SISTEM DETEKSI RETINOPATI DIABETIK MENGGUNAKAN SUPPORT VECTOR MACHINE

Tesis
untuk memenuhi sebagian persyaratan
mencapai derajat Sarjana S-2
Program Studi Magister Sistem Informasi

Oleh:

Wahyudi Setiawan
24010410400060

PROGRAM PASCA SARJANA
UNIVERSITAS DIPONEGORO
SEMARANG
2012

ABSTRAK

Retinopati Diabetik merupakan salah satu penyakit komplikasi dari Diabetes Melitus. Penyakit ini dapat menyebabkan kebutaan menetap jika tidak ditangani sedini mungkin. Sistem yang dibangun pada tesis ini adalah deteksi tingkat retinopati diabetik dari citra yang didapatkan dari foto fundus. Terdapat tiga tahap utama untuk menyelesaikan permasalahan yaitu prapengolahan, ekstraksi ciri dan klasifikasi. Metode prapengolahan yang digunakan diantaranya citra kanal hijau, Filter *Gaussian*, *Contrast Limited Adaptive Histogram Equalization* dan *Masking*. Metode *Two Dimensional Linear Discriminant Analysis* (2DLDA) digunakan sebagai ekstraksi ciri. *Support Vector Machine* (SVM) dan *k-Nearest Neighbour* (kNN) digunakan sebagai metode klasifikasi . Hasil pengujian dilakukan dengan mengambil dataset MESSIDOR dengan sejumlah citra yang bervariasi untuk tahap pelatihan, sisanya digunakan untuk tahap pengujian. Hasil pengujian menunjukkan akurasi optimal sebesar 84% untuk metode 2DLDA-SVM dan 80% untuk metode 2DLDA-kNN.

Kata Kunci : Retinopati Diabetik, *Two Dimensional Linear Discriminant Analysis, Support Vector Machine, k-Nearest Neighbour,* MESSIDOR.

ABSTRACT

Diabetic Retinopathy is a complication of Diabetes Melitus. It can be a blindness if untreated settled as early as possible. System created in this thesis is the detection of diabetic retinopathy level of the image obtained from fundus photographs. There are three main steps to resolve the problems, preprocessing, feature extraction and classification. Preprocessing methods that used in this system are Grayscale Green Channel, Gaussian Filter, Contrast Limited Adaptive Histogram Equalization and Masking. Two Dimensional Linear Discriminant Analysis (2DLDA) is used for feature extraction. Support Vector Machine (SVM) and k-Nearest Neighbour (kNN) are used for classification. The test result performed by taking a dataset of MESSIDOR with number of images that vary for the training phase, otherwise is used for the testing phase. Test result show the optimal accuracy are 84% for 2DLDA-SVM and 80% for 2DLDA-kNN.

Keywords : Diabetic Retinopathy, Support Vector Machine, Two Dimensional Linear Discriminant Analysis, *k-Nearest Neighbour,* MESSIDOR.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penelitian dan pengembangan aplikasi dengan berbagai metode dalam pencitraan medis telah berkembang sangat luas. Salah satu penelitian dalam pencitraan medis adalah klasifikasi citra retina untuk deteksi penyakit. Pada citra retina dapat dianalisa untuk mendapatkan informasi penting misalnya informasi tentang tingkat resiko penyakit retinopati diabetik.

Retinopati diabetik merupakan salah satu komplikasi Diabetes Melitus (DM) pada mata yang paling banyak menyebabkan kebutaan menetap, terjadinya seiring dengan lamanya menderita DM. Semakin lama DM diderita semakin tinggi kemungkinan terjadinya retinopati. Retinopati diabetik ditandai dengan adanya gangguan pembuluh darah di retina berupa kebocoran, sumbatan dan pada tahap selanjutnya timbul pembuluh darah tidak normal yang sangat rapuh dan mudah menimbulkan pendarahan dengan segala akibat yang merugikan (Siahaan, 2010).

Retinopati diabetik tidak bisa dideteksi langsung secara kasat mata karena tanda-tandanya berada di bagian syaraf retina. Tanda-tanda penyakit ini hanya dapat dilihat menggunakan foto fundus tetapi memerlukan waktu yang relatif lama untuk mengetahui hasilnya. Permasalahan tersebut diselesaikan dengan membangun sebuah sistem yang dapat mendeteksi tingkat resiko retinopati diabetik dengan waktu yang relatif cepat.

Sistem deteksi yang dibangun memerlukan sebuah model komputasi untuk mengubah piksel citra retina menjadi suatu ciri retina yang terindikasi retinopati diabetik. Tiga permasalahan utama pada sistem, yaitu prapengolahan, ekstraksi ciri dan teknik klasifikasi.

Prapengolahan berfungsi mempersiapkan citra agar dapat menghasilkan ciri yang lebih baik pada tahap berikutnya. Pada tahap ini sinyal informasi ditonjolkan dan sinyal pengganggu (derau) diminimalisasi (Putra, 2010).

Prapengolahan menggunakan metode citra kanal hijau, Filter *Gaussian*, *Contrast Limited Adaptive Histogram Equalization* (CLAHE), dan *Masking*.

Ekstraksi ciri adalah tahapan untuk memunculkan ciri dan mereduksi dimensi citra dari dimensi tinggi ke dimensi lebih rendah. Teknik ekstraksi ciri yang handal merupakan kunci utama dalam penyelesaian masalah pengenalan pola (Purnomo dan Muntasa, 2010). Metode yang sering digunakan untuk ekstraksi ciri diantaranya adalah Analisa Komponen Utama (PCA). Metode PCA bertujuan untuk memproyeksikan data pada arah yang memiliki variasi terbesar, yang ditunjukkan oleh vektor eigen yang bersesuaian dengan nilai eigen terbesar dari matrik kovarian. Disamping itu PCA juga bertujuan untuk mereduksi dimensi dengan melakukan transformasi linier dari suatu ruang berdimensi tinggi ke dalam ruang berdimensi rendah. Kelemahan dari metode PCA adalah kurang optimal dalam pemisahan antar kelas (Yang *et al*, 2004) .

Metode ekstraksi ciri selanjutnya adalah Analisa Diskriminan Linier (LDA). Metode ini mencoba menemukan subruang linear yang memaksimalkan perpisahan dua kelas pola menurut *Fisher Criterion* J_F. Hal ini dapat diperoleh dengan meminimalkan jarak matrik sebaran antar kelas S_w dan memaksimalkan jarak matrik sebaran dalam kelas S_b secara simultan sehingga menghasilkan *Fisher Criterion* J_F yang maksimal. *Diskriminan Fisher Linier* akan menemukan subruang dimana kelas-kelas saling terpisah linier dengan memaksimalkan *Fisher Criterion* J_F. Jika dimensi data jauh lebih tinggi daripada jumlah data pelatihan akan menyebabkan S_w menjadi *singular*. Hal tersebut merupakan kelemahan dari metode LDA (Belhumeur *et al*, 1997).

Metode *Two Dimensional Linear Discriminant Analysis* (2DLDA) menilai secara langsung matrik sebaran antar kelas dari matrik citra tanpa transformasi citra ke vektor. 2DLDA mengatasi *singular problem* dalam matrik sebaran antar kelas (Gao *et al*, 2008). 2DLDA menggunakan *fisher criterion* untuk menemukan proyeksi diskriminatif yang optimal. (Kong *et al*, 2005).

Dalam pengenalan sebuah citra, proses klasifikasi sama pentingnya dengan proses ekstraksi ciri. Setelah ciri-ciri penting data atau citra retina dihasilkan pada proses ekstraksi ciri, ciri-ciri tersebut nantinya akan digunakan

untuk proses klasifikasi. Metode klasifikasi yang digunakan adalah pengklasifikasi *Support Vector Machine* (SVM). Pengklasifikasi SVM menggunakan sebuah fungsi atau *hyperplane* untuk memisahkan dua buah kelas pola. Berbeda dengan jaringan syaraf tiruan yang hanya mencari *hyperplane* saja, SVM berusaha mencari *hyperplane* yang optimal dimana dua kelas pola dapat dipisahkan dengan maksimal (Nugroho dkk, 2003).

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya maka masalah dari penelitian ini adalah bagaimana mengimplementasikan beberapa metode prapengolahan, ekstraksi ciri 2DLDA dan klasifikasi SVM dalam sistem deteksi retinopati diabetik.

1.3 Batasan Masalah

Batasan ruang lingkup permasalahan dari penelitian ini adalah sebagai berikut :

1. Data yang digunakan adalah dataset citra retina MESSIDOR.
2. Dataset terdiri dari 5 kelas, masing-masing kelas terdiri dari 25 citra.
3. Ukuran piksel dari tiap citra yang akan diujicobakan adalah sama yaitu 92x112 piksel.
4. Citra retina berformat BMP.
5. Verifikasi dilakukan dengan membandingkan hasil klasifikasi tahapan pengujian dengan data hasil diagnosis yang telah ada pada dokumen MESSIDOR.

1.4 Keaslian Penelitian

Beberapa kegiatan dan perkembangan mengenai penelitian dengan topik sejenis diantaranya penelitian yang membahas tentang deteksi *hard execudates* retinopati diabetik dengan metode segmentasi *contextual clustering* dan klasifikasi *fuzzy art neural network.* Penelitian ini menggunakan data 25 citra. Dari hasil penelitian didapatkan sensitivitas 93,4% , spesifisitas 80% (Jayakumari dan Santhanam, 2007).

Penelitian selanjutnya membahas tentang deteksi otomatis salah satu penyakit retinopati diabetik yaitu *exudates* dengan menggunakan klasifikasi FCM *clustering* , data terdiri dari 40 citra retina dari 32 pasien yang diambil dari data Thammasat *University Hospital* dengan format JPEG, ukuran citra 500×752 piksel. Dari hasil penelitian didapatkan sensitivitas 92,18% dan spesifisitas 91,52% (Sopharak *et al*, 2009).

Penelitian selanjutnya membahas tentang deteksi retinopati diabetik dengan menggunakan model segmentasi, ekstraksi ciri dan klasifikasi SVM. Data yang digunakan dari basis data MESSIDOR berjumlah 81 citra fundus. Dari hasil penelitian didapatkan sensitivitas 97,5% dan spesifisitas 100% (Zohra dan Mohamed, 2009).

Beberapa penelitian tersebut, belum ada yang menunjukkan tingkat sensitivitas hingga 100%. Perbandingan metode sulit dilakukan karena beberapa penelitian menggunakan lingkungan pengujian yang berbeda, misalnya penggunaan data citra fundus yang tidak sama. Data pelatihan dan pengujian seharusnya menggunakan data yang berbeda sehingga dapat diketahui keakuratan sistem yang dibuat.

Penelitian ini mengintegrasikan beberapa tahap dalam pengenalan pola yaitu pra-pengolahan, ekstraksi ciri menggunakan 2DLDA dan klasifikasi SVM.

1.5 Manfaat Penelitian

Penelitian ini memberikan kontribusi bagi pengembangan sistem diagnosa retinopati diabetik secara otomatis, sehingga dapat membantu dokter spesialis untuk mendiagnosa dan menganalisa penyakit tersebut. Pada ranah pengolahan citra, penelitian ini bermanfaat untuk mengembangkan aplikasi klasifikasi *Support Vector Machine* dalam bidang medis.

1.6 Tujuan Penelitian

Tujuan pada penelitian ini adalah merancang bangun perangkat lunak yang mampu melakukan klasifikasi tingkat resiko retinopati diabetik secara otomatis.

BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Beberapa literatur tentang deteksi retinopati diabetik diantaranya deteksi retinopati diabetik menggunakan metode prapengolahan dengan *utilize local contrast enhancement* dan standarisasi warna. Deteksi abnormalitas menggunakan algoritma *region growing*, *adaptive intensity thresholding* dan operator *edge enhancement*. Metode klasifikasi dilakukan dengan model Jaringan Syaraf Tiruan, dataset yang digunakan berasal dari 1273 pasien untuk tahap pelatihan dan pengujian. Hasil pengujian menunjukkan sensitifitas 94,8% dan spesifisitas 52,8% (Usher *et al*, 2003).

Penelitian selanjutnya adalah deteksi mikoanerisma pada citra fundus menggunakan *top-hat transformation* dan klasifikasi *candidate lesion* berdasarkan pada bentuk *lesion* dan warna. Hasil pengujian didapatkan sensitivitas 90% , spesifisitas 87,5% , namun dari 18 citra yang digunakan tahap pelatihan dan pengujian adalah citra yang sama (Raman *et al*, 2004).

Penelitian selanjutnya adalah mendiagnosa retinopati diabetik dengan tahapan prapengolahan, segmentasi *stage clusters* citra kedalam dua kelas yang berbeda. Metode untuk diagnosa *Red Spots, Bleeding* dan deteksi *Vein-Artery Crossover Points*, menggunakan informasi warna, bentuk, ukuran dan lebar obyek. Data yang diuji terdiri dari 25 citra fundus dan didapatkan sensitivitas 98%, spesifisitas 61% (Iqbal *et al*, 2006).

Penelitian selanjutnya adalah melakukan segmentasi retina untuk identifikasi retinopati diabetik proliferatif. Dataset terdiri dari 27 citra retina. Segmentasi menggunakan *Gabor wavelet transform* dan klasifikasi menggunakan ciri area, *perimeter*, dan lima morfologi ciri berdasar pada *Gaussian wavelet*. Metode *wavelet* mampu melakukan segmentasi pembuluh darah di retina dan mengklasifikasi citra retinopati proliferatif atau tidak (Jelinek *et al*, 2007).

Penelitian selanjutnya adalah mendeteksi *hard exudates* dengan *contextual clustering* dan *fuzzy art neural network*, menggunakan dataset 25 citra. Hasil pengujian menunjukkan tingkat sensitivitas 93,4%, spesifisitas 80% (Jayakumari dan Santhanam, 2007).

Penelitian selanjutnya adalah mendeteksi *exudates* dengan menggunakan klasifikasi *fuzzy C means clustering* , dataset terdiri dari 40 citra retina dari 32 pasien, citra diambil dari Thammasat *University Hospital*, format jpeg . Ukuran citra 500×752 piksel. Dari hasil penelitian didapatkan tingkat sensitivitas 92,18 %, spesifisitas 91,52 % (Sopharak *et al*, 2009).

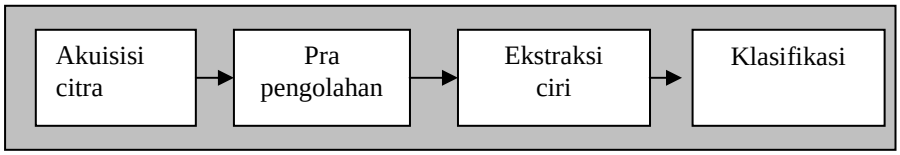
Penelitian selanjutnya adalah melakukan deteksi retinopati diabetik dengan menggunakan model segmentasi, ekstraksi ciri dan klasifikasi SVM. Data yang digunakan dari dataset MESSIDOR berjumlah 81 citra fundus. Dari hasil penelitian didapatkan sensitivitas 97,5% dan spesifisitas 100% (Zohra dan Mohamed, 2009).

Penelitian selanjutnya adalah melakukan segmentasi citra retina digital retinopati diabetik untuk membantu pendeteksian mikroaneurisma. Pada penelitian ini dilakukan kombinasi terhadap metode-metode seperti variasi skala keabuan (skala keabuan biasa, kanal merah, kanal hijau, kanal biru), filter *gaussian*, histogram modifikasi (ekualisasi histogram dan ekualisasi adaptif Histogram), binerisasi (iterasi dan pengambangan ganda), filter *median* dan pelabelan komponen terhubung. Pengujian masing-masing kombinasi dilakukan pada citra retina yang berasal dari dataset DIARETDB1. Dihitung akurasi dengan membandingkan hasil penandaan dokter antara citra asli dan citra hasil segmentasi. Hasilnya kombinasi metode dengan kanal hijau, *filter gaussian*, adaptif ekualisasi histogram 9 x 9, ambang ganda dengan t1=70 dan t2=90, dan filter *median* memberikan akurasi sistem yang paling tinggi yaitu sebesar 94% (Putra dan Suarjana, 2010).

2.2 Landasan Teori

2.2.1 Pengenalan Pola

Pengenalan pola adalah suatu ilmu untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif citra atau sifat dari obyek. Pola sendiri merupakan suatu entitas yang terdefinisi, dapat diidentifikasi dan diberi nama. Pola dapat berupa kumpulan hasil pengukuran atau pemantauan dan dapat dinyatakan dalam notasi vektor atau matrik (Putra, 2009). Struktur sistem pengenalan pola terdapat pada Gambar 2.1, terdiri dari akuisisi citra, pra pengolahan, ekstraksi ciri dan klasifikasi.



Gambar 2.1. Struktur sistem pengenalan pola

2.2.2 Akuisisi citra

Akuisisi citra merupakan cara untuk mendapatkan citra yang akan digunakan dalam proses pengolahan citra. Dalam penelitian ini citra retina yang akan dilatih dan diuji berasal dari foto kamera fundus.

2.2.3 Pra-pengolahan

Pra-pengolahan bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dalam penelitian ini, proses pra pengolahan terdiri dari citra kanal hijau, filter *Gaussian*, *Contrast Limited Adaptive Histogram Equalization* (CLAHE) dan *Masking*.

2.2.3.1 Skala Keabuan

Citra retina yang diterima adalah citra berwarna, sehingga terlebih dahulu perlu dilakukan proses skala keabuan untuk mendapatkan citra dengan aras keabuan. Jumlah warna pada citra keabuan adalah 256, karena citra keabuan jumlah bitnya adalah 8, sehingga jumlah warnanya adalah 2^8=256, nilainya berada

pada jangkauan 0-255. Untuk mendapatkan citra keabuan digunakan persamaan (1).

$$I(x, y) = \alpha.R + \beta.G + \gamma.B \quad (1)$$

dimana I (x,y) adalah level keabuan pada suatu koordinat yang diperoleh dengan mengatur komposisi warna R (merah), G (hijau), B (biru) yang ditunjukkan oleh nilai parameter α, β dan γ. Secara umum nilai α, β dan γ adalah 0.33. Nilai yang lain juga dapat diberikan untuk ketiga parameter tersebut asalkan total keseluruhan nilainya adalah 1 (Putra, 2009). Citra skala keabuan memiliki 4 jenis variasi yaitu skala keabuan biasa, kanal merah, kanal hijau dan kanal biru. Pada penelitian ini digunakan citra kanal hijau. Citra kanal hijau memiliki refleksi cahaya paling baik sehingga dapat dihasilkan informasi yang signifikan tentang pembuluh darah dan struktur retina lain (Kolar dan Harabis, 2009).

2.2.3.2 Filter Gaussian

Filter *Gaussian* adalah salah satu filter linier dengan nilai pembobotan untuk setiap anggotanya dipilih berdasarkan bentuk fungsi *Gaussian*. Filter ini sangat baik untuk menghilangkan derau yang bersifat sebaran normal. Secara alami derau juga memiliki sebaran *Gaussian*, sehingga secara teoritis akan menjadi netral jika dilawan dengan fungsi lain yang juga memiliki fungsi *Gaussian*, hal ini disebut sebagai *zero mean. Zero mean* dari fungsi *Gaussian* dengan nilai pembobotan 2 dimensi ditunjukkan pada persamaan (2) (Ahmad, 2005).

$$\frac{g(x,y)}{k} = e^{\frac{(x^2+y^2)}{2\sigma*\sigma}} \quad (2)$$

dengan k = konstanta normalisasi dan σ menyatakan standar deviasi dari distribusi. Fungsi diatas diasumsikan memiliki *zero mean* (pusat distribusi pada garis x=0). Semakin besar nilai σ maka kurva distribusi *Gaussian* semakin

melebar dan puncaknya menurun. Bentuk 2-D dari fungsi gaussian ditunjukkan pada persamaan (3).

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp\big((x)\big) \quad (3)$$

Tabel 2.1. Tapis distribusi *Gaussian* 2-D dengan ukuran tapis 5 x 5

(x,y)	-2	-1	0	1	2
-2	0,0029	0,0131	0,0215	0,0131	0,0029
1	0,0131	0,0585	0,0965	0,0585	0,0131
0	0,0215	0,0965	0,1592	0,0965	0,0215
1	0,0131	0,0585	0,0965	0,0585	0,0131
2	0,0029	0,0131	0,0215	0,0131	0,0029

2.2.3.3 Ekualisasi Histogram

Histogram digunakan untuk mengetahui informasi frekuensi pemakaian tingkat keabuan dalam suatu citra. Ekualisasi histogram merupakan metode untuk memperbaiki kualitas citra dengan mengubah sebaran tingkat keabuan citra. Hal ini dimaksudkan agar sebaran tingkat keabuan lebih merata dibandingkan dengan citra aslinya (Gonzales dan Woods, 2002). Distribusi ulang terhadap histogram awal dilakukan dengan memetakan setiap nilai piksel pada histogram awal menjadi nilai piksel baru dengan persamaan (4).

$$n(g) = \max\left(0, round\left((L-1) * \frac{c(g)}{N}\right) - 1\right) \quad (4)$$

dengan n(g) adalah nilai piksel baru, N menyatakan banyaknya piksel pada citra (bila citra berukuran 8 x 8, maka N adalah 64), g menyatakan nilai level keabuan awal yang nilainya dari 1 … L-1 (L menyatakan nilai level keabuan maksimum),

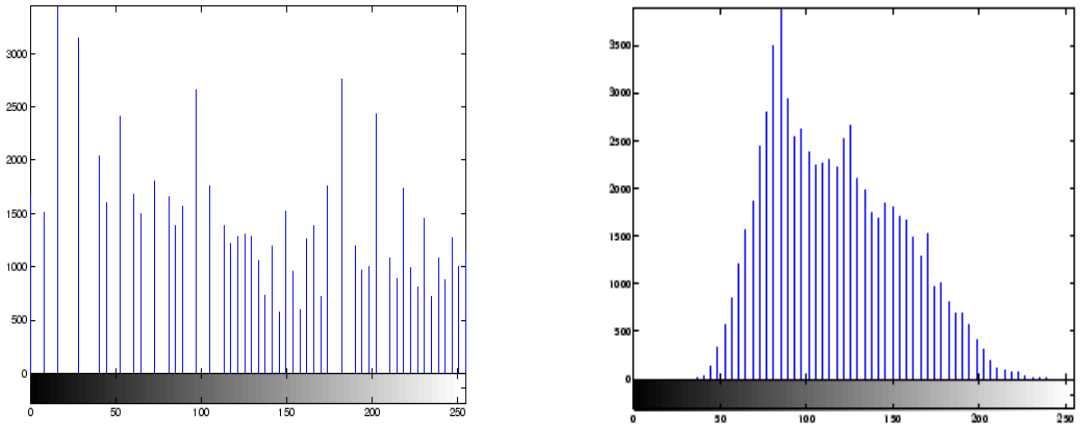
sedangkan c(g) menyatakan banyaknya piksel yang memiliki nilai sama dengan g atau kurang, yang secara matematis dapat dinyatakan pada persamaan (5).

$$c(g) = \sum_{i=0}^{g} h(i), g = 1,2,3,..., L-1 \quad (5)$$

dengan h(i) menyatakan histogram awal.

2.2.3.4 Contrast-Limited Adaptive Histogram Equalization (CLAHE)

CLAHE dapat digunakan sebagai alternatif pengganti ekualisasi histogram. Ekualisasi histogram bekerja pada seluruh citra, sedangkan CLAHE beroperasi pada daerah kecil di citra yang disebut blok. Setiap blok ditingkatkan nilai kontrasnya, sehingga histogram dari wilayah sekitar cocok untuk histogram tertentu. Setelah melakukan pemerataan, CLAHE menggabungkan blok tetangga menggunakan interpolasi bilinier untuk menghilangkan batas-batas artifisial. CLAHE juga dapat digunakan untuk menghindari derau yang ada pada citra dengan membatasi kontras pada daerah homogen. Ilustrasi pada Gambar 2.2 membandingkan antara ekualisasi histogram dan CLAHE dari citra yang sama. CLAHE menghasilkan output citra yang memiliki nilai merata di seluruh bagian citra (Zuiderveld, 1994).



(a) (b)

Gambar 2.2 Histogram hasil Ekualisasi Histogram(a),Histogram hasil CLAHE (b) dari citra pout.tif

2.2.3.5 *Masking*

Citra hasil histogram dilakukan *masking* agar nantinya latar belakang (*background*) berwarna hitam pada citra retina, tidak dihitung sebagai obyek (Putra dan Suarjana, 2010).

2.2.4 Ekstraksi ciri

2.2.4.1 *Analisa Diskriminan Linier (LDA)*

LDA bekerja berdasarkan analisa matrik penyebaran yang bertujuan menemukan suatu proyeksi optimal sehingga dapat memproyeksikan data input pada ruang dengan dimensi yang lebih kecil dimana semua pola dapat dipisahkan semaksimal mungkin. Karenanya untuk tujuan pemisahan tersebut maka LDA akan mencoba untuk memaksimalkan penyebaran data-data input diantara kelas-kelas yang berbeda dan sekaligus juga meminimalkan penyebaran input pada kelas yang sama. Perbedaan antar kelas direpresentasikan oleh matrik S_b dan perbedaan dalam kelas direpresentasikan oleh matrik S_w.

Sekelompok *n* data pelatihan (x_1, x_2, …, x_m) yang memiliki nilai-nilai didalam ruang dimensi N (*N-dimensional space*). *k* adalah jumlah kelas dan n_i adalah jumlah data pelatihan pada kelas ke-*i*, dimana *i = 1, …, k*. Maka matrik S_b dan matrik S_w dibentuk sebagai berikut (Liang, 2008) :

$$S_b = \sum_{i=1}^{k} n_i (m_i - m_0)(m_i - m_0)^T \quad (6)$$

$$S_w = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_i^{(j)} - m_i)(x_i^{(j)} - m_i)^T \quad (7)$$

dimana $x_i^{(j)}$ adalah rata-rata kelas ke-*i*,dan $m = \frac{1}{n} \sum_{i=1}^{k} \sum_{x \in \Pi_i} x$ adalah rata-rata global.

Metode LDA berusaha untuk menemukan proyeksi matrik yang memaksimumkan rasio antara jarak antar kelas dengan jarak dalam kelas dalam ruang proyeksi:

$$J_1(W) = max \frac{trace(W^T S_b W)}{trace(W^T S_w W)} \quad (8)$$

dimana W adalah matrik $N \times q$ yang kolom-kolomnya terdiri dari q vektor-vektor diskriminan.

Persamaan (8) memiliki semantik yang jelas untuk pembilang dan penyebut. $Trace(W^T S_b W)$ mengukur pemisahan antar kelas-kelas dalam ruang proyeksi dan $trace\ (W^T S_w W)$ mengukur kedekatan dari vektor-vektor didalam kelas pada ruang proyeksi.

Sebagai pengganti dari persamaan (8), beberapa peneliti sering mencari vektor-vektor yang paling diskriminan dengan menggunakan patokan LDA yang klasik sebagai berikut :

$$J_2(W) = max\ trace((W^T S_w W)^{-1}(W^T S_b W)) \quad (9)$$

Untuk menemukan diskriminan matrik W dengan mendiagonalkan secara simultan antara matrik S_b dan S_w ($W^T S_w W=I$, $W^T S_b W=\lambda$,dimana I adalah matrik identitas dan λ adalah matrik diagonal yang elemen-elemennya adalah terurut turun). Persamaan (8) diselesaikan dengan menyamaratakan masalah nilai eigen $S_b W_i=\lambda_i S_w W_i$, dimana adalah vektor eigen yang berhubungan dengan i nilai eigen terbesar λ_i dan W_i merupakan i kolom dari matrik W.

Berikut ini algoritma dari LDA (Damayanti dkk, 2010):

1. Input adalah matrik x
2. Menghitung rata-rata dalam kelas (m_i) dan rata-rata keseluruhan kelas (m).
3. Menghitung matrik sebaran antar kelas. Matrik sebaran antar kelas (S_b) adalah jarak matrik antar kelas, sesuai dengan persamaan (6).
4. Menghitung matrik sebaran dalam kelas. Matrik sebaran dalam kelas (S_w) adalah jarak matrik dalam kelas yang sama, sesuai dengan persamaan (7).
5. Mencari vektor eigen (V) dan nilai eigen (λ)

$$S_b V = \lambda S_w V \tag{10}$$

6. Mengurutkan vektor eigen sesuai dengan urutan nilai yang ada pada nilai eigen dari besar ke kecil. Selanjutnya untuk proses proyeksi menggunakan k-1 eigenvector (dimana k adalah jumlah kelas). Vector ini disebut *Fisher Basis Vector*.
7. Memproyeksikan seluruh citra asal (bukan *centered image*) ke *fisher* basis *vector* dengan menghitung *dot product* dari citra asal ke tiap-tiap *fisher basis vector*.

$$\tilde{x}^i = V^T x^i \tag{11}$$

2.2.4.2 Two-Dimensional Linear Discriminant Analysis (2DLDA)

2DLDA adalah pengembangan dari metode LDA. Pada LDA matrik 2D terlebih dahulu ditransformasikan kedalam bentuk citra vektor satu dimensi, sedangkan pada 2DLDA atau disebut teknik proyeksi citra secara langsung, matrik citra 2D tidak perlu ditransformasikan kedalam bentuk citra vektor namun matrik sebaran citranya dapat dibentuk langsung dengan menggunakan matrik citra aslinya.

$\{A_1,\ldots,A_n\}$ adalah n matrik citra, dimana A_i ($i=1,\ldots,k$) adalah r x c matrik. M_i ($i=1,\ldots,k$) adalah rata-rata citra pelatihan dari kelas ke i dan M adalah rata-rata citra dari semua data pelatihan. ℓ_1 x ℓ_2 adalah ruang dimensi (*dimensional space*) $L \otimes R$, dimana $\otimes$ menunjukkan *tensor product*, L menjangkau $\{u_1,\ldots,u_{\ell_1}\}$ dan R menjangkau $\{v_1,..,v_{\ell_2}\}$, sehingga didefinisikan dua matrik $L = [u_1,\ldots,u_{\ell_1}]$ dan $R = [v_1,..,v_{\ell_2}]$ (Liang Z, 2008).

Metode ekstraksi ciri adalah untuk menemukan L dan R sehingga ruang citra asli A_i dirubah ke dalam ruang citra dimensi rendah menjadi $B_i=L^T A_i R$. Ruang dimensi rendah diperoleh dengan transformasi linier L dan R, jarak antar kelas D_b dan jarak dalam kelas D_w didefinisikan sebagai berikut :

$$D_b = \sum_{i=1}^{k} n_i \left\| L^T (M_i - M) R \right\|_F^2 \quad (12)$$

$$D_w = \sum_{i=1}^{k} \sum_{x \in \Pi_i} \left\| L^T (X - M_i) R \right\|_F^2 \quad (13)$$

dimana $\| \quad \|_F$ merupakan *Frobenius norm*.

Meninjau bahwa $\|A\|_F^2$ = Ptrace($A^T A$) = trace(AA^T) untuk matrik *A*. Sedemikian sehingga persamaan (12) dan (13) dapat direpresentasikan lebih lanjut.

$$D_b = trace(\sum_{i=1}^{k} n_i L^T (M_i - M) R R^T (M_i - M)^T L) \quad (14)$$

$$D_w = trace(\sum_{i=1}^{k} \sum_{x \in \Pi_i} L^T (X - M_i) R R^T (X - M_i)^T L). \quad (15)$$

Sama halnya dengan LDA, metode 2DLDA adalah untuk menemukan matrik *L* dan *R*, sedemikian hingga struktur kelas dari ruang orisinil tetap didalam ruang proyeksi, sehingga patokan (*criterion*) dapat didefinisikan sebagai:

$$J_3(L,R) = max \frac{D_b}{D_W}. \quad (16)$$

Hal tersebut jelas bahwa persamaan (16) terdiri dari matrik transformasi *L* dan *R*. Matrik transformasi optimal *L* dan *R* dapat diperoleh dengan memaksimalkan D_b dan meminimumkan D_w. Bagaimanapun, sangat sulit untuk menghitung *L* dan *R* yang optimal secara simultan. Dua fungsi optimasi dapat didefinisikan untuk memperoleh *L* dan *R*. Untuk sebuah *R* yang pasti, *L* dapat diperoleh dengan menyelesaikan fungsi optimasi sebagai berikut :

$$J_4(L) = maxtrace((L^T S_W^R L)^{-1} (L^T S_b^R L)) \quad (17)$$

dimana

$$S_b^R = \sum_{i=1}^{k} n_i (M_i - M) R R^T (M_i - M)^T \quad (18)$$

$$S_W^R = \sum_{i=1}^{k} \sum_{x \in \Pi_i} (X - M_i) R R^T (X - M_i)^T, \quad (19)$$

Dengan catatan bahwa ukuran matrik S_W^R dan S_b^R adalah *r x r* yang lebih kecil daripada ukuran matrik S_w dan S_b pada *LDA* klasik.

Untuk sebuah *L* yang pasti, *R* dapat diperoleh dengan menyelesaikan fungsi optimasi sebagai berikut :

$$J_5(R) = maxtrace((R^T S_W^L R)^{-1}(R^T S_b^L R)), \quad (20)$$

dimana

$$S_b^L = \sum_{i=1}^{k} n_i (M_i - M)^T L L^T (M_i - M) \quad (21)$$

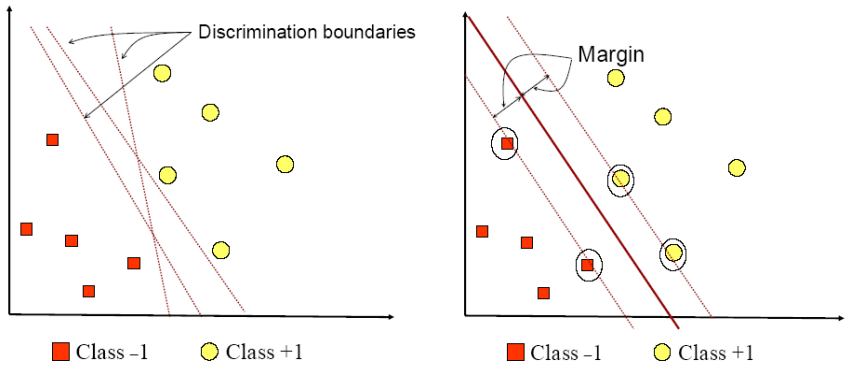
dan

$$S_W^L = \sum_{i=1}^{k} \sum_{x \in \Pi_i} (X - M_i)^T L L^T (X - M_i). \quad (22)$$

2.2.5 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) dikembangkan oleh Boser, Guyon, Vapnik. Pertama kali dipresentasikan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Konsep dasar SVM sebenarnya merupakan kombinasi harmonis dari teori-teori komputasi yang telah ada puluhan tahun sebelumnya, seperti *margin hyperplane* (Duda dan Hart tahun 1973, Cover tahun 1965,Vapnik tahun 1964, dan sebagainya.), kernel diperkenalkan oleh Aronszajn tahun 1950, dan demikian juga dengan konsep-konsep pendukung yang lain.

Berbeda dengan strategi jaringan syaraf tiruan yang berusaha mencari *hyperplane* pemisah antar kelas, SVM berusaha menemukan *hyperplane* yang terbaik pada *input space*. Prinsip dasar SVM adalah pengklasifikasi linier, dan selanjutnya dikembangkan agar dapat bekerja pada permasalahan nonlinier. dengan memasukkan konsep *kernel trick* pada ruang kerja berdimensi tinggi. Perkembangan ini memberikan rangsangan minat penelitian di bidang pengenalan pola untuk investigasi potensi kemampuan SVM secara teoritis maupun dari segi aplikasi. Dewasa ini SVM telah berhasil diaplikasikan dalam problem dunia nyata dan secara umum memberikan solusi yang lebih baik dibandingkan metode konvensional seperti misalnya jaringan syaraf tiruan (Nugroho dkk, 2003).



Gambar 2.3 SVM berusaha menemukan *hyperplane* terbaik yang memisahkan kedua class –1 dan +1

2.2.5.1 Pengenalan pola menggunakan SVM

Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah class pada *input space*. *Hyperplane* dalam ruang vektor berdimensi d adalah *affine subspace* berdimensi d-1 yang membagi ruang vektor tersebut ke dalam dua bagian, yang masing-masing berkorespondensi pada kelas yang berbeda.

Gambar 2.3 memperlihatkan beberapa pola yang merupakan anggota dari dua buah kelas : +1 dan –1. Pola yang tergabung pada class –1 disimbolkan dengan warna merah (kotak), sedangkan pola pada class +1, disimbolkan dengan warna kuning (lingkaran). Problem klasifikasi dapat diterjemahkan dengan usaha

menemukan garis (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah (*discrimination boundaries*) ditunjukkan pada Gambar 2.3 (a).

Hyperplane pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin hyperplane* tersebut. dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* tersebut dengan pola terdekat dari masing-masing kelas. Pola yang paling dekat ini disebut sebagai *support vector*. Garis solid pada Gambar 2.3 (b) menunjukkan *hyperplane* yang terbaik, yaitu yang terletak tepat pada tengah-tengah kedua kelas, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM

Data yang tersedia dinotasikan sebagai $\vec{x}_i \in \Re^d$, sedangkan label masing-masing dinotasikan $y_i = \{+1,-1\}$ untuk *i=1,2,3 …. l.* Yang mana *l* adalah banyaknya data. Diasumsikan kedua class –1 dan +1 dapat terpisah secara sempurna oleh hyperplane berdimensi d , yang didefinisikan

$$\vec{w} \cdot \vec{x} + b = 0 \tag{23}$$

Pattern $\vec{w}$ yang termasuk class –1 (sampel negatif) dapat dirumuskan sebagai pola yang memenuhi pertidaksamaan

$$\vec{w} \cdot \vec{x} + b \leq -1 \tag{24}$$

Sedangkan pattern $\vec{w}$ yang termasuk class +1 (sampel positif)

$$\vec{w} \cdot \vec{x} + b \geq +1 \tag{25}$$

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara *hyperplane* dan titik terdekatnya, yaitu $\frac{1}{\|\vec{w}\|}$. Hal ini dapat dirumuskan

sebagai *Quadratic Programming* (QP) problem, yaitu mencari titik minimal persamaan (26), dengan memperhatikan constraint persamaan (27).

$$\min_{\vec{w}} \quad \tau(w) = \frac{1}{2}\left\|\vec{w}\right\|^2 , \tag{26}$$

$$y_i\left(\vec{x}_i \cdot \vec{w} + b\right) - 1 \geq 0, \quad \forall i \tag{27}$$

Problem ini dapat dipecahkan dengan berbagai teknik komputasi, di antaranya *Lagrange Multiplier*.

$$L\left(\vec{w}, b, \alpha\right) = \frac{1}{2}\left\|\vec{w}\right\|^2 - \sum_{i=1}^{l} \alpha_i \left(y_i\left(\vec{x}_i \cdot \vec{w} + b\right) - 1\right)\right) \tag{28}$$

dengan $i = 1, 2, \ldots, l$.

α_i adalah *Lagrange multipliers*, yang bernilai nol atau positif ($\alpha_i \geq 0$). Nilai optimal dari persamaan (28) dapat dihitung dengan meminimalkan L terhadap $\vec{w}$ dan *b*, dan memaksimalkan L terhadap α_i. Dengan memperhatikan sifat bahwa pada titik optimal gradient L =0, persamaan (28) dapat dimodifikasi sebagai maksimalisasi problem yang hanya mengandung saja α_i, sebagaimana persamaan (29).

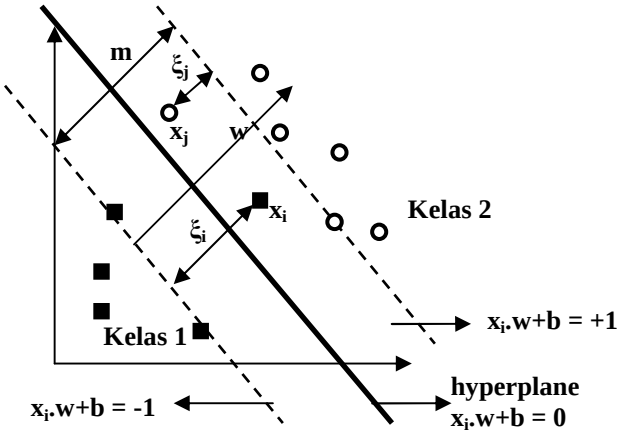
$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,i=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j , \tag{29}$$

dimana $\alpha_i \geq 0 (i = 1,2,...,l) \quad \sum_{i=1}^{l} \alpha_i y_i = 0$. (30)

Dari hasil dari perhitungan ini diperoleh α_i yang kebanyakan bernilai positif. Data yang berkorelasi dengan α_i yang positif inilah yang disebut sebagai *support vector* (Nugroho dkk, 2003).

2.2.5.2 SVM untuk Data Nonlinier

Untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier formula SVM harus dimodifikasi. Oleh karena itu, kedua bidang pembatas pada persamaan (24) harus diubah sehingga lebih fleksibel (untuk kondisi tertentu) dengan penambahan variabel ξ_i ($\xi_i \geq 0$, $\forall_i$; $\xi_i = 0$ jika x_i diklasifikasikan dengan benar) menjadi $x_i .w + b \geq 1 - \xi_i$ untuk kelas 1 dan $x_i .w + b \leq -1 + \xi_i$ untuk kelas 2. Pencarian bidang pemisah terbaik dengan dengan penambahan variabel ξ_i sering juga disebut *soft margin hyperplane* (Burges,1998). Gambar 2.4 menunjukkan Gambar *soft margin hyperplane*, dengan penambahan variabel ξ_i.



Gambar 2.4 *Soft margin hyperplane*.

Dengan demikian formula pencarian bidang pemisah terbaik berubah menjadi:

$$\min \frac{1}{2}|w|^2 + C\left(\sum_{i=1}^{n} \xi_i\right), \tag{31}$$

dimana $y_i(x_i.w+b) \geq 1-\xi_i$ dan $\xi_i \geq 0$. *C* adalah parameter yang menentukan besar penalti akibat kesalahan dalam klasifikasi data dan nilainya ditentukan oleh pengguna.

Selanjutnya, bentuk *primal problem* sebelumnya berubah menjadi:

$$Lp = \frac{1}{2}|w|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{1}^{n}\alpha_i\{y_i(x_i.w+b)-1+\xi_i\} - \sum_{i=1}^{n}\mu_i\xi_i \qquad (32)$$

Dengan cara yang sama dengan penurunan persamaan dual problem pada data linier, maka persamaan *dual problem* untuk data nonlinier adalah sebagai berikut:

$$L_D(w,b,\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i x_j \qquad (33)$$

dimana nilai α_i adalah $0 \geq \alpha_i \geq C$. Hal ini lebih dikenal dengan C-SVM.

Metode lain untuk menyelesaikan permasalahan data nonlinier dalam SVM adalah dengan cara memetakan data ke ruang dimensi lebih tinggi (ruang ciri atau *feature space*) (Burges, 1998), dimana data pada ruang tersebut dapat dipisahkan secara linier, dengan menggunakan transformasi Φ.

$$\Phi : \Re^d \mapsto \mathrm{H}\,. \qquad (34)$$

Dengan demikian algoritma pelatihan tergantung dari data melalui *dot product* dalam H. Sebagai contoh $\Phi(x_i)$. $\Phi(x_j)$. Jika terdapat fungsi kernel K, sedemikian hingga $K(x_i,x_j) = \Phi(x_i).\ \Phi(x_j)$, dengan demikian dalam algoritma pelatihan hanya memerlukan fungsi kernel *K*, tanpa harus mengetahui transformasi Φ secara pasti.

Dengan mentransformasikan $x_k \rightarrow \Phi(x_k)$, maka nilai *w* menjadi $w = \sum_{i=1}^{nsv}\alpha_i y_i \Phi(x_i)$

dan fungsi pembelajaran menjadi:

$$f(x_d) = \sum_{i=1}^{nsv} \alpha_i y_i \Phi(x_i).\Phi(x_d) + b\,. \qquad (35)$$

Feature space biasanya mempunyai dimensi yang lebih tinggi, hal ini mengakibatkan komputasi pada *feature space* mungkin sangat besar. Untuk mengatasi hal ini, maka digunakan "*kernel trick*" atau $K(x_i,x_j) = \Phi(x_i).\ \Phi(x_j)$, maka persamaan (35) menjadi :

$$f(x_d) = \sum_{i=1}^{nsv} \alpha_i y_i K(x_i, x_d) + b\,, \qquad (36)$$

dimana x_i adalah *support vector*.

2.2.5.3 SVM untuk Multiclass

SVM pertama kali dikembangkan oleh Vapniks untuk klasifikasi biner, namun selanjutnya dikembangkan untuk klasifikasi *multiclass* (banyak kelas). Pada dasarnya terdapat dua pendekatan untuk menyelesaikan permasalahan SVM untuk *multiclass*. Pendekatan pertama adalah dengan menggabungkan semua data dalam suatu permasalahan optimasi, pendekatan kedua adalah dengan membangun *multiclass classifier*, yaitu dengan cara menggabungkan beberapa SVM biner. Pendekatan pertama menuntut penyelesaian masalah optimasi yang lebih rumit dan komputasi yang tinggi, sehingga pendekatan ini tidak banyak dikembangkan. Berikut adalah beberapa metode untuk mengimplementasi SVM untuk *multiclass* dengan menggunakan pendekatan kedua.

1. Metode One Against All

Metode ini akan membangun sejumlah k SVM biner, dimana k adalah banyaknya kelas (Hsu, 2002). SVM ke-i dilatih dengan seluruh sampel pada kelas ke-i dengan label kelas positif dan seluruh sampel lainnya dengan label kelas negatif. Jika diberikan l data pelatihan $(x_i,y_i),\ldots,(x_l,y_l)$, dimana $x_i \in \Re^n, i = 1,.....l$ dan $y_i \in (1,...,k)$ adalah kelas dari x_i, maka SVM ke-i akan menyelesaikan permasalahan berikut:

$$\min_{w^i, b^i{}_i, \xi^i} \frac{1}{2}(w^i)^T w^i + C\sum_{j=1}^{l} \xi_j^i \,,$$

$$(w^i)^T \Phi(x_j) + b^i \geq 1 - \xi_j^i, \text{ jika } y_j = i \,,$$

$$(w^i)^T \Phi(x_j) + b^i \leq -1 + \xi_j^i, \text{ jika } y_j \neq i \qquad (37)$$

$$\xi_j^i \geq 0, j = 1,....,l \,.$$

dimana data pelatihan x_i dipetakan ke ruang dimensi yang lebih tinggi dengan menggunakan fungsi Φ dan C sebagai parameter pinalti.

Meminimisasi $\frac{1}{2}(w^i)^T w^i$ berarti memaksimalkan $\frac{2}{|w|^2}$ atau margin antara dua kelompok data . Ketika data tidak tepisah secara linier, maka terdapat pinalti sebesar $C\sum_{j=1}^{l} \xi_j^i$ yang dapat mengurangi jumlah *error* pelatihan. Ide dari SVM adalah menyeimbangkan regulasi $\frac{1}{2}(w^i)^T w^i$ dan *error* pelatihan.

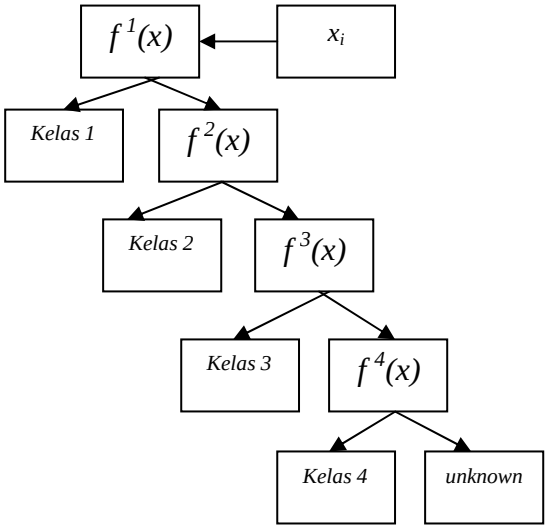
Setelah menyelesaikan permasalahan pada minimisasi, maka terdapat sejumlah *k* fungsi keputusan.

$f^1(x) = (w^1)x + b^1, . . ., f^k(x) = (w^k)x + b^k$

Kelas data *x* akan ditentukan berdasarkan nilai fungsi keputusan yang tertinggi.Untuk pencarian solusi minimisasi diatas menggunakan *quadratic programming.*

Tabel 2.2 Contoh 4 SVM biner dengan metode *One-against-all*

$y_i = 1$	$y_i = -1$	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 2.5 Contoh klasifikasi *One Against All* untuk 4 kelas

2. Metode One Against One

Metode ini akan membangun sejumlah *k(k-1)/2* SVM biner (Hsu, 2002). Dimana setiap SVM biner akan dilatih dengan menggunakan data dari 2 kelas. Untuk data dari kelas ke-*i* dan kelas ke- *j* digunakan permasalahan klasifikasi biner sebagai berikut:

$$\min_{w^{ji},b^{ji},\xi^{ji}} \frac{1}{2}(w^{i,j})^T w^{i,j} + C\sum_t \xi_t^{i,j},$$

$$(w^{i,j})^T \Phi(x_t) + b^{i,j} \geq 1 - \xi_t^{ji}, \text{ jika } y_t = i,$$

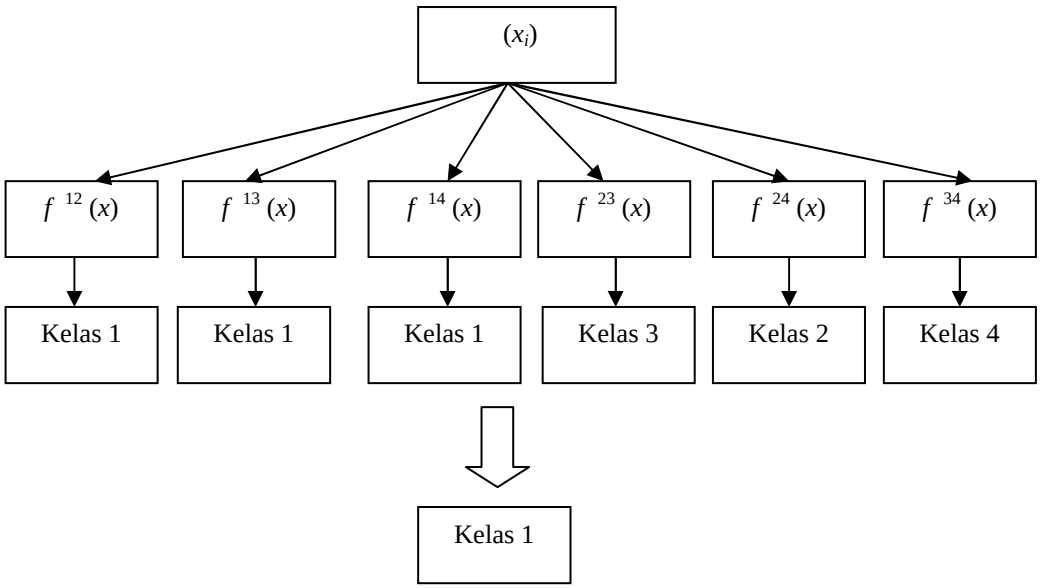
$$(w^{i,j})^T \Phi(x_t) + b^{i,j} \leq -1 + \xi_t^{i,j}, \text{ jika } y_t \neq j, \qquad (38)$$

$$\xi_t^{i,j} \geq 0.$$

Terdapat *k(k-1)/2* fungsi keputusan. Untuk menentukan kelas dari data *x* biasanya dilakukan dengan metode voting. Gambar 2.6 merupakan contoh pelatihan untuk mengklasifikasi 4 kelas dengan menggunakan 6 SVM biner.

Tabel 2.3 Contoh 4 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 2.6 Contoh klasifikasi metode *One Against One* untuk 4 kelas

Dari Gambar 2.6, karena kelas 1 mempunyai suara terbanyak, maka x_i diklasifikasikan sebagai kelas 1. Pada Gambar 2.6 jika data x_i dimasukkan ke dalam fungsi hasil pelatihan pada persamaan berikut :

$$f(x) = (w^{ij})^T \phi(x) + b. \quad (39)$$

Hasilnya menyatakan x_i adalah kelas i, maka suara untuk kelas i ditambah satu. Kelas dari data x_i 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 $k(k-1)/2$ buah permasalahan *quadratic programming* yang masing-masing memiliki $2n / k$ variabel (n adalah jumlah data pelatihan).

2.2.6 k-*Nearest Neighbour* (k-NN)

Metode k-*Nearest Neighbour* (k-NN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan ciri dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran (Rismawan dkk, 2008). Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak Euclidean dengan rumus umum sebagai berikut:

$$d_i = \sqrt{\sum_{i=1}^{p}(x_{2i} - x_{1i})^2} \tag{40}$$

dengan:
x_1 = sampel data
x_2 = data uji
i = variabel data
d = jarak
p = dimensi data

2.2.7 Retinopati Diabetik

Pada penderita diabetes melitus dapat terjadi kelainan retina yang disebut sebagai retinopati diabetik. Retinopati Diabetik akan menyebabkan gangguan ketajaman penglihatan, sehingga penglihatan penderita akan semakin menurun dan dapat menyebabkan kebutaan.

Prosentase terjadinya retinopati diabetik pada penderita diabetes melitus cukup tinggi yaitu berkisar 40%-50%. Pada umumnya retinopati diabetik terjadi pada penderita diabetes melitus yang telah terjangkit selama 10 tahun. Pada usia lanjut sering terlihat retinopati diabetik sebelum penderita menyadari adanya diabetes melitus. Kelainan pada retina yang dapat terjadi akibat retinopati diabetik diantaranya (Kuivalainen dkk, 2005) :

1. Mikroaneurisma merupakan penonjolan dinding kapiler terutama daerah vena dengan bentuk berupa bintik merah kecil yang terletak dekat pembuluh darah.
2. *Hemorrhages* biasanya tampak pada dinding kapiler dan terlihat bercak darah keluar dari pembuluh darah, terlihat berwarna merah gelap, lebih besar dari mikroaneurisma.
3. *Hard exudates* merupakan infiltrasi *lipid* ke dalam retina. Gambarannya khusus yaitu tidak beraturan dan kekuning-kuningan.
4. *Soft exudates* sering disebut *cotton wool patches* merupakan iskemia retina, terlihat bercak berwarna kuning bersifat difus dan berwarna putih.
5. Neovaskularisasi atau pembuluh darah baru biasanya terletak di permukaan jaringan, tampak sebagai pembuluh darah yang berkelok-kelok, dalam, berkelompok dan tidak beraturan.

Tabel 2.4 Tipe penyakit retina abnormal retinopati diabetik

Jenis	Ukuran	Warna	Bentuk	Ket. lain
Mikroaneurisma	sangat kecil	merah gelap	bercak	-
Hemorrhage	kecil hingga besar	merah gelap	titik atau *flame*	-
Hard Exudates	kecil hingga besar	kuning	tidak beraturan	tepi jelas
Soft Exudates	kecil hingga medium	keputih-putihan	biasanya oval	tepi *blur*
Neovaskularisasi	bervariasi	merah	bervariasi	pembuluh darah baru

Hasil diagnosa medis setiap citra dapat menunjukkan tingkat retinopati diabetik:

0 (Normal): ($\mu A = 0$) AND ($H = 0$)

1 : ($0 < \mu A <= 5$) AND ($H = 0$)

2 : (($5 < \mu A < 15$) OR ($0 < H < 5$)) AND ($NV = 0$)

3 : ($\mu A >= 15$) OR ($H >= 5$) OR ($NV = 1$)

μA adalah jumlah mikroaneurisma, H adalah jumlah *hemorrhages*, NV = 1 artinya terdapat neovaskularisasi, NV = 0 artinya tidak terdapat neovaskularisasi.

BAB III
METODE PENELITIAN

3.1 Bahan Penelitian

Bahan penelitian menggunakan dataset MESSIDOR (*Methods to evaluate segmentation and indexing techniques in the field of retinal ophthalmology*). MESSIDOR merupakan sebuah proyek *Techno Vision* yang didanai oleh Kementrian Penelitian dan Pertahanan Perancis pada tahun 2004. MESSIDOR didukung oleh 11 konsorsium yang terdiri dari beberapa universitas, laboratorium dan rumah sakit *opthalmology* di Perancis.

Dataset MESSIDOR yang digunakan pada penelitian ini terbagi menjadi 5 kelas yang terdiri dari 25 citra retina normal, 25 citra retinopati diabetik tingkat 1, 25 citra retinopati diabetik tingkat 2, 25 citra retinopati diabetik tingkat 3 dan 25 citra tidak dikenali (*unrecognized*).

3.2 Alat Penelitian

Dataset citra retina diujikan dengan menggunakan komputer (laptop) dengan dukungan *processor* Intel Pentium Core i3 M370 2,4 GHz, kapasitas *memory* 3 Gigabyte, kapasitas *harddisk* 320 GB. Perangkat lunak pendukung adalah sistem operasi windows 7, Matlab versi 7.12.0.635 (R2011a).

3.3 Jalan Penelitian

3.3.1 Desain Sistem

Sistem klasifikasi retina meliputi tahap pelatihan dan pengujian. Tahap pelatihan dimulai dengan menginputkan citra retina, selanjutnya pada citra akan dilakukan proses pra-pengolahan. Citra diubah terlebih dahulu kedalam format kanal hijau, selanjutnya dilakukan operasi filter *Gaussian* untuk menghilangkan derau. Proses selanjutnya dilakukan ekualisasi histogram untuk mengubah sebaran tingkat keabuan citra, menggunakan *Contrast Limited Adaptive Histogram*

Equalization (CLAHE), selanjutnya citra akan dilakukan operasi *masking* untuk memisahkan obyek dengan latar belakang (*background*).

Ekstraksi ciri pada proses pelatihan dilakukan dengan menggunakan metode 2DLDA. Tahap ini bertujuan untuk mendapatkan ciri-ciri yang terpilih dari masukan data-data pelatihan. Ciri-ciri yang terpilih nantinya digunakan untuk proses klasifikasi pelatihan dan digunakan untuk ekstraksi ciri data pengujian.

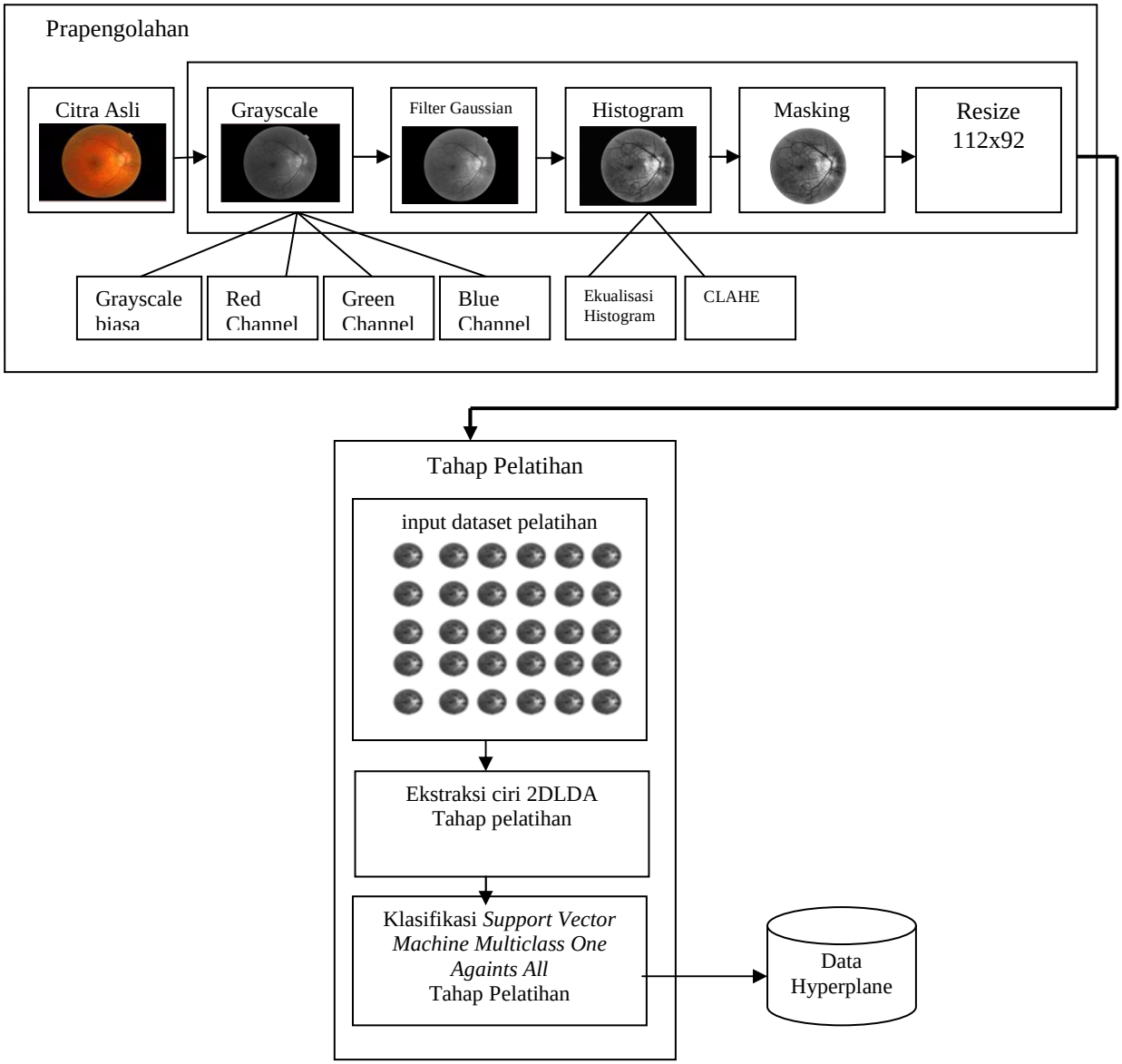
Ekstraksi ciri pada proses pengujian dilakukan dengan mengambil hasil ekstraksi ciri pada proses pelatihan diterapkan pada data pengujian. Hasil ekstraksi ciri pada data pengujian ini nantinya digunakan sebagai inputan pada proses klasifikasi pengujian.

Proses klasifikasi pelatihan dilakukan setelah data-data pelatihan diambil ciri-ciri khusus, ciri-ciri khusus ini berupa vektor ciri yang dimensinya lebih kecil. Dalam penelitian ini menggunakan SVM *multiclass One Against All* dengan kernel *gaussian*. Pada proses klasifikasi pelatihan variabel *hyperplane* untuk setiap pengklasifikasi (*classifier*) yang didapat akan disimpan dan nantinya akan digunakan sebagai data tiap pengklasifikasi dalam proses pengujian, dengan kata lain proses klasifikasi pelatihan adalah untuk mencari *support vector* dari data input menggunakan *quadratic programming*.

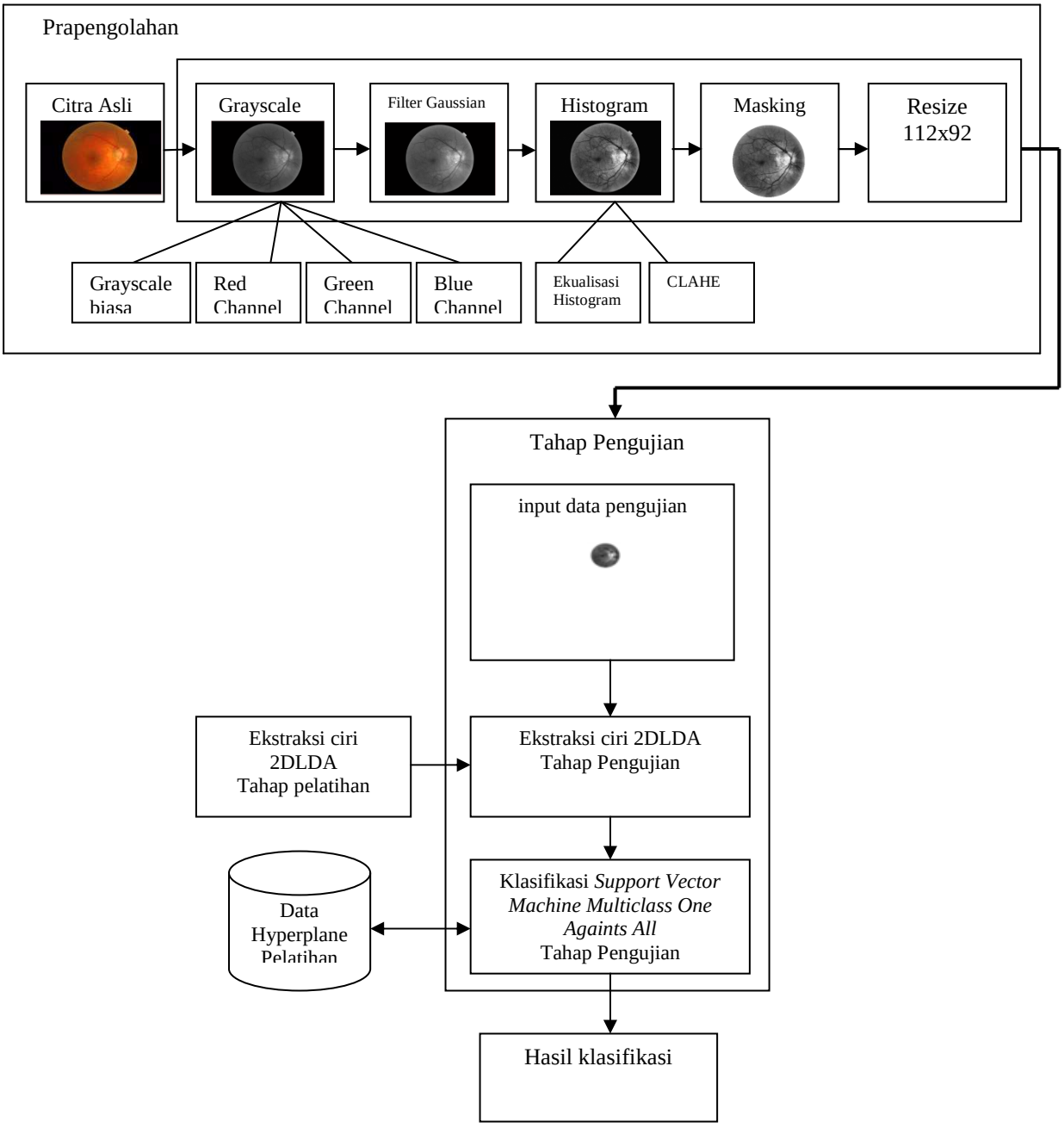
Pada proses klasifikasi pengujian menggunakan hasil ekstraksi ciri data pengujian dan hasil proses klasifikasi pelatihan. Hasil dari proses ini berupa nilai indeks dari fungsi keputusan yang terbesar yang menyatakan kelas dari data pengujian. Jika kelas yang dihasilkan dari proses klasifikasi pengujian sama dengan kelas data pengujian, maka pengenalan dinyatakan benar. Hasil akhirnya berupa citra retina yang sesuai dengan nilai indeks dari fungsi keputusan yang terbesar hasil dari proses klasifikasi pengujian.

Pada Gambar 3.1 dan 3.2 merupakan tahapan proses pelatihan dan proses pengujian sistem deteksi retinopati diabetik. Pada proses pelatihan terdapat metode 2DLDA yang digunakan untuk mengekstraksi ciri, ciri-ciri yang terpilih pada saat proses pelatihan digunakan dalam proses klasifikasi dan juga digunakan untuk mengekstraksi ciri pada data uji coba. Masing-masing dataset citra retina

yang digunakan dibagi menjadi dua, sebagian digunakan untuk proses pelatihan (*training*) dan sisanya digunakan untuk proses pengujian (*testing*).



Gambar 3.1 Tahapan Proses Pelatihan Sistem Deteksi Retinopati Diabetik



Gambar 3.2 Tahapan Proses Pengujian Sistem Deteksi Retinopati Diabetik

3.4 Desain Algoritma

Bagian ini menjelaskan tentang algoritma 2DLDA yang digunakan untuk ekstraksi ciri data pelatihan, data pengujian dan algoritma SVM untuk klasifikasi.

3.4.1 Desain Algoritma 2DLDA

Desain algoritma 2DLDA dibagi menjadi dua subsistem yaitu subsistem pelatihan dan subsistem pengujian. Berikut ini adalah penjabaran masing-masing subsistem.

3.4.1.1 Proses Pelatihan 2DLDA

Untuk proses pelatihan 2DLDA dibagi menjadi tiga tahapan, yaitu : tahap pertama menghitung nilai rata-rata kelas dan rata-rata global, tahap kedua menghitung matrik sebaran dalam kelas dan matrik sebaran dalam kelas, dan tahap terakhir menghitung matrik ciri ekstraksi data-data pelatihan.

1. Rata-rata citra

Berikut ini adalah langkah-langkah dalam proses 2DLDA terhadap suatu dataset citra pelatihan untuk menghitung nilai rata-rata :

1. Jika dalam suatu dataset citra retina terdapat himpunan sebanyak n citra pelatihan $A_i = [A_1,A_2,\ldots,A_n]$ $(i = 1,2,\ldots,n)$ dengan dimensi citra (r x c), maka himpunan total matrik dari semua citra tersebut adalah :

$$A_n = \begin{bmatrix} A_{(n)11} & A_{(n)12} & \cdots & A_{(n)1c} \\ A_{(n)21} & A_{(n)22} & \cdots & A_{(n)2c} \\ \cdots & \cdots & \cdots & \cdots \\ A_{(n)r1} & A_{(n)r2} & \cdots & A_{(n)rc} \end{bmatrix}.$$

   Matrik ini digunakan sebagai data inputan. Data inputan lainnya adalah jumlah kelas (k), jumlah data perkelas (n_i), dan banyaknya data pelatihan (n).
2. Tahapan berikutnya adalah menghitung rata-rata citra pelatihan dari kelas ke i:

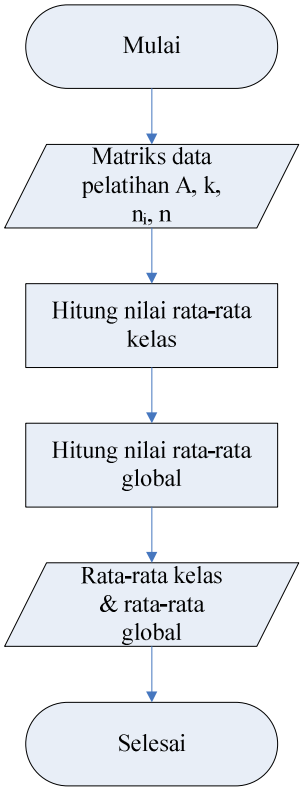
$$M_i = \frac{1}{n_i} \sum_{X \in \Pi_i} X \ .$$

3. Menghitung rata-rata semua citra pelatihan :

$$M = \frac{1}{n} \sum_{i=1}^{k} \sum_{X \in \Pi_i} X \quad .$$

Diagram alir untuk menghitung rata-rata citra dapat dilihat pada Gambar 3.3. Inputannya adalah matrik data pelatihan, pada matrik data pelatihan tidak

ditransformasikan kedalam vektor tetapi tetap berupa matrik. Inputan lainnya adalah jumlah kelas (*k*), jumlah data perkelas (n_i), dan *n* (banyaknya data pelatihan). Outputnya berupa rata-rata global dan rata-rata kelas.



Gambar 3.3. Diagram alir rata-rata citra

2. *Matrik sebaran dalam kelas dan matrik sebaran antar kelas*

Berikut ini adalah langkah-langkah dalam proses 2DLDA terhadap suatu dataset citra pelatihan untuk menghitung matrik sebaran antar kelas dan matrik sebaran dalam kelas:

1. Menentukan nilai ℓ_1 (dimensi proyeksi baris) dan ℓ_2 (dimensi proyeksi kolom). Nilai $\ell_1 \leq r$ dan $\ell_2 \leq c$.
2. Menetapkan matrik transformasi *R* ukuran (c, ℓ_2) yang diperoleh dari gabungan antara matrik identitas ukuran (ℓ_2, ℓ_2) dengan matrik nol ukuran (c-ℓ_2, ℓ_2).
3. Menghitung matrik sebaran antar kelas *R* sesuai dengan persamaan

$S_b^R = \sum_{i=1}^{k} n_i (M_i - M) RR^T (M_i - M)^T$, ukuran matriknya (*r* x *r*).

Ukuran matrik S_b^R lebih kecil dari ukuran matrik S_b pada LDA klasik (Dimensi x Dimensi).

4. Menghitung matrik sebaran dalam kelas *R* sesuai dengan persamaan

   $S_W^R = \sum_{i=1}^{k} \sum_{x \in \Pi_i} (X - M_i) RR^T (X - M_i)^T$, ukuran matriknya (*r* x *r*).

   Ukuran matrik S_W^R lebih kecil dari ukuran matrik S_w pada LDA klasik (Dimensi x Dimensi).

5. Hitung *generalized* nilai eigen (λ_i) dari S_b^R dan S_W^R sesuai dengan persamaan (17).

   $J_4(L)$ = maxtrace$((L^T S_W^R L)^{-1} (L^T S_b^R L))$, ukuran matriknya (*r* x *r*).

6. Ambil sebanyak ℓ_1 vektor eigen terbesar dari langkah 5 sebagai matrik transformasi baris (*L*). $L = [\phi_1^L, ..., \phi_{\ell_1}^L]$, ukuran matriknya (*r* x ℓ_1).

7. Menghitung matrik sebaran antar kelas *L* sesuai dengan persamaan

   $S_b^L = \sum_{i=1}^{k} n_i (M_i - M)^T LL^T (M_i - M)$, ukuran matriknya (*c* x *c*).

   Ukuran matrik S_b^L lebih kecil dari ukuran matrik S_b pada LDA klasik (Dimensi x Dimensi)

8. Menghitung matrik sebaran dalam kelas *L* sesuai dengan persamaan

   $S_W^L = \sum_{i=1}^{k} \sum_{x \in \Pi_i} (X - M_i)^T LL^T (X - M_i)$, ukuran matriknya (*c* x *c*).

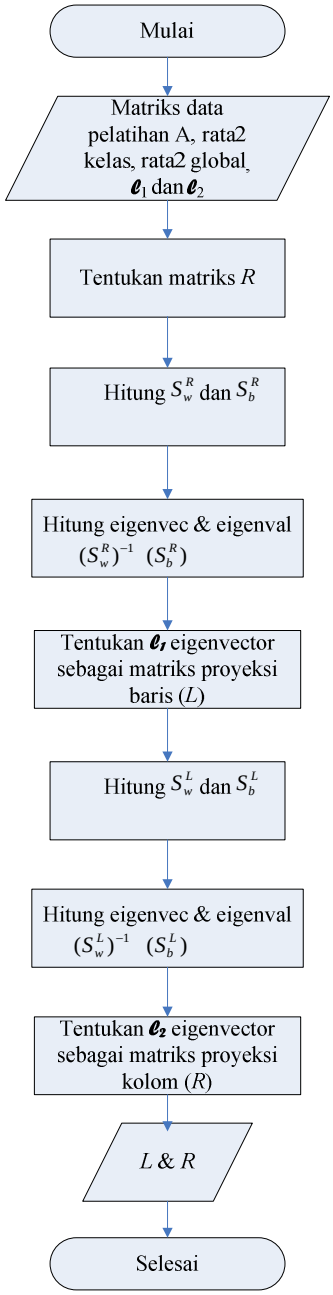
   Ukuran matrik S_W^L lebih kecil dari ukuran matrik S_w pada LDA klasik (Dimensi x Dimensi)

9. Hitung *generalized* nelai eigen (λ_i) dari S_b^L dan S_W^L sesuai dengan persamaan (20).

$J_5(R)=maxtrace((R^T S_W^L R)^{-1}(R^T S_b^L R))$, ukuran matriknya ($c$ x c).

10. Ambil sebanyak ℓ_2 vektor eigen terbesar dari langkah 9 sebagai matrik transformasi kolom (R). $R = [\phi_1^R, ..., \phi_{\ell_2}^R]$, ukuran matriknya ($c$ x ℓ_2).

Diagram alir untuk menghitung matrik sebaran dalam kelas dan matrik sebaran antar kelas dapat dilihat pada Gambar 3.4. Inputannya adalah matrik data pelatihan *A*, jumlah kelas (*k*), jumlah data perkelas (n_i), dan *n* (banyaknya data pelatihan), rata-rata kelas, rata-rata global, ℓ_1(dimensi proyeksi baris), dan ℓ_2 (dimensi proyeksi kolom). Proses ini digunakan untuk mendapatkan matrik transformasi *L* dan matrik transformasi *R* sehingga ruang citra asli (*original image space*) dirubah kedalam ruang citra dimensi rendah (*low-dimensional image*). Matrik transformasi L dapat diperoleh dari pengambilan sebanyak ℓ_1 vektor eigen terbesar dari proses *generalized* nilai eigen (λ_i) dari S_b^R dan S_W^R sesuai dengan persamaan $J(L) = maxtrace((L^T S_W^R L)^{-1}(L^T S_b^R L)$. Sedangkan matrik transformasi R dapat diperoleh dari pengambilan sebanyak ℓ_2 vektor eigen terbesar dari proses *generalized* nilai eigen (λ_i) dari S_b^L dan S_W^L sesuai dengan persamaan $J(R) = maxtrace((R^T S_W^L R)^{-1} (R^T S_b^L R))$. Outputnya berupa *L* (matrik transformasi baris) dan *R* (matrik transformasi kolom).



Gambar 3.4 Diagram alir sebaran antar kelas dan sebaran dalam kelas

3. Ekstraksi Ciri Pelatihan

Gambar 3.5 adalah diagram alir ekstraksi ciri pelatihan yang merupakan langkah terakhir dari proses pelatihan citra yang digunakan untuk pencarian ekstraksi ciri pada setiap citra (*feature image*) dalam dataset citra pelatihan. Langkah-langkahnya sebagai berikut :

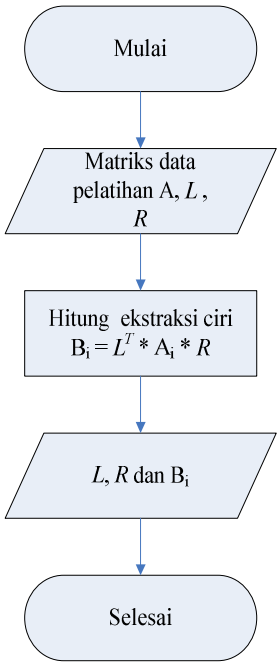
1. Inputan berupa matrik data pelatihan :

$$A_n = \begin{bmatrix} A_{(n)11} & A_{(n)12} & \cdots & A_{(n)1c} \\ A_{(n)21} & A_{(n)22} & \cdots & A_{(n)2c} \\ \cdots & \cdots & \cdots & \cdots \\ A_{(n)r1} & A_{(n)r2} & \cdots & A_{(n)rc} \end{bmatrix}.$$

   Inputan lainnya : matrik transformasi baris (L) dan matrik transformasi kolom (R).

2. Hitung matrik ekstraksi ciri adalah $B_i=L^T A_i R$, ukuran matriknya (ℓ_1 x ℓ_2).

3. Output : matrik ektraksi ciri B_i, matrik transformasi baris L, dan matrik transformasi kolom R.



Gambar 3.5 Diagram alir ekstraksi ciri pelatihan.

3.4.1.2 Proses Pengujian 2DLDA

Proses pengujian 2DLDA hanya terdiri satu proses yaitu proses ekstraksi ciri data pengujian. Gambar 3.6 adalah diagram alir ekstraksi ciri pengujian yang

bertujuan untuk pencarian ciri ekstraski pada citra pengujian. Langkah-langkahnya sebagai berikut :

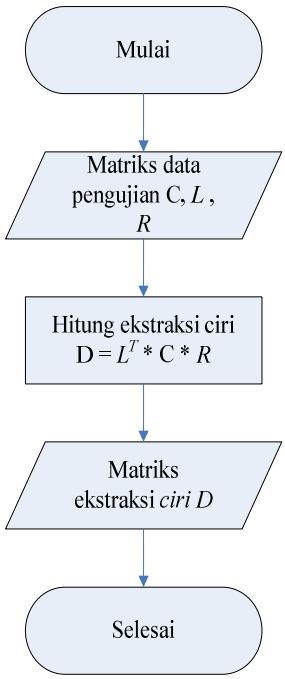
1. Inputan berupa matrik data pengujian *C* yang ukuran dimensi matriknya sama dengan matrik data pelatihan yaitu (*r x c*) :

$$C = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1c} \\ C_{21} & C_{22} & \dots & C_{2c} \\ \dots & \dots & \dots & \dots \\ C_{r1} & C_{r2} & \dots & C_{rc} \end{bmatrix}.$$

   Inputan lainnya : matrik transformasi baris (*L*) dan matrik transformasi kolom (*R*), yang kedua – duanya didapat dari proses pelatihan 2DLDA.

2. Hitung matrik ekstraksi ciri adalah $D=L^T CR$, ukuran matriknya (ℓ_1 x ℓ_2).

3. Output : matrik ektraksi ciri *D*.



Gambar 3.6 Diagram alir ekstraksi ciri pengujian

3.5 Desain Algoritma SVM

Pengklasifikasi SVM untuk *multiclass One Against All* akan membangun sejumlah *k* SVM biner (*k* adalah jumlah kelas). Fungsi keputusan yang mempunyai nilai maksimal, menunjukkan bahwa data x_d merupakan anggota dari kelas fungsi keputusan tersebut.

Pengklasifikasian dengan SVM dibagi menjadi dua proses, yaitu proses pelatihan dan pengujian. Pada proses pelatihan SVM menggunakan matrik ciri yang dihasilkan pada proses ekstraksi ciri pelatihan sebagai input. Sedangkan pada pengujian SVM memanfaatkan matrik ciri yang dihasilkan pada proses ekstraksi ciri pengujian sebagai input.

3.5.1 Proses Pelatihan SVM

Algoritma pelatihan untuk masing-masing pengklasifikasi SVM biner dapat dituliskan sebagai berikut : input berupa matrik *B* (matrik hasil ektraksi ciri pelatihan) dan vektor *Y* sebagai pasangan input-target dan outputnya adalah *w*, *x*, *b* (variabel-variabel persamaan *hyperplane*). Langkah–langkahnya dijelaskan sebagai berikut :

1. Tentukan Input (*Z* = *B*) dan Target (*Y*) sebagai pasangan pelatihan dari dua kelas.
2. Hitung Kernel Gaussian $K(Z,Z_i) = exp\ (\frac{-|Z - Z_i|^2}{(2\sigma^2)})$.
3. Hitung Matrik Hessian $H = K(Z,Z_i) * Y * Y^T$.
4. Tetapkan *c* dan *epsilon*.
5. Tetapkan vektor *e* sebagai vektor satuan yang memiliki dimensi sama dengan dimensi *Y*.
6. Hitung solusi *quadratic programming*:

   $$\min\ L(\alpha) = \frac{1}{2}\alpha^T H\alpha - e^T\alpha,$$

   dimana $y^T\alpha = 0$ dan $0 \le \alpha \le c$.

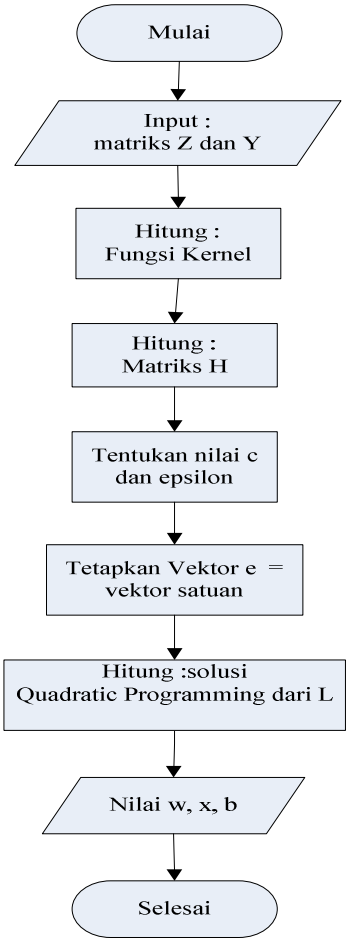
Input matrik *Z* merupakan matrik ciri yang dihasilkan pada proses ekstraksi ciri dan vektor *Y* sebagai target. Contoh untuk dataset MESSIDOR yang terdiri dari 5 kelas, maka jika digunakan sepuluh sampel tiap kelas dan dimensi proyeksi baris ℓ_1 = 10, dimensi proyeksi kolom ℓ_2 = 10, maka matrik *Z* yang dihasilkan adalah matrik ciri dengan dimensi 50 x 100 (50 didapat dari jumlah sampel tiap kelas dikalikan dengan banyaknya kelas, 100 didapat dari perkalian dimensi proyeksi baris dengan dimensi proyeksi kolom). Vektor *Y* merupakan vektor kolom untuk pengklasifikasi pertama dimana semua citra retina dari kelas pertama akan disimbolkan dengan angka 1, semua citra retina dari kelas lainnya dengan angka -1. Vektor *Y* untuk pengklasifikasi kedua semua citra retina dari kelas kedua disimbolkan dengan 1 dan semua citra retina bukan kelas kedua disimbolkan dengan -1, demikian seterusnya untuk pengklasifikasi ketiga sampai ke *k*. Pada penelitian ini, digunakan fungsi *kernel gaussian* dengan nilai varian (σ) = 1.

Langkah selanjutnya adalah menghitung matrik Hessian, yaitu perkalian antara *kernel gaussian* dengan *Y*. *Y* disini adalah berupa *vector* yang berisi nilai 1 dan -1. Dari contoh di atas jika *classifier* pertama yang dilatih, maka nilai *Y* untuk 10 elemen pertama (jika digunakan 10 sampel citra retina per kelas) akan bernilai 1 dan elemen lainnya bernilai -1. Jika *classifier* kedua dilatih, maka 10 elemen berikutnya bernilai 1, sedangkan sisanya bernilai -1. Matrik Hessian ini nantinya digunakan sebagai variabel input dalam *quadratic programming*.

Fungsi *quadratic programming* monqp memerlukan variabel *c* dan *epsilon*. Untuk itu tetapkan nilai *c* dan *epsilon* (c adalah batas atas nilai α_i) dari C-SVM. Vektor satuan e juga dibentuk dengan dimensi sama dengan vektor *Y*.

Penyelesaian min $L(\alpha) = \frac{1}{2}\alpha^T H\alpha - e^T\alpha$ dengan *quadratic programming*, merupakan implementasi dari pencarian solusi atas permasalahan $\min\frac{1}{2}|w|^2 + C\left(\sum_{i=1}^{n}\xi_i\right)$. Jika diimplementasikan dalam bentuk matrik menjadi $\min\frac{1}{2}w^T w + C\left(\sum_{i=1}^{n}\xi_i\right)$, dengan $y_i{}^T(w^T\Phi(x_i)) + b) \geq 1 - \xi_i$. Jika formula dalam bentuk

matrik tersebut diubah ke bentuk dual problem, maka formula tersebut menjadi min $L(\alpha) = \frac{1}{2}\alpha^T H\alpha - e^T\alpha$, dimana $y^T\alpha = 0$ dan $0 \le \alpha \le C$. Disini $H = y_i y_j K(x_i, x_j)$, dan $K(x_i,x_j) = exp(\frac{-|x_i - x_j|^2}{(2\sigma^2)})$ adalah fungsi kernelnya, dan *e* adalah vektor satuan yang dimensi sama dengan *Y* , sedangkan *c* > 0 adalah batas atas dari nilai α Dalam penelitian ini digunakan nilai *c* = 1000 dan *epsilon* = $1x10^{-7}$. Hasil dari fungsi monqp (*quadratic programming*) adalah nilai variabel *w*, *x*, dan *b* yang nantinya akan digunakan untuk proses pengujian. Diagram alir untuk algoritma pelatihan SVM dapat dilihat pada Gambar 3.7.



Gambar 3.7 Diagram alir pelatihan SVM

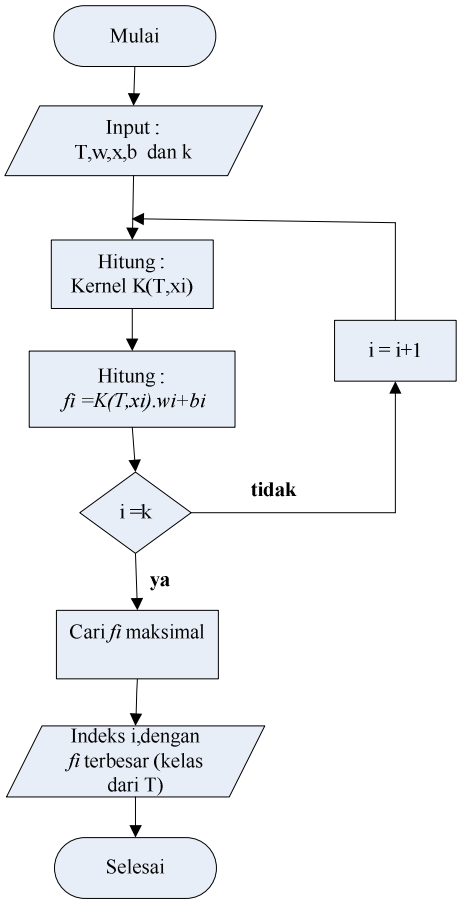
3.5.2 Proses Pengujian SVM

Setelah pada proses pelatihan didapat nilai variabel *w*, *x*, dan *b* untuk masing - masing kelas. Nilai variabel *w, x,* dan *b* untuk didefinisikan sebagai vektor *w, x,* dan *b*. Untuk input data yang akan diklasifikasikan adalah matrik ciri *D* yang dihasilkan pada proses ekstraksi ciri pengujian. Matrik ciri *D* tersebut ditransformasikan dulu kedalam bentuk vektor menjadi 1 x (ℓ_1 x ℓ_2) diberi nama *T*. Langkah – langkahnya sebagai berikut :

1. Input : vektor *T* (data pengujian), vektor *w*, *x*, *b*, dan *k* (jumlah kelas).
2. Hitung Kernel Gaussian $K(T,x_i) = exp\ (\frac{-|T-x_i|^2}{(2\sigma^2)})$.
3. Hitung $f_i = K(T, x_i)w_i + b_i$.
4. Ulangi langkah 2 dan 3 untuk i = 2 sampai k.
5. Tentukan nilai f_i yang paling maksimal.
6. Kelas *i* dengan f_i terbesar adalah kelas dari vektor *T*.

Nilai *T* adalah transformasi matrik ciri *D* kedalam bentuk vektor. Langkah selanjutnya adalah dengan menghitung kernel *Gaussian* $K(T, x_i)$, dengan *T* adalah data input dan x_i adalah *support vector* yang dihasilkan pada proses pelatihan SVM.

Fungsi keputusan $f_i = K(T, x_i)w_i + b_i$ dihitung untuk masing-masing nilai *i*. dimana *i* = 1 sampai *k* (*k* adalah jumlah kelas). Output dari algoritma ini berupa indeks *i* dengan f_i terbesar yang merupakan kelas dari vektor *T*. Diagram alir untuk algoritma pengujian dapat dilihat pada Gambar 3.8.

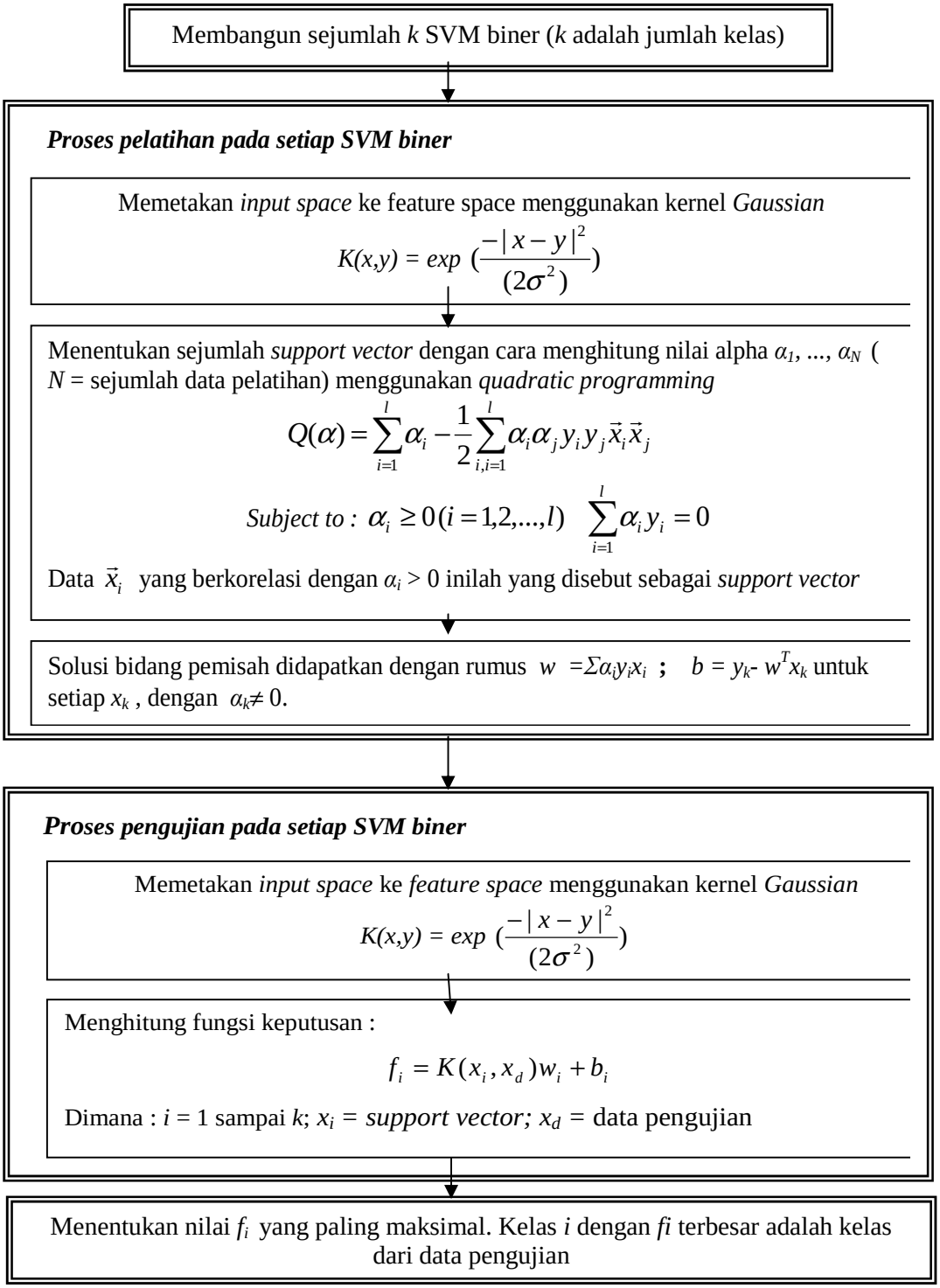


Gambar 3.8 Diagram alir pengujian SVM.

