupan dari tulisan. Abstrak skripsi dalam laporan penelitian merupakan suatu rangkuman dari keseluruhan suatu penelitian yang sudah dilakukan. Halaman abstrak di UNIKOM terdiri dari beberapa informasi, seperti judul penelitian, nama penulis, NIM penulis, isi abstrak dan kata kunci untuk penelitian yang dilakukan.

2.9 Bahasa Pemrograman Python

Python adalah bahasa pemrograman bersifat umum. Python diciptakan pada tahun 1990 oleh Guido van Rossum. Bahasa level tinggi, stabil, dinamis, orientasi objek dan *cross platform* adalah karakteristik yang membuat bahasa python disukai oleh banyak pengembang. Bahasa pemrograman python berjalan dikebanyakan hardware dan sistem operasi, sehingga kebanyakan komputer bisa menjalankannya. Bahasa pemrograman Python saat ini dikembangkan dan dikelola oleh suatu tim relawan dengan nama *Python Software Foundation* [25].

2.10 *Scikit-learn*

Scikit-learn adalah sebuah pustaka gratis untuk kebutuhan *machine learning* yang diperuntukan untuk bahasa pemrograman Python. Didalam pustaka ini tersedia berbagai jenis algoritma klasifikasi, regresi dan klustering seperti *Support Vector Machine, Random Forest, K-means* dan lainnya [26].

2.11 *Unified Modeling Language*

UML singkatan dari *Unifed Modeling Language* yang berarti bahasa pemodelan standar. UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Ada beberapa 26 diagram yang digunakan proses pembuatan perangkat lunak berorientasi objek diantaranya, use case diagram, activity diagram, class diagram dan sequence diagram [27].

2.11.1 *Use Case Diagram*

Use case diagram merupakan pemodelan untuk tingkah laku (*behavior*) pada sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case* diagram digunakan untuk mengetahui fungsi apa saja yang terdapat pada sistem. Terdapat dua hal utama yang diperlukan dalam pembentukan suatu *use case diagram* yaitu aktor dan *use case*.

1. Aktor merupakan orang, benda maupun sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. Use Case merupakan fungsionalitas atau layanan yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.11.2 Activity Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem, proses bisnis atau menu yang ada pada perangkat lunak. Setiap use case yang telah dibentuk digambarkan aktivitasnya dalam *activity diagram*, mulai dari peran aktor, peran sistem, dan *decision*. *Activity diagram* juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem.
2. Urutan atau pengelompokan tampilan dari sistem / user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan tampilan antarmuka.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

2.11.3 *Class Diagram*

Class diagram menggambarkan interaksi dan relasi antar kelas yang ada di dalam suatu sistem. Kelas memiliki atribut dan metode. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut:

1. *Private*, tidak dapat dipanggil dari luar kelas yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh kelas yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

Class diagram menggambarkan relasi atau hubungan antar kelas dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam *class diagram*:

1. *Association*

Hubungan antar class yang statis. *Class* yang mempunyai relasi asosiasi menggunakan *class* lain sebagai atribut pada dirinya.

2. *Aggregation*

   Relasi yang membuat *class* yang saling terikat satu sama lain namun tidak terlalu berkegantungan.

3. *Composition*

   Relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.

4. *Dependency*

   Hubungan antar *class* dimana *class* yang memiliki relasi *dependency* menggunakan *class* lain sebagai atribut pada method.

5. *Realization*

   Hubungan antar *class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya.