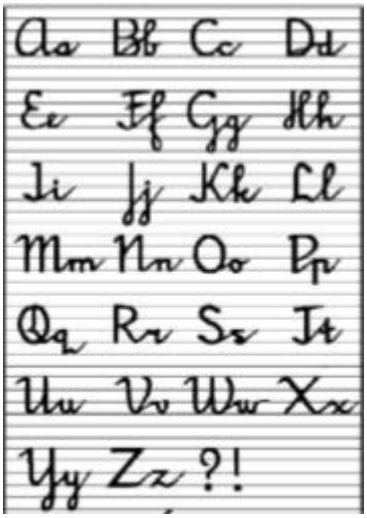
BAB 2
LANDASAN TEORI

2.1 Huruf Latin Bersambung

Dalam hal tulisan huruf latin bersambung, pemerintah telah menetapkan bentuk tulisan tangan yang baku dan resmi pada Keputusan Direktur Jendral Pendidikan Dasar dan Menengah Departemen Pendidikan dan Kebudayaan No. 094/C/Kep/I.83 tanggal 7 Juni 1983. Jadi tulis tangan huruf latin bersambung harus mengacu pada contoh yang ada bawah ini[12].



Gambar 2.1 Tulis Tangan Huruf Latin Bersambung

2.2 Pengolahan Citra Digital

Pengolaha citra digital adalah pemrosesan citra menjadi citra yang lebih baik dalam kualitasnya[13]. Menurut Sutoyo Pengolahan citra digital adalah teknik pengolahan citra untuk memperbaiki kualitas citra agar lebih mudah di kenali oleh manusia atau mesin komputer yang dapat berupa foto atau gambar bergerak[14].

2.2.1 *Resize*

Resize dilakukan untuk menyamakan setiap ukuran citra karena tidak setiap citra memiliki panjang dan lebar yang sama. Metode yang digunakan menggunakan perbandingan ukuran dari setiap hasil citra dengan ukuran citra yang diinginkan. Berikut ini adalah rumus yang digunakan dalam proses *resize*:

Menghitung nilai koordinat baris:

$$x = \frac{pb * pp}{pa} \qquad (2.1)$$

Keterangan:

x = Nilai piksel baris baru

pb = Ukuran panjang matriks baru

pp = Posisi piksel baris

pa = Ukuran panjang matriks lama

Menghitung nilai koordinat kolom:

$$y = \frac{lb * pp}{la} \qquad (2.2)$$

Keterangan:

y = Nilai piksel kolom baru

lb = Ukuran lebar matriks baru

lp = Posisi piksel kolom

la = Ukuran lebar matriks lama

2.2.2 Citra *Grayscale*

Citra *grayscale* adalah citra yang hanya memiliki 1 buah kanal sehingga yang ditampilkan hanya nilai intensitas atau nilai keabuan saja. Dalam citra *grayscale* terdapat kedalam warna 8 bit (256 derajat keabuan). Adapun citra *grayscale* pada Gambar 2.2.

Gambar 2.2 Citra *Grayscale*

Adapun rumus yang biasa digunakan untuk merubah hasil dari citra RGB ke citra *grayscale* dapat dinyatakan pada Persamaan (2.3).

Grayscale = 0.299 * red + 0.587 * green + 0.114 * blue (2.3)

Keterangan :

R = memiliki warna merah pada warna RGB

G = memiliki warna hijau pada warna RGB

B = memiliki warna biru pada warna RGB

2.2.3 Thresholding

Metode yang digunakan untuk *thresholding* disini adalah metode *Otsu Thresholding*. Metode otsu sendiri merupakan salah satu metode yang paling terkenal dan efektif[15]. Tujuan dari metode otsu untuk membagi histogram citra keabuan ke dalam dua daerah yang berbeda dengan otomatis tanpaadanya bantuan dari pengguna untuk memasukkan nilai batas ambang[16].

Dalam mendapatkan nilai threshold ada perhitungan yang harus dilakukan. Langkah pertama buat dalam histogram untuk mengetahui jumlah pixel dari setiap

tingkat keabuan. Dalam tingkat keabuan citra dinyatakan dengan i hingga *L*. Probabilitas pada setiap piksel pada level ke-i dinyatakan pada Persamaan (2.4).

$$P_i = \frac{n_i}{N} \quad (2.4)$$

Keterangan:

P_i = Probabilitas piksel ke-i

n_i = Jumlah piksel dengan tingkat keabuan i

N = Total jumlah piksel pada citra

Langkah selanjutnya mencari nilai dari kumulatif, rerata kumulatif dan intesnsitas global. Dalam mencari nilai tersebut bisa dilihat pada formulasi dalam menghitung jumlah kumulatif dari ω(k), untuk L = 0,1, 2, …, L-1 pada persamaan (2.5), formulasi dalam menghitung jumlah rerata kumulatif dari μ(k), untuk L = 0,1, 2, …, L-1 pada persamaan (2.6), dan formulasi dalam menghitung jumlah rerata intensitas global dari $\mu_T(k)$, untuk L = 0,1, 2, …, L-1 pada persamaan (2.7) .

$$\omega(k) = \sum_{i=0}^{k} P_i \quad (2.5)$$

$$\mu(k) = \sum_{i=0}^{k} i.P_i \quad (2.6)$$

$$\mu_T(k) = \sum_{i=0}^{L-1} i.P_i \quad (2.7)$$

Pada persamaan diatas nilai *k* menyatakan tingkat level keabuan dimana setiap rentang piksel akan dihitung. Menentukan varian atara kelas (*between class variance*), langkah selanjutnya yang dapat dilihat pada persamaan (2.8)

$$\sigma_B^2(k) = \frac{[\mu_T\omega(k)-\mu(k)]^2}{\omega(k)[1-\omega(k)]} \quad (2.8)$$

Dari hasil perhitungan *between class variance* dicari nilai maksimal. Nilai yang paling besar digunakan sebagai *threshold* atau nilai ambang (*k*), dengan persamaan (2.9)

$$\sigma_B^2(k*) = max_{1\le x\le L}\, \sigma_B^2(k) \quad (2.9)$$

Keterangan :

$\omega(k)$ = Jumlah Kumulatif

$\mu(k)$ = Rerata Kumulatif

$\mu_T(k)$ = Rerata Intensitas Global

σ_B^2 = Nilai Ambang

Between class variance bertujuan untuk mencari nilai ambang dari citra *grayscale*, dimana nilai *threshold* akan digunakan sebagai nilai acuan untuk merubah citra *grayscale* ke citra biner. Setiap nilai citra tidak memiliki nilai ambang yang sama[15].

2.2.4 Binerisasi

Binerisasi mengubah bentuk warna citra *grayscale* ke hitam dan putih atau biner. Jika dalam piksel yang ada maka warna akan berubah pada citra yang bernilai 0 dan 255 ke dalam piksel yang bernilai 0 dan 1. Sehingga citra berwarna hitam dan putih. Adapun rumus yang digunakan untuk merubah citra grayscale ke citra hitam dan putih atau citra biner yakni menggunakan persamaan (2.10).

$$g(x,y) = \begin{Bmatrix} 0\ jika\ g(x,y) < T \\ 1\ jika\ g(x,y) \geq T \end{Bmatrix} \quad (2.10)$$

2.2.5 *Thinning*

Operasi *thinning* (penipisan) sama seperti erosi, tetapi tidak menyebabkan objek menjadi menghilang. Operasi *thinning* akan mereduksi menjadi hanya rangka yang mendekati garis sumbu objek dan bertujuan untuk mengurangi bagian yang tidak perlu hingga tebal objek tersebut hanya menjadi satu piksel. Pada penelitian ini digunakan algoritma *thinning Zhang-Suen*. Algorimat ini untuk citra biner di mana piksel background bernilai 0, dan *pixel foreground* (region) bernilai 1. Algoritma ini memeliki beberapa tahapan, di mana setiap tahapan terdiri dari 2 langkah dasar yang diaplikasan terhadap titik yang piksel bernilai 1, dan memiliki paling sedikit 1 piksel

dari 8-tetangganya yang bernilai 0. Bisa dilihat proses pada Gambar 2.3 mengilustrasikan titik objek P1:

P9	P2	P3
P8	P1	P4
P7	P6	P5

***Gambar 2.3* Piksel P1**

Langkah pertama dari sebuah tahapan adalah menandai semua titik objek untuk dihapus, jika titik objek memenuhi syarat berikut:

a. $2 \leq N(P1) \leq 6$
b. $S(P1) = 1$
c. $P2 * P4 * P6 = 0$
d. $P4 * P6 * P8 = 0$

Keterangan:

1. Jumlah dari tetangga titik objek P1, yang pikselnya bernilai 1, yaitu: N(P1) = P2 + P3 + P4 + … + P9
2. S(P1) adalah jumlah perpindahan nilai nilai dari 0 (nol) ke 1 (satu) mulai dari P2, P3, …, P8, P9.
3. P2 * P4 * P6 = 0, memiliki arti P2 atau P4 atau P6 bernilai 0 (nol). Dan pada langkah kedua, kondisi (a) dan (b) sama dengan langkah pertama, sedangkan kondisi (c) dan (d) diubah menjadi: (c') P2 * P4 * P8 = 0 (d') P2 * P6 * P8 = 0

2.3 Segmentasi Karakter

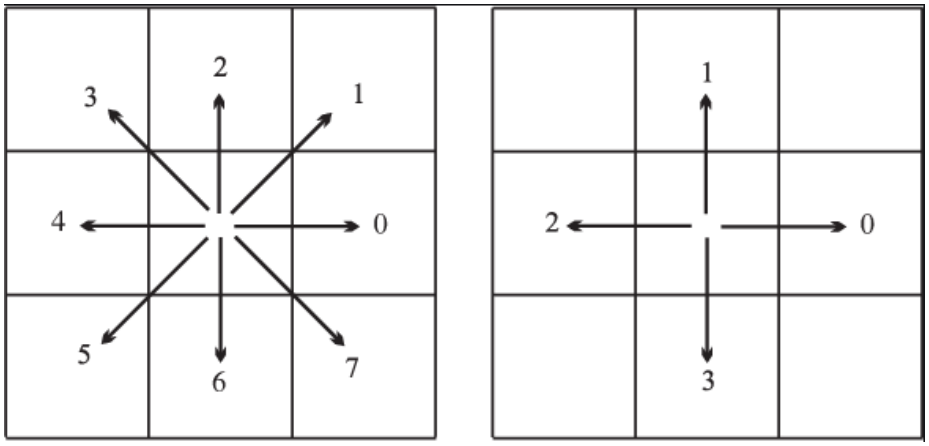
Segmentasi adalah langkah *pre-processing* yang penting dan mempengaruhi ketelitian pada pengenalan tulis tangan. Segmentasi sendiri pada penelitian pengenalan tulis tangan dibagi menjadi dua bagian, yaitu *explicit segmentation* dan *implicit segmentation*. *explicit segmentation* merupakan pemisah citra tulis tangan yang diubah

menjadi huruf. Sedangkan *implicit segmentation* sebagai hasil dari sebuah proses pengenalan tulis tangan.

Pada penelitian ini akan dilakukan proses *explicit segmentation* pada tulis tangan huruf latin bersambung. Mengenai beberapa metode segmentasi yang digunakan, antara lain *hole detection, left and center character*, beserta pengembangnya.

2.4 Ekstarksi Ciri

Freeman chain code banyak digunakan untuk pengolahan citra untuk merepresentasikan garis, kurva atau batas tepi dari suatu citra[17]. *Freeman chain code* merupakan salah satu algoritma yang berhasil mengenali pola pada karakter citra[18]. Dalam merepresentaskian batas dengan urutan garis lurus yang terhubung dengan arah tertentu[19]. Representasi yang biasa digunakan didasarkan pada pola 4 atau 8 konektivitas segmen, seperti pada Gambar 2.4.



Gambar 2.4 Pola Chain Code 8 arah dan 4 arah.

Dalam mengenali pola dai suatu objek didalam citra, membutuhkan adanya ciri-khusus yang berguna untuk membedakan antar pola satu objek dengan objek yang lainnya. Ciri yang bagus adalah ciri yang memiliki pembeda pola yag tinggi, sehingga mampu mengelompokkan pola berdasarkan ciri-ciri yang dapat menghasilkan keakuratan tinggi. Ekstraksi ciri merupakan proses pengambilan suatu ciri dari objek

dari dalam citra untuk membedakan objek yang satu dengan yang lainnya. *Chain code* adalah metode yang melakukan penelusuran piksel-piksel objek dengan panduan arah mata angin.

Teknik *chain code* dapat menemukan struktur pembentuk suatu objek. Hasil akhir dari teknik *chain code* adalah sebuah vector ciri yang berisi informasi urutan kode *chain code* pembentuk objek.

2.5 Support Vector Machine (SVM)

Support Vector Machine atau SVM adalah sebuah algoritma untuk penyelesaian masalah dalam pengklasifikasian, yang dikembangkan oleh Boser, Guyon, Vapnik, dan pertama kali di persentasikan tahun 1992. SVM termasuk dalam *supervised learning*[20]. Dan juga SVM tergolong metode baru yang lebih baik dalam *bioinformatics*, klasifikasi text, dan pengenalan tulis tangan.

Hyperplane pemisah terbaik antara ledia *class* dapat diukur dengan margin *hyperplane* dan mencari titik optimum *hyperplane*[21]. Margin merupakan jarak antara *hyperplane* dan *pattern* terdekat dari masing masing *class.* Dalam usaha mencari lokasi *hyperplane* itu merupakan proses dari pembelajaran pada SVM.

Support Vector Machine dalam liner itu merupakan sebuah permisah untuk fungsi linear. Data latih dinyatakan oleh (xi, yi) dan $xi = \{x1, x2, \ldots, xiq\}$ merupakan atribut (fitur) set untuk data latih kelas ke-i. Untuk menyatakan label kelas. Bisa didefinisikan persamaan suatu *hyperplane* pemisah yang ditulis dengan :

$$wxi + b = 0 \qquad (2.11)$$

Data xi yang terbagi ke dalam dua kelas, yang termasuk kelas -1 (sampel negatif) didefinisikan sebagai vektor yang memenuhi pertidaksamaan (2.12) berikut.

$$wxi + b < 0\ untuk\ yi = 1 \qquad (2.12)$$

Sedangkan yang termasuk kelas +1 (sampel positif) memenuhi pertidaksamaan (2.13) berikut.

$$wxi + b > 0\ untuk\ yi = 1 \qquad (2.13)$$

Dimana:
xi = data input
yi = label yang diberikan
w = nilai dari bidang normal
b = posisi bidang relatif terhadap pusat koordinat

Parameter w dan b adalah parameter yang akan dicari nilainya. Bila label data yi = -1, maka pembatas menjadi persamaan (2.14) berikut:

$$wxi + b \geq -1 \qquad (2.14)$$

Bila label data yi = +1, maka pembatas menjadi persamaan (2.15) berikut.

$$wxi + b \geq -1 \qquad (2.15)$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antar bidang pembatas kedua kelas dan titik terdekatnya, yaitu $\frac{2}{W}$. Hal ini dirumuskan sebagai permasalahan quadratic programming (QP) problem yaitu mencari titik minimal persamaan (2.16) dengan memperhatikan persamaan (2.17) berikut.

$$min\ \tau(w) = \frac{1}{2}\left|\left|w\right|\right|^2 \qquad (2.16)$$

$$yi(wx_i + b) - 1 \geq 0, (i = 1, \dots, n) \qquad (2.17)$$

Permasalahan ini ini dapat dipecahkan dengan berbagi teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan 2.15 ke dalam fungsi *Lagrangian* pada persamaan (2.18), dan menyederhanakannya menjadi persamaan (2.19) berikut.

$$L(w, a, b) = \frac{1}{2}\left|\left|w\right|\right|^2 - \sum_{i=1}^{n} \alpha_i \, y_i \, (w^T x_i + b) + \sum_{i=1}^{n} \alpha_i \qquad (2.18)$$

$$L(w, a, b) = \frac{1}{2}\left|\left|w\right|\right|^2 - \sum_{i=1}^{n} \alpha_i \, y_i \, (w^T x_i + b) + \sum_{i=1}^{n} \alpha_i \qquad (2.19)$$

Dimana *ai* adalah *lagrange multiplier* yang bernilai nol atau positif ($ai \geq 0$). Nilai optimal dari persamaan (2.20) dapat dihitung dengan meminimalkan L terhadap w, b, dan a. Dapat dilihat pada persamaan (2.21) sampai (2.22) berikut ini.

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i \, y_i \, x_i \; = 0 \qquad (2.20)$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i \, y_i \, x_i \; = 0 \qquad (2.21)$$

$$\frac{\partial L}{\partial a} = w - \sum_{i=1}^{n} \alpha_i \, y_i \, (w^T x_i + b) - \sum_{i=1}^{n} \alpha_i \, y_i \; = 0 \qquad (2.22)$$

Maka masalah *Lagrange* untuk klasifikasi dapat dinyatakan pada persamaan (2.23) berikut.

$$min\ L(w,a,b) = \frac{1}{2}\left|\left|w\right|\right|^2 - \sum_{i=1}^{n} \alpha_i\, y_i\, (w^T x_i + b) + \sum_{i=1}^{n} \alpha_i \qquad (2.23)$$

Dengan memperhatikan persamaan (2.24) dan (2.25) berikut ini.

$$w - \sum_{i=1}^{n} \alpha_i\, y_i\, x_i\ = 0 \qquad (2.24)$$

$$\sum_{i=1}^{n} \alpha_i\, y_i\ = 0 \qquad (2.25)$$

Model persamaan (2.25) diatas merupakan model primal *Lagrange*. Sedangkan dengan memaksimalkan L terhadap ai, persamannya menjadi persamaan (2.26) berikut ini.

$$max \sum_{i=1}^{n} \alpha_i\ - \frac{1}{2}\sum_{i=1,j=1}^{n} \alpha_i\, y_i\, y_j^T x_i\, x_j^T \qquad (2.26)$$

Dengan memperhatikan persamaan (2.27) berikut ini

$$\sum_{i=1}^{n} \alpha_i\, y_i\ = 0, \alpha_i\ > 0 (i, j = 1, \dots, n) \qquad (2.27)$$

Untuk mencari nilai xi dan yi dapat dilakukan ketika sudah didapatkan nilai dari pembobotan dari masing-masing metode seleksi fitur dan inisialisasi kelas . Hasil dari pembobotan diubah ke dalam bentuk format data svm (x), sedangkan data kelas menjadi label data svm (y).

Untuk mendapatkan nilai ai, langkah pertama adalah mengubah setiap feature data training menjadi nilai vektor (support vector) =$\begin{bmatrix} x \\ y \end{bmatrix}$. Kemudian nilai vektor dari setiap data training dimasukkan ke persamaan (2.28) *kernel trick* phi (φ) berikut ini.

$$\varphi\begin{bmatrix}x\\y\end{bmatrix}=\begin{cases}\sqrt{x_n^2+y_n^2}>2, maka\begin{bmatrix}\sqrt{x_n^2+y_n^2}-x+|x-y|\\\sqrt{x_n^2+y_n^2}-x+|x-y|\end{bmatrix}\\\sqrt{x_n^2+y_n^2}<2, maka\begin{bmatrix}x\\y\end{bmatrix}\end{cases} \quad (2.28)$$

Nilai x didapatkan dari persamaan (2.29) *kernel linear* untuk x berikut

$$\sum_{i=1}^{n} x_i^T x_j \quad (2.29)$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan dari substitusi nilai x dan nilai y ke persamaan 2.28 diberi nilai bias = 1. Kemudian cari parameter *ai*, dengan terlebih dahulu mencari nilai fungsi setiap data training menggunakan persamaan (2.30), lalu mencari nilai *ai* pada persamaan linear menggunakan persamaan (2.31)dengan memperhatikan *i*,*j*=1,…,*n* berikut

$$\sum_{i=1}^{n} a_i T_i^T T_j \quad (2.30)$$

$$\sum_{i=1}^{n} a_i T_i^T T_j = y_i \quad (2.31)$$

Setelah parameter *ai* didapatkan, kemudian masukkan ke persamaan (2.32)berikut.

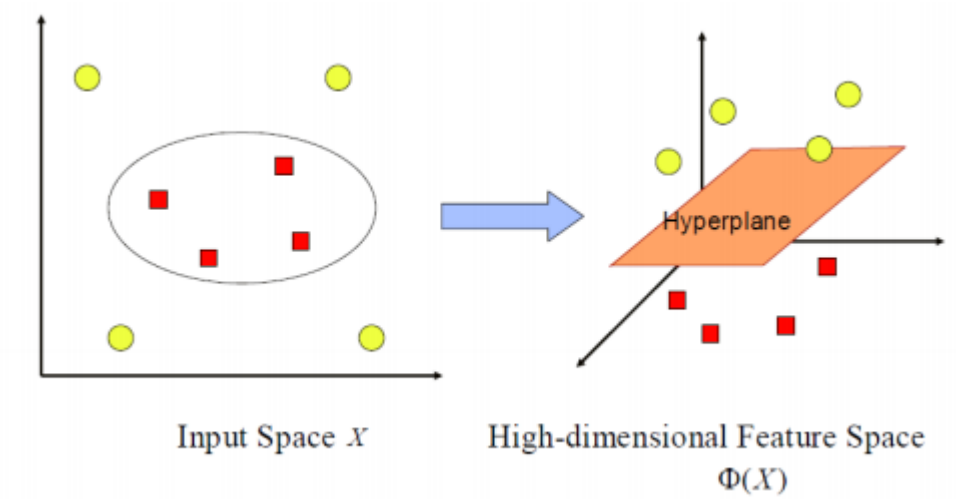
$$w = \sum_{i=1}^{n} a_i T_i \quad (2.32)$$

2.5.1 *Karnel Trick* dan *Non-liniear Clasification* pada SVM

Dalam dunia nyata masalah yang selalu di hadapi jarang sekali bersifat *linier sparable*. Kebanyakan bersifat *non linear*, untuk menyelesaikan masalah *non linear*, SVM menambahkan fungsi *karnel*.

Dalam non linear SVM, pertama-tama data x dipetakan oleh fungsi Φ (x) ke ruang vektor yang berdimensi lebih tinggi. *Hyperlane* yang dapat memisahkan kedua

class tersebut dapat dikonstruksikan. Ilustrasi dari konsep ini dapat dilihat pada gambar 2.5



Gambar 2.5 Fungsi Φ memetakan data ke ruang vektor yang berdimensi lebih tinggi, sehingga kedua class dapat dipisahkan secara linear oleh sebuah hyperplane

Diperlihatkan data pada *class* kuning dan data pada *class* merahyang berbeda input space berdimenrsi dua tidak dapat dipisahkan secara linier. Selanjutnya fungsi Φ memetakan tiap data pada input space tersebut ke ruang vektor baru yang berdimensi lebih tinggi (dimensi 3), dimana kedua *class* dapat dipisahkan secara *linear* oleh sebuah *hyperplane*. Berikut rumus persamaan (2.33).

$$\Phi : \Re^d \rightarrow \Re^q \; d < q \tag{2.33}$$

Pemetaan ini dilakukan dengan menjaga topologi data, dalam artian dua data sebaliknya dua data yang berjarak jauh pada *input space* akan juga berjarak jauh pada *feature space*. Pembelajaran SVM dalam menemukan titik-titik *support vector*, hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi, yakni $\Phi(x_i).\Phi(x_i)$.

Karena umumnya transformasi ini tidak diketahui, dan sangat sulit untuk difahami secara mudah, maka perhitungan *dot product* sesuai teori Mercer dapat digantikan dengan fungsi karnel yang ada di persamaan (2.34)

$$\mathrm{K}(x_i . x_j \Phi(x_i) . \Phi(x_i) \tag{2.34}$$

Beberapa *kernel* yang umumnya dipakai pada SVM adalah :

1. *Polynomial*

Kernel tirck polynomial cocok digunakn untuk menyelesaikan masalah klasifikasi, dimana data pelatihan sudah normal. *Kernel trick* ini dinyatakan dalam persamaan (2.35)

$$K\left(\overrightarrow{x_i}, \overrightarrow{x_j}\right) = (\overrightarrow{x_i} . \overrightarrow{x_j} + 1)^p \tag{2.35}$$

2. *Radial Basis Function* atau *Gaussian*

Kernel trick radial basis function atau gaussian paling banyak digunakan untuk menyelesaikan masalah klasifikasi pada dataset yang tidak terpisah secara *linear*, dimana akurasi pelatihan dan akurasi prediksi yang sangat baik pada *kernel* ini dinyatakan pada persamaan (2.36)

$$K\left(\overrightarrow{x_i}, \overrightarrow{x_j}\right) = exp\left(-\left|\left|\overrightarrow{x_i} - \overrightarrow{x_j}\right|\right|^2\right)\gamma \tag{2.36}$$

3. *Sigmoid*

Kernel sigmoid merupakan *kernel trick* svm hasil dari pengembangan jaringan saraf tiruan, yang dinyatakan pada persamaan (2.37)

$$K\left(\overrightarrow{x_i}, \overrightarrow{x_j}\right) = tanh\left(\overrightarrow{\alpha x_i} . \overrightarrow{x_j} + \beta\right) \tag{2.37}$$

Kernel trick memberikan kemudahan dengan proses pembelajaran SVM, dapat menentikan *support vector* dengan mengetahui fungsi *kernel trick* yang digunakan, tanpa perlu mengetahui fungsi *non linear.*

Dari semua *karnel trick* yang ada *Kernel trick radial basis function* adalah *kernel trick* yang terbaik hasilnya pada proses klasifikasi khususnya pada data yang tidak bisa

dipisahkan secara *linear*. Hasil klasifikasi dari data yang diperoleh dari persamaan (2.38) sebagai berikut :

$$f(x) = \sum_{i=1,\vec{x_i} \in SV}^{n} \alpha_i \; y_i \; K(\vec{x_i}, \vec{x_j}) + b \qquad (2.38)$$

2.5.2 *Multiclass* SVM

SVM memiiki dua pilihan dalam mengimplementasikan multi *class* SVM yaitu dengan menggabungkan SVM biner tau menggabungkan semua data yang terdiri dari beberapa kelas dibuat dalam bentuk permasalahan optimasi. Namun pada pendekatan kedua permasalahn optimasi yang harus diseesaikan malah jadi rumit. Berikut ini metode umum yang digunakan untuk mengimplementasikan *multi class* SVM yang pertama :

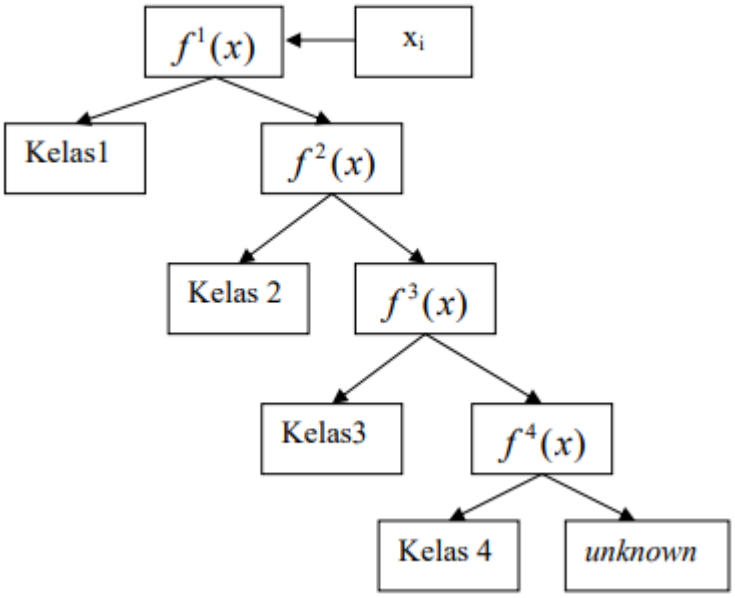
2.5.2.1 Metode One-Against-All

Dengan menggunakan metode ini, dibangun k buah model SVM biner (k adalah jumlah kelas). Mode klasifikasi ke-I dilatih dengan menggunakan seluruh data, untuk mencari sousi permasalahn (2.39).

$$\begin{aligned} &\min_{w^i, b^i, \xi^i} \frac{1}{2}\left(w^i\right)^T w^i + C\sum_t \xi_t^i \\ &s.t \; \left(w^i\right)^T \phi(x_t) + b^i \geq 1 - \xi_t^i \rightarrow y_t = i, \\ &\left(w^i\right)^T \phi(x_t) + b^i \geq -1 + \xi_t^i \rightarrow y_t \neq i, \\ &\xi_t^i \geq 0 \end{aligned} \qquad (2.39)$$

Tabel 2.1 Contoh Klasifikasi 4 buah kelas

$Y_i = 1$	$Y_i = -1$	Hipotesis
Kelas 1	Bukan Kelas 1	$f^1(x) = (w^1)x + b^1$
Kelas 2	Bukan Kelas 2	$f^2(x) = (w^2)x + b^2$
Kelas 3	Bukan Kelas 3	$f^3(x) = (w^3)x + b^3$
Kelas 4	Bukan Kelas 4	$f^4(x) = (w^4)x + b^4$



Gambar 2.6 Contoh klasifikasi dengan metode *One-against-all*

2.5.2.2 Metode One-Against-One

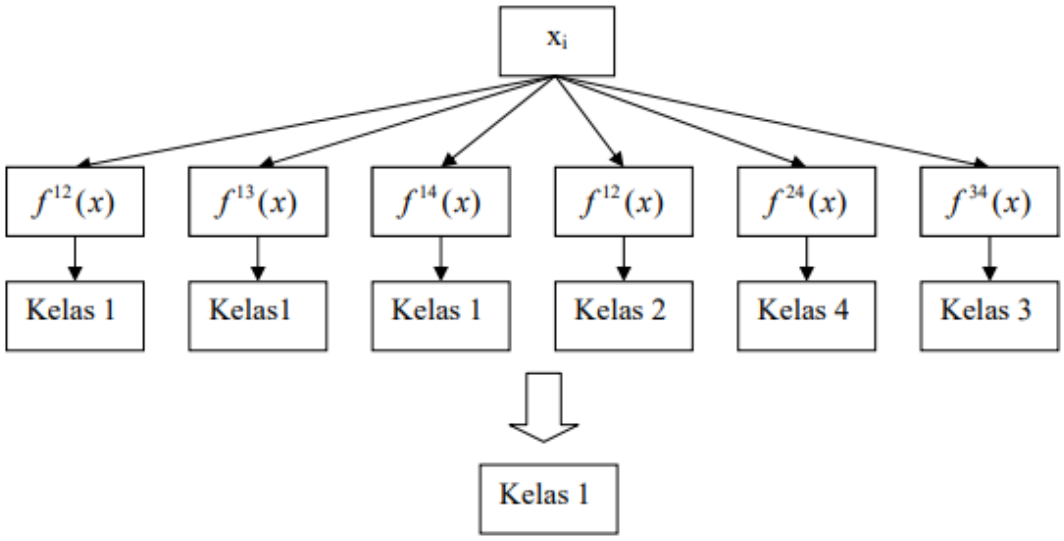
Dengan menggunakan metode ini, $\frac{k(k-1)}{2}$ buah model klasifikasi biner (k adalah jumlah kelas). Setiap model kasifikasi dilatih pada data dari dua kelas. Data pelatihan kelas ke-i dan kelas-j, dilakukan pencarian solusi untuk persoalan optimasi konstrain sebagai berikut :

$$\min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}\left(w^{ij}\right)^T w^{ij} + C\sum_t \xi_t^{ij}$$
$$s.t\ \left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i,$$
$$\left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq -1 + \xi_t^{ij} \rightarrow y_t = j,$$
$$\xi_t^{ij} \geq 0 \qquad (2.40)$$

Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan k(k-1)/2 model klasifikasi selesai dibangun. Salah satunya metode voting [HSU02].

Tabel 2.2 Contoh 6 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 2.7 Contoh klasifikasi dengan metode *one-against-one*

Hasil pelatihan dimana data x dimasukan ke dalam fungsi $(f(x) = (w^{ij})^T \phi(x) + b)$ dan hasil menyatakan x adalah kelas I, maka suara untuk kelas i ditambah satu. Kelas data x akan ditentukan dari jumlah suara yang terbanyak. Jika ada dua buah kelas dengan jumlah suara sama, maka kelas yang indeksnya lebih kecil dinyatakan sebagai kelas dari data. Contoh pada tabel 2.2 adalah permasalahan 6 buah SVM biner dalam memprediksi kelas data baru.

2.6 Pengujian Akurasi

Akurasi merupakan seberapa dekat suatu angka hasi pengukuhan terhadap angka sebenarnya (*true value* atau *reference value*). Tingkat akurasi diperoleh dengan persamaan sebagai berikut

$$Akurasi = \frac{Jumlah\ karakter\ yang\ sama}{jumlah\ total\ karakter} \times 100\%$$

Jumlah akurasi yang sama adalah jumlah karakter yang diprediksi dan hasilnya sama dengan kelas sebenarnya. Jumlah total karakter adalah jumah keseluruhan karakter yang diprediksi kelasnya atau jumlah karakter yang diharapkan.