TUGAS AKHIR –KI091391

UJI KINERJA PENGGABUNGAN ALGORITMA *SUPPORT VECTOR MACHINE* DAN *SIMULATED ANNEALING* PADA PERMASALAHAN KLASIFIKASI POLA

ASTRIS DYAH PERWITA
NRP 5110100178

Dosen Pembimbing I
Dr.Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Rully Soelaiman, S.Kom, M.Kom.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2014

UNDERGRADUATE THESES – KI091391

PERFORMANCE TEST OF HYBRID SUPPORT VECTOR MACHINE AND SIMULATED ANNEALING FOR PATTERN RECOGNITION PROBLEM

ASTRIS DYAH PERWITA
NRP 5109100178

First Supervisor
Dr.Chastine Fatichah, S.Kom., M.Kom.

Second Supervisor
Rully Soelaiman, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2014

UJI KINERJA PENGGABUNGAN ALGORITMA *SUPPORT VECTOR MACHINE* DAN *SIMULATED ANNEALING* PADA PERMASALAHAN KLASIFIKASI POLA

Nama	: Astris Dyah Perwita
NRP	: 5110100178
Jurusan	: Teknik Informatika – FTIf ITS
Dosen Pembimbing I	: Dr.Chastine Fatichah, S.Kom., M.Kom.
Dosen Pembimbing II	: Rully Soelaiman, S.Kom., M.Kom.

ABSTRAK

Analisis dan pengenalan pola merupakan salah satu metode yang bertugas untuk mengenali pola yang ada pada data dan memegang peranan penting terhadap permasalahan kecerdasan buatan dan ilmu komputer. Salah satu contoh dari permasalahan pengenalan pola adalah klasifikasi data biomedik seperti data Hepatitis dan Breast Cancer. Data biomedik ini membutuhkan akurasi yang tinggi karena hasil keputusan yang diambil merupakan sebuah keputusan yang menyangkut kesehatan seseorang.

Pada Tugas Akhir ini, akan diterapkan metode Support Vector Machine untuk menyelesaikan klasifikasi pada permasalahan pengenalan pola khususnya data biomedik. Metode SVM akan digabungkan dengan algoritma Simulated Annealing sebagai algortima pemilihan parameter SVM. Tugas Akhir ini juga mengimplementasikan metode pembobotan kernel Gradient Descent untuk memperbaiki akurasi. Penggabungan metode SVM, SA, dan pembobotan kernel pada data uji akan menghasilkan akurasi hingga 98,75%. Berdasarkan hasil uji coba, penggabungan metode

SVM, SA, serta pembobotan kernel dapat memaksimalkan akurasi klasifikasi.

Kata kunci: Klasifikasi, Support Vector Machine, Simulated Annealing, Optimasi Parameter.

PERFORMANCE TEST OF HYBRID SUPPORT VECTOR MACHINE AND SIMULATED ANNEALING FOR PATTERN RECOGNITION PROBLEM

Name	: Astris Dyah Perwita
NRP	: 5110100178
Department	: Teknik Informatika – FTIf ITS
Supervisor I	: Dr.Chastine Fatichah, S.Kom., M.Kom.
Supervisor II	: Rully Soelaiman, S.Kom., M.Kom.

ABSTRACT

Analysis and pattern recognition is one of the methods assigned to recognize the data pattern and plays an important role in artificial intelligence and computer science problem. One example of the problem is pattern recognition in biomedical data classification such as data Hepatitis and Breast Cancer. The biomedical data requires high accuracy since the results of the decision will affect someone's health.

In this final project, Support Vector Machine method will be applied to solve classification problems for some dataset, especially in biomedical pattern recognition. SVM methods will be combined with Simulated Annealing algorithm as SVM algorithm parameter selection. The final project also implements the Gradient Descent method as kernel weighting method to improve the accuracy. Hybrid of SVM method, SA, and weighted kernel on the test data will generate up to 98.75% accuracy. Based on the test results, the hybrid of SVM method, SA, as well as the weighted kernel can maximize classification accuracy.

Keywords: Classification, Support Vector Machine, Simulated Annealing, Parameter Optimization.

LEMBAR PENGESAHAN

IMPLEMENTASI PENGGABUNGAN ALGORITMA *SUPPORT VECTOR MACHINE* DAN *SIMULATED ANNEALING* PADA PERMASALAHAN KLASIFIKASI POLA

TUGAS AKHIR
Diajukan Untuk Memenuhi Salah Satu Syarat Memperoleh
Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Jurusan Teknik Informatika
Fakultas Teknologi Informasi

Oleh:
ASTRIS DYAH PERWITA
NRP: 5110 100 178

Disetujui oleh Pembimbing Tugas Akhir:

Dr.Chastine Fatichah, S.Kom., M.Kom. .....
NIP : 132.298.829 (Pembimbing 1)

Rully Soelaiman, S.Kom, M.Kom. .....
NIP : 132.085.802 (Pembimbing 2)

SURABAYA
JUNI 2014

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "*Uji Kinerja Penggabungan Algoritma Support Vector Machine dan Simulated Annealing Pada Permasalahan Klasifikasi Pola*" dengan tepat waktu.

Harapan dari penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata bagi kampus Teknik Informatika, ITS, dan bangsa Indonesia.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku dosen pembimbing penulis yang telah memberikan bimbingan, saran, kritik, dan ilmu yang sangat bermanfaat hingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Ibu Dr.Chastine Fatichah, S.Kom., M.Kom. selaku dosen pembimbing yang telah memberikan nasihat, arahan, dan bimbingan dengan penuh kesabaran sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Ibu Dr. Nanik Suciati selaku ketua jurusan Teknik Informatika ITS, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya
4. Mas Dommy Asfiandy yang bersedia membagi ilmu melalui tugas akhirnya yang berkaitan dengan metode yang diimplementasikan pada Tugas Akhir ini.

5. Keluarga dan teman-teman penulis yang waktunya tersita selama proses pengerjaan Tugas Akhir ini.
6. Juga tak lupa kepada semua pihak yang belum sempat disebutkan satu per satu di sini yang telah membantu terselesaikannya tugas akhir ini.

Tugas Akhir ini merupakan persembahan penulis untuk kedua orang tua penulis yang selalu mengingatkan untuk menuntut ilmu setinggi-tingginya.

Kesempurnaan tentu masih jauh tercapai pada Tugas Akhir ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca untuk perbaikan ke depan. Semoga Tugas Akhir ini dapat bermanfaat bagi perkembangan ilmu pengetahuan dan bagi semua pihak.

Surabaya, Juni 2014

Astris Dyah Perwita

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

DAFTAR VARIABEL

Nomor	Variabel	Keterangan
1	D	Representasi kumpulan data
2	x_i	Sampel data ke-*i*
3	y_i	Label data ke-*i*
4	w	Nilai vektor normal terhadap *hyperplane*
5	B	Nilai *offset* dari *hyperplane* terhadap garis normal
6	$\frac{1}{\|\|w\|\|}$	Jarak antara data terdekat atau *support vector* terdekat dengan *hyperplane*
7	α	Nilai Lagrange Multiplier
8	C	Variabel yang menentukan penalti pada hasil klasifikasi dengan *soft margin*
9	ξ	*Slack Variable*, digunakan untuk menghitung penalti dari *decision boundary*dengan *soft margin*
10	$\emptyset(x)$	Fungsi transformasi dimensi pada SVM
11	$K(x_i x_j)$	Fungsi *kernel trick*
12	Σ	Parameter *kernel* RBF
13	SV	Subset dari data pelatihan yang terpilih sebagai *support vector*
14	E	*Searching speed* pada metode Gradient Descent
15	γ	Margin pada SVM
16	P	Probabilitas Boltzman

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Analisis pola merupakan salah satu permasalahan yang sering ditemui di semua bidang. Analisis pola bertugas untuk mengenali pola yang ada pada data dan memegang peranan penting terhadap permasalahan kecerdasan buatan dan ilmu komputer. Deteksi sebuah pola yang signifikan dari sebuah data dapat menjadi prediksi dan ramalan keadaan selanjutnya pada jenis data yang sama [1].

Support Vector Machine (SVM) merupakan salah satu metode yang biasanya digunakan untuk menyelesaikan permasalahan analisis pola. SVM ini merupakan algoritma yang dapat digunakan untuk memecahkan masalah klasifikasi seperti mendiagnosis suatu penyakit karena memiliki kemampuan prediksi yang relatif baik. Selain itu *error* dari hasil klasifikasi SVM relatif kecil, cocok untuk permasalahan berdimensi tinggi, dan memiliki landasan teori yang dapat dianalisis dengan jelas. Poin-poin inilah yang menjadi keunggulan SVM sehingga metode ini sering digunakan untuk menyelesaikan permasalahan klasifikasi pola. Namun SVM memiliki beberapa kelemahan antara lain harus memperhatikan aspek-aspek seperti jenis *kernel* dan parameter. Parameter yang ada dalam SVM juga perlu diperhatikan agar dapat menghasilkan hasil yang optimal contohnya parameter C dan σ [2].

Algoritma Simulated Annealing (SA) merupakan salah satu algoritma yang pada Tugas Akhir ini mengoptimasi metode SVM dalam menentukan parameter. SA merupakan metode metaheuristik yang dapat mengatasi model permasalahan nonlinier. Selain itu metode ini juga dapat mengatasi data yang memiliki banyak batasan. SA adalah metode yang fleksibel atau tidak bergantung pada ketentuan atau model tertentu. Namun

pada aplikasinya, algoritma ini membutuhkan banyak pengaturan dan akan menambah waktu komputasi.

Pada Tugas Akhir ini diharapkan metode SA akan menghasilkan parameter terbaik untuk metode SVM sehingga menghasilkan sebuah sistem yang baik untuk permasalahan klasifikasi, dapat diimplementasikan dengan mudah, dan menghasilkan akurasi yang lebih baik daripada tanpa menggabungkan kedua metode [3].

1.2 Rumusan Masalah

Rumusan masalah yang diangkat pada Tugas Akhir ini adalah sebagai berikut:

1. Penerapan algoritma Simulated Annealing untuk mengoptimasi parameter SVM yaitu C dan σ.
2. Penerapan algoritma Simulated Annealing untuk mengoptimasi parameter dengan SVM yang telah dibobotkan *kernel*-nya untuk memperbaiki hasil klasifikasi.
3. Pelaksanan uji coba atas algoritma dan metode yang telah diimplementasikan pada permasalahan klasifikasi pola.

1.3 Batasan Masalah

Implementasi dari metode pada perangkat lunak yang dibangun pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. *Dataset* yang digunakan adalah Hepatitis dan Breast Cancer yang keduanya didapatkan dari situs UC Irvine Machine Learning Repository.
2. Implementasi dilakukan pada Matlab 2008a
3. Parameter yang dioptimasi dengan algoritma Simulated Annealing adalah parameter C dan σ.
4. Nilai *missing value* pada *dataset* dihapus karena fokus pada Tugas Akhir ini adalah optimasi parameter Support Vector Machine dengan metode Simulated Annealing

1.4 Tujuan

Tugas Akhir ini memiliki beberapa tujuan sebagai berikut:

1. Mengimplementasikan algoritma SA untuk mengoptimasi parameter SVM
2. Mengimplementasikan algoritma SA untuk mengoptimasi parameter SVM yang telah dibobotkan *kernel*-nya untuk memperbaiki hasil klasifikasi.
3. Melakukan uji coba dan analisis kinerja dari algoritma yang telah diimplementasikan.

1.5 Metodologi

Tahap yang dilakukan untuk menyelesaikan Tugas Akhir ini adalah sebagai berikut:

1. **Penyusunan Proposal Tugas Akhir**

   Penyusunan proposal ini merupakan tahap awal dalam pengerjaan Tugas Akhir. Proposal ini berisi gambaran secara umum percobaan yang akan diimplementasikan nantinya. Proposal ini juga menjelaskan rencana implementasi percobaan klasifikasi dengan menggunakan metode SVM-SA dan pembobotan *kernel*.

2. **Studi Literatur**

   Pada tahap ini dilakukan pencarian informasi dan studi literatur yang digunakan untuk pemrosesan data dan desain perangkat lunak yang akan dibuat. Informasi didapatkan dari buku dan literatur lain yang berhubungan dengan algoritma yang digunakan dalam pengerjaan Tugas Akhir. Informasi yang dicari dan dipahami antara lain konsep Support Vector Machine, algoritma Simulated Annealing, dan metode pembobotan *kernel*.

3. **Implementasi**

   Implementasi merupakan tahap pembangunan perangkat lunak yang mengimplementasikan algoritma-algoritma yang sudah diajukan. Sesuai dengan rancangan yang diajukan pada proposal, pembangunan perangkat

lunak diimplementasikan sesuai dengan konsep yang telah didapatkan saat studi literatur.

4. **Pengujian dan Evaluasi**

   Pada tahap ini dilakukan uji coba dengan *dataset* yang direkomendasikan oleh jurnal dan sebuah *dataset* bebas lainnya untuk perbandingan. Uji coba ini dilakukan untuk membuktikan bahwa perangkat lunak yang dibangun telah bekerja sesuai dengan tujuan dan menjadi solusi dari permasalahan.

5. **Penyusunan Buku Tugas Akhir**

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang diterapkan dalam Tugas Akhir. Buku Tugas Akhir merupakan dokumentasi perangkat lunak yang mencakup teori, penerapan, serta kesimpulan dari perangkat lunak yang dibangun.

1.6 Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

1. **Bab I. Pendahuluan**

   Bab ini berisi penjelasan mengenai latar belakang, masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Bab ini juga berisi rumusan masalah, batasan masalah, dan sistematika penulisan buku Tugas Akhir.

2. **Bab II. Kajian Teori**

   Bab ini berisi tentang teori yang digunakan dan diimplementasikan pada Tugas Akhir. Teori-teori ini mencakup konsep dasar, persamaan yang digunakan, dan ilustrasi dari algoritma maupun permasalahan yang berhubungan dengan algoritma tersebut. Kajian teori utama yang dijelaskan pada bab ini antara lain adalah teori pengenalan pola, SVM, SA, dan pembobotan *kernel*.

3. **Bab III. Desain dan Perancangan Perangkat Lunak**

   Bab ini merupakan gambaran perangkat lunak secara umum. Pada bab ini dijelaskan tahapan-tahapan

implementasi dengan menggunakan diagram alir, penjelasan variabel, serta *pseudocode*.

4. **Bab IV. Implementasi**

   Bab ini menjelaskan tentang pembangunan aplikasi, penjelasan fungsi-fungsi yang digunakan, dan kode Matlab yang diaplikasikan agar perangkat lunak berjalan sesuai dengan rencana yang diajukan.

5. **Bab V. Uji Coba dan Analisis Hasil**

   Bab ini berisi hasil pemrosesan data dengan menggunakan perangkat lunak yang telah dibangun. Pada bab ini juga disertakan analisis dari hasil perangkat lunak.

6. **Bab VI. Kesimpulan dan Saran.**

   Bab ini merupakan penjelasan dari hasil akhir yang dapat ditarik dari keseluruhan proses dan percobaan Tugas Akhir. Pada bab ini juga merupakan penjelasan atas permasalahan yang ada sebelum percobaan ini dilakukan. Pada bab ini juga dituliskan saran-saran yang berisi hal-hal yang masih dapat diperbaiki dan dikembangkan.

BAB II
KAJIAN TEORI

2.1 Pengenalan Pola

Pengenalan pola adalah sebuah cabang ilmu pengetahuan yang berfungsi untuk mengklasifikasikan satu atau lebih objek ke dalam beberapa kategori atau yang biasa disebut *class* atau label. Objek-objek ini bisa berbentuk gambar, gelombang, atau berbagai tipe data yang dapat diklasifikasikan. Objek-objek inilah yang biasa disebut pola. Pengenalan pola saat ini menjadi penting seiring perkembangan mesin-mesin yang membutuhkan kecerdasan atau *intelligence* termasuk untuk mengenali pola dan mengambil keputusan.