Dari proses pelatihan dan pengujian SVM dapat diringkas dalam bentuk blok diagram proses pelatihan dan pengujian yang ditunjukkan pada Gambar 3.9. Gambar 3.9 menjelaskan secara garis besar proses pelatihan dan pengujian pada SVM.

Data pelatihan yang sudah diproyeksikan oleh 2DLDA, selanjutnya menjadi data pelatihan SVM. Jika sebaran data yang dihasilkan pada proses 2DLDA mempunyai distribusi yang tidak linier, maka salah satu metode yang digunakan SVM untuk mengklasifikasikan data tersebut adalah dengan mentransformasikan data ke dalam dimensi ruang ciri (*feature space*), sehingga dapat dipisahkan secara linier pada ruang ciri. Karena ruang ciri dalam prakteknya biasanya memiliki dimensi yang lebih tinggi dari vektor input (*input space*). Hal

ini mengakibatkan komputasi pada ruang ciri mungkin sangat besar, karena ada kemungkinan ruang ciri dapat memiliki jumlah ciri yang tidak terhingga.



Gambar 3.9. Blok diagram proses pelatihan dan klasifikasi menggunakan SVM. (Damayanti dkk, 2010)

Maka pada SVM digunakan "*kernel trick*". Fungsi kernel yang digunakan pada penelitian ini adalah *Gaussian*

$$K(x,y) = exp\left(\frac{-|x-y|^2}{(2\sigma^2)}\right). \quad (41)$$

Sejumlah *support vector* pada setiap data pelatihan harus dicari untuk mendapatkan solusi bidang pemisah terbaik. Persoalan solusi bidang pemisah terbaik dapat dirumuskan :

$$Q(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,i=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j \,, \quad (42)$$

dimana : $\alpha_i \geq 0 (i = 1,2,...,l)$ $\sum_{i=1}^{l} \alpha_i y_i = 0$.

Data $\vec{x}_i$ yang berkorelasi dengan $\alpha_i > 0$ inilah yang disebut sebagai *support vector*. Dengan demikian, dapat diperoleh nilai yang nantinya digunakan untuk menemukan *w*. Solusi bidang pemisah didapatkan dengan rumus $w = \Sigma \alpha_i y_i x_i$; $b = y_k - w^T x_k$ untuk setiap x_k , dengan $\alpha_k \neq 0$.

Proses pengujian atau klasifikasi dilakukan juga pada setiap SVM biner menggunakan nilai *w*, *b*, dan x_i yang dihasilkan pada proses pelatihan di setiap SVM biner. Fungsi yang dihasilkan untuk proses pengujian adalah

$$f_i = K(x_i, x_d) w_i + b_i, \quad (43)$$

dimana : i = 1 sampai k; x_i = *support vector;* x_d = data pengujian. Outputnya adalah berupa indeks i dengan f_i terbesar yang merupakan kelas dari data pengujian.

3.6 Implementasi Perangkat Lunak

Beberapa potongan kode Matlab sebagai implementasi dari proses-proses dan algoritma-algoritma yang telah dijelaskan pada bagian sebelumnya.

3.6.1 Membaca citra data pelatihan dan citra data pengujian

```
1  function [Covar_train, Covar_test] = buildCovar(nots, noc)
2   %nots : jumlah data training, noc : banyaknya kelas
3  C=pwd;
4  cd([C, '\Messidor1']);
5  % Membaca dataset retina untuk data training
6  counter=0;
7  for i=1:noc
8      for j=1:nots
9          file=['retina' int2str((i-1)*25+j) '.bmp'];
10         [retina,MAP]=imread(file);
11         [m,n]= size(grayretina);
12         vector_retina=reshape(grayretina,m*n,1);
13         counter=counter+1;
14         Covar_train(:,counter)=vector_retina;
15     end
16 End
17 Covar_train=double(Covar_train)/255;
18 % Membaca database retina untuk data testing
19 counter=0;
20 for i=1:noc
21     for j=nots+1:25
22         file=['retina' int2str((i-1)*25+j) '.bmp'];
23         [retina,MAP]=imread(file);
24         [m,n]=size(grayretina);
25         vector_retina=reshape(grayretina,m*n,1);
26         counter=counter+1;
27         Covar_test(:,counter)=vector_face;
28     end
29 End
30 Covar_test=double(Covar_test)/255;
31 cd(C);
```

Pada program atau proses diatas merupakan proses untuk membaca citra data pelatihan dan citra data pengujian. Jika pada masing-masing kelas menggunakan data pelatihan (*number of training set* atau nots) sepuluh data citra, maka sisanya digunakan sebagai data pengujian.

3.6.2 Fungsi untuk melakukan ekstraksi ciri

Menghitung rata-rata kelas dan rata-rata global

```
1  function [R, L]=iterative2DLDA(Covar_train, LabelTrain, p,
```

```
    q, r, c)
2   % Covar_train : data training
    % LabelTrain : label dari data training,LabelTrain=1
    menunjukkan kelas 1
    % r : ukuran baris pada gambar dan c : ukuran kolom pada
    gambar, N=r*c
    % q : right projected dimension, p : left projected
    dimension
    % R : right projected vectors, L : left projected vectors
    % Iterasi sebanyak 10
3   [m,n]=size(Covar_train);
4   ClassNumber=max(LabelTrain);
5   for i=1:ClassNumber
6       temp=find(LabelTrain==i);
7       temp1=temp';
8       [m1, n1]=size(temp1);
9       Trainset1=Covar_train(:,temp1);
10      aa(:,i)=mean(Trainset1'); %Mencari rata2 kelas
11  End
12  bb=mean(Covar_train'); %Mencari rata2 global
13  bb1=bb';
```

Sebelum menghitung matrik *within class scatter* dan matrik *between class scatter* terlebih dahulu dicari matrik rata-rata kelas dan matrik rata-rata global.

Menentukan matrik transformasi kolom R yang nanti digunakan untuk menghitung S_w^R dan S_b^R

```
14  R=[eye(q,q)
15     zeros(c-q,q)]; %Mendapatkan nilai R untuk perhitungan
    rumus: S_w^R, S_b^R
```

Menetapkan matrik transformasi *R* ukuran (c, q) yang diperoleh dari gabungan antara matrik identitas ukuran (q, q) dengan matrik nol ukuran (c-q, q).

Melakukan proses perhitungan S_w^R, S_b^R

```
16  for j=1:4  %Banyaknya iterasi
17      sb1=zeros(r,r);
18      sw1=zeros(r,r);
19      for i=1:ClassNumber
20          temp=find(LabelTrain==i);
21          temp1=temp';
22          [m1, n1]=size(temp1);
23          Trainset1=Covar_train(:,temp1);
24          [m2,n2]=size(Trainset1);
25          for s=1:n2
26              sw1=sw1+(reshape(Trainset1(:,s), r,c)-
    reshape(aa(:,i), r,c))*R*R'*(reshape(Trainset1(:,s), r,c)-
```

```
    reshape(aa(:,i), r,c))';
27  % Berdasarkan rumus :  S_w^R = sum_{i=1}^{k} sum_{X in Pi_i} (X - M_i) R R^T (X - M_i)^T
28          end
29          sb1=sb1+n1*(reshape(aa(:,i), r,c)-reshape(bb1,
    r,c))*R*R'*(reshape(aa(:,i), r,c)-reshape(bb1, r,c))';
30  % Berdasarkan rumus :  S_b^R = sum_{i=1}^{k} n_i (M_i - M) R R^T (M_i - M)^T
31      end
```

Menghitung S_w^R yang rumusnya berdasarkan persamaan (18) dan S_b^R yang rumusnya berdasarkan persamaan (19).

Menghitung vektor eigen dan nilai eigen dari $(S_w^R)^{-1}(S_b^R)$

```
32    [U,S] =eig(pinv(sw1)*sb1);
        tt=diag(S);
        [B,IX]=sort(tt,'descend');
        U11=U(:,IX);    %Mencari eigenvector dan eigenvalue
dari (S_w^R)^-1(S_b^R)
33    L=U(:,1:p); % Nilai L = eigenvector sebanyak p kolom yang
sesuai dengan p eigenvalue terbesar, L digunakan untuk
perhitungan rumus: S_w^L, S_b^L
```

Vektor eigen dan nilai eigen pada proses diatas didapatkan berdasarkan pada persamaan (10) dimana hasilnya digunakan untuk menentukan ciri ekstraksi pada citra pelatihan dan citra pengujian.

Melakukan proses perhitungan S_w^L dan S_b^L

```
34      sb2=zeros(c,c);
35      sw2=zeros(c,c);
36      for i=1:ClassNumber
37          temp=find(LabelTrain==i);
38          temp1=temp';
39          [m1, n1]=size(temp1);
40          Trainset1=Covar_train(:,temp1);
41          [m2,n2]=size(Trainset1);
42          for s=1:n2
43              sw2=sw2+(reshape(Trainset1(:,s), r,c)-
reshape(aa(:,i), r,c))'*L*L'*(reshape(Trainset1(:,s), r,c)-
reshape(aa(:,i), r,c));
44
```

	$S_w^L = \sum_{i=1}^{k} \sum_{X \in \Pi_i} (X - M_i)^T LL^T (X - M_i)$ % Berdasarkan rumus :
45	end
46	sb2=sb2+n1*(reshape(aa(:,i), r,c)-reshape(bb1, r,c))'*L*L'*(reshape(aa(:,i), r,c)-reshape(bb1, r,c));
47	$S_b^L = \sum_{i=1}^{k} n_i (M_i - M)^T LL^T (M_i - M)$ % Berdasarkan rumus :
48	end

Menghitung S_w^L yang rumusnya berdasarkan persamaan (21) dan S_b^L yang rumusnya berdasarkan persamaan (22).

Menghitung vektor eigen dan nilai eigen dari $(S_w^L)^{-1}(S_b^L)$

49	[U1,S1] =eig(pinv(sw2)*sb2); tt1=diag(S1); [B,IX1]=sort(tt1,'descend'); U12=U1(:,IX1); %Mencari eigenvector dan eigenvalue dari $(S_w^L)^{-1}(S_b^L)$
50	R=U1(:,1:q); % Mengupdate nilai R = eigenvector sebanyak q kolom yang sesuai dengan q eigenvalue terbesar
51	end

Vektor eigen dan nilai eigen pada proses diatas, hasilnya digunakan untuk menentukan ciri ekstraksi pada citra pelatihan dan citra pengujian.

Menghitung ciri ekstraksi data pelatihan dan data pengujian

1	[m1,n1]=size(Covar_Train);
2	[m2,n2]=size(Covar_Test);
3	[m3,n3]=size(R);
4	MTrain=[];
5	MTest=[];
6	for i=1:n1
7	Temp=reshape(Covar_Train(:,i), row, col);
	MTrain(:,:,i)=L'*Temp*R; %Proses menghitung ciri ekstraksi data pelatihan
8	end
9	MatTrain=[];
10	[x y z]=size(MTrain);
11	for j=1:z
12	a = MTrain(:,:,j)';
13	MatTrain(j,:)= a(:)'; %Matrik ciri ekstraksi data pelatihan
14	end

```
15          for j=1:n2
16              Temp=reshape(Covar_Test(:,j), row, col);
17              MTest(:,:,j)=L'*Temp*R; %Proses menghitung ciri
    ekstraksi data pengujian
18          end
19          MatTest=[];
20          [x y z]=size(MTest);
21          for i=1:z
22              a = MTest(:,:,i)';
23              MatTest(i,:)= a(:)'; %Matrik ciri ekstraksi data
    pengujian
24          end
```

Menghitung matrik ekstraksi ciri data pelatihan berdasarkan rumus $B_i = L^T A_i R$, dimana B_i adalah matrik ekstraksi ciri data pelatihan dan A_i adalah matrik data pelatihan. Menghitung matrik ciri data pengujian berdasarkan rumus $D = L^T C R$, dimana D adalah matrik ekstraksi ciri data pengujian dan C adalah matrik data pengujian.