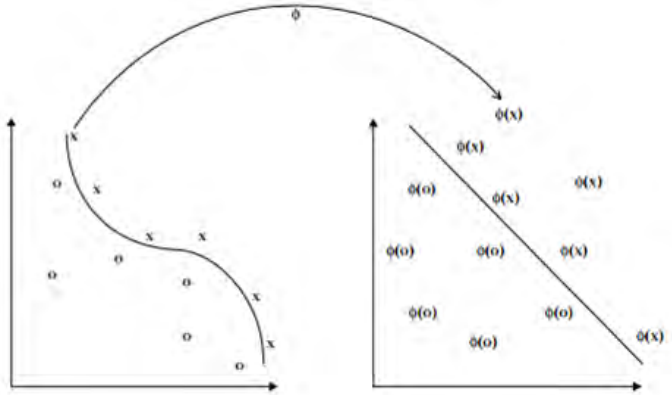
Pengenalan pola dalam bidang kesehatan juga mengalami banyak perkembangan. Pengenalan pola pada bidang kesehatan selama ini digunakan untuk mengenali dan membandingkan rangkaian DNA, mengidentifikasikan sidik jari, sampai pengenalan raut wajah serta gerakan manusia. Pemanfaatan yang sering digunakan untuk masyarakat umum adalah mengidentifikasikan keadaan sehat dan sakit pasien.

Pola sendiri adalah representasi dari data-data yang berelasi baik relasi data yang identik, berdekatan, atau berelasi secara statistik [1].

Proses pencocokan sebuah data baru terhadap pola yang telah terbentuk dapat juga disebut klasifikasi atau biasa disebut *supervised problem*. Properti yang digunakan sebagai acuan pada proses klasifikasi pola disebut dengan fitur. Pada sebagian besar kasus, klasifikasi akan menggunakan lebih dari satu fitur sehingga kumpulan fitur ini biasa disebut vektor fitur (*feature vector*). Kumpulan data pada suatu pola biasanya akan dipisahkan oleh sebuah garis yang disebut *decision line* atau *decision boundary* [4].

Perubahan koordinat pada data-data yang berelasi akan menyebakan perubahan pada pola. Perubahan koordinat ini dapat dieksekusi dalam sebuah fungsi pola. Pengubahan koordinat

dengan fungsi pola ini juga akan berpengaruh terhadap persebaran data. Pengubahan persebaran data ini biasa disebut *mapping* atau transformasi data.Fungsi pola dapat mengubah data-data yang berelasi ini menjadi bentuk yang lebih mudah untuk dipisahkan menurut properti atau sifatnya. Fungsi pola untuk transformasi ini dibentuk sedemikian rupa sehingga data dapat dipisahkan secara linier seperti ilustrasi pada Gambar 2.1



Gambar 2.1 Aplikasi Fungsi Pola Ø Terhadap Data Klasifikasi [1].

Permasalahan pengenalan pola dapat diselesaikan dengan menerapkan *classifier* atau pemisah yang tepat pada data-data yang dituju. *Classifier* ini didapatkan dengan beberapa metode antara lain *linear classifier* dan *nonlinear classifier*.

2.1.1 Permasalahan Analisis Pola

Tujuan dari analisis pola secara umum adalah memprediksi bahwa sebuah fitur dari data memiliki fungsi yang berhubungan dengan nilai fitur yang lain. Oleh karena itu, banyak analisis pola yang menitikberatkan sebuah fitur sebagai tujuan prediksi. Dari sebuah bentuk data (*x*,*y*), variabel *y* merupakan fitur yang dituju untuk diprediksi dan *x* adalah vektor yang mengandung nilai-nilai fitur yang lain. Variabel *x* merupakan variabel yang biasa

dianggap masukan sedangkan *y* merupakan variabel yang biasa disebut target keluaran atau label.

2.1.1.1 Permasalahan Supervised

Data dapat dianggap data uji ketika data tersebut memiliki fitur tujuan yang digunakan sebagai acuan untuk prediksi. Analisis pola dengan bentuk seperti ini merupakan *supervised task* karena setiap masukan memiliki label yang berkaitan. Permasalahan ini biasa disebut klasifikasi.

Permasalahan analisis pola seperti ini dapat dibedakan sesuai dengan jenis target atau label yang tersedia. Pada permasalahan *binary classification*, vektor masukan akan tergolong pada dua kategori. Sedangkan ketika label termasuk dalam sebuah set {1,2,…,*N*} kategori, maka permasalahan ini dapat disebut *multiclass classification* [1].

Pada permasalahan ini, setiap proses terdiri atas beberapa observasi dan fitur khusus pada observasi ini menjadi nilai untuk observasi berikutnya. Oleh karena itu, tujuan dari analisis pola jenis ini adalah membuat perkiraan berdasarkan nilai-nilai fitur relevan pada observasi sebelumnya.

Regresi juga termasuk *supervised task* untuk analisis pola. Salah satu jenis regresi adalah *time-series analysis*. Pada kasus ini, setiap percobaan akan menjadi acuan percobaan berikutnya. Tujuan dari analisis pola pada kasus ini adalah menghasilkan ramalan atas seri berikutnya [4].

2.1.1.2 Permasalahan Semisupervised

Beberapa permasalahan pola ditunjukkan dengan fitur yang tidak diketahui seluruhnya. Contohnya pada kasus *ranking*, informasi yang tersedia hanya urutan dari data yang digunakan sebagai data pelatihan sedangkan tujuan penyelesaian yang harus dilakukan adalah membangkitkan urutan yang sama terhadap data uji.

Permasalahan lainnya adalah permasalahan *transduction*. Pada permasalahan ini hanya beberapa data yang memiliki nilai

label. Tujuan analisis pola pada permasalahan ini adalah memprediksi label dari data yang tidak memiliki label [1].

2.1.1.3 Permasalahan Unsupervised

Bertolak belakang dengan *supervised task*, *unsupervised task* memiliki data-data yang sama sekali tidak memiliki label. Informasi atau pola harus diekstraksi tanpa adanya 'informasi eksternal' yang biasa disediakan oleh label.

Clustering adalah salah satu permasalahan dari *unsupervised task*. Tujuan dari *clustering* adalah menemukan pembagian sesuai dengan sifat data menjadi grup-grup yang homogen. Selain itu, terdapat permasalahan *anomaly* atau *novelty-detection*. Permasalahan ini bertujuan untuk mendeteksi data yang bersifat berbeda dari data normal [1].

2.2 Support Vector Machine

Support Vector Machine atau SVM pertama kali dicetuskan oleh Vladimir N. Vapnik dan dipresentasikan pada tahun 1992 di Annual Workshop on Computational Learning Theory. Konsep SVM adalah pengambilan keputusan berdasarkan bidang-bidang terbatas. Setiap keputusan direpresentasikan dalam sebuah bidang yang memiliki karakteristik sama di setiap *dataset*. Berbeda dengan konsep Neural Network yang mencari *hyperplane* pemisah antar *class*.

SVM pada awalnya merupakan metode *linier classifier* dengan ruang dimensi sederhana. SVM kemudian dikembangkan hingga dapat menyelesaikan permasalahan nonlinier berdimensi tinggi dengan memasukkan konsep *kernel trick*. Ide menyertakan konsep *kernel* ini adalah mengubah persebaran data dengan menggunakan *kernel* sebagai set fungsi matematika.

Dewasa ini SVM telah berhasil diaplikasikan dalam permasalahan dunia nyata dan secara umum memberikan solusi yang lebih baik daripada metode konvensional seperti *artificial neural network*.

2.2.1 Klasifikasi dengan SVM

Pada bidang yang terbentuk dari pola data, SVM akan membentuk *hyperplane* yang memisahkan 2 *class*. Bentuk yang paling sederhana pada SVM adalah Linier SVM. Sebagai acuan, terdapat sebuah training set *D* yang dapat dinotasikan dengan Persamaan 2.1

$$D = \{(x_i, y_i) | x_i \in R^p, y_i \in \{-1,1\}\}_{i=1}^{n} \quad (2.1)$$

Nilai $y_i \in \{-1,1\}$ merupakan penunjuk *class* atau label yang bernilai 1 atau -1 dari kumpulan data x_i. *Hyperplane* yang terbaik merupakan *hyperplane* yang memisahkan data dengan $y_i = 1$ dan $y_i = -1$. *Hyperplane* yang memisahkan tersebut dapat dinotasikan dengan Persamaan 2.2.

$$w \cdot x - b = 0 \quad (2.2)$$

Titik data x_i yang termasuk kelas -1 dapat dinotasikan sebagai titik data yang memenuhi Pertidaksamaan 2.3 dan titik data x_i yang termasuk kelas 1 dapat dinotasikan dengan Pertidaksamaan 2.4.

$$w \cdot x - b \geq 1, y_i = 1 \quad (2.3)$$

$$w \cdot x - b \leq -1, y_i = -1 \quad (2.4)$$

Hyperplane yang ideal adalah *hyperplane* yang mempunyai margin yang maksimal dalam memisahkan kelas-kelas data. Sesuai dengan penerapan rumus perhitungan jarak antara titik dengan garis, maka jarak antara data terdekat dengan batas tengah *hyperplane* adalah $\frac{1}{||w||}$. Untuk memaksimalkan margin, maka nilai $||w||$ harus diminimalisasi. Peminimalisasian nilai $\left||w|\right|$ sama artinya dengan menyelesaikan permasalahan *quadratic programming* [1]. Permasalahan *quadratic programming* ini akan mencari titik minimal Persamaan 2.5 dengan memperhatikan batasan Persamaan 2.6.

$$f: \frac{1}{2}||w||^2 \tag{2.5}$$

$$y_i(x_i.w + b) - 1 \geq 0, \forall_i \tag{2.6}$$

Permasalahan *quadratic programming* ini dapat dipecahkan dengan berbagai teknik komputasi. Salah satu metode pemecahannya adalah dengan menggunakan Lagrange Multiplier. Lagrange Multiplier menggunakan variabel α sehingga persamaan akan lebih mudah dihitung. Variabel α bernilai nol atau positif untuk Persamaan 2.7.

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i(y_i(x_i.w + b) - 1) \tag{2.7}$$

Nilai optimal dari Persamaan 2.7 dapat dihitung dengan meminimalkan L terhadap w dan b, dan memaksimalkan L terhadap α_i. Persamaan 2.7 juga dapat dimodifikasi sehingga hanya mengandung α_i dengan memperhatikan nilai $L = 0$ pada titik optimal gradien menjadi Persamaan 2.8 dengan batasan Persamaan 2.9. Data yang berkorelasi dengan α_i yang positif inilah yang disebut *support vector*.

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.8}$$

$$\alpha_i \geq 0 \text{ dan } \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.9}$$

2.2.2 Soft Margin

Terkadang dalam beberapa kasus, terdapat beberapa data pelatihan yang *error* atau terpisah dari yang lain baik karena sifatnya maupun dari nilai datanya. Kasus seperti ini biasa disebut dengan *non-separable case*. Kasus ini menyebabkan dua buah ruang masukan tidak dapat dipisahkan dengan sempurna sehingga pembentukan *decision boundary* dari kasus ini membutuhkan *soft margin*. *Soft margin* ini dapat menjadi toleransi data-data yang *error* tersebut sementara margin yang sebenarnya membentuk bidang/*plane* yang maksimum.

Pembentukan *decision boundary* akan membutuhkan *slack variable* (ξ) yang memberikan perkiraan *error* dari *decision boundary* terhadap data yang dilatih. Sementara untuk data pengujian, bidang yang terbentuk akan sangat lebar sehingga dapat diperkirakan banyak data pengujian yang tidak berhasil diklasifikasi. Oleh karena itu fungsi objektif harus dimodifikasi dengan parameter C dan k agar dapat memberi penalti dari nilai fungsi dengan *slack variable.*

Dalam aplikasi *soft margin*, Persamaan 2.10 merupakan modifikasi Persamaan 2.6 dengan memasukkan *slack variable* $\xi_i(\xi_i > 0)$ dan Persamaan 2.11 adalah modifikasi Persamaan 2.5 dengan cara yang sama. Parameter C dan k biasanya ditentukan oleh pengguna dan kedua parameter ini merepresentasikan nilai penalti dari kesalahan klasifikasi terhadap data pelatihan.

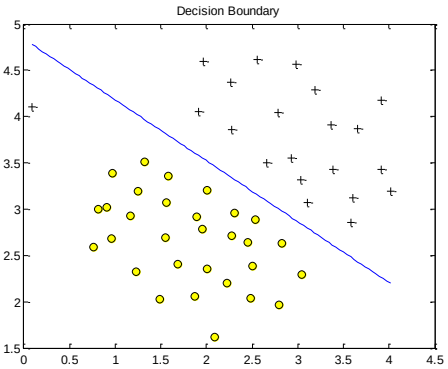
$$y_i(x_i.w + b) \geq 1 - \xi, \forall_i \tag{2.10}$$

$$f(x) = \frac{||w||^2}{2} + C(\sum_{i=1}^{N} \xi_i)^k \tag{2.11}$$

2.2.3 Nonlinier SVM dan Kernel Trick Pada SVM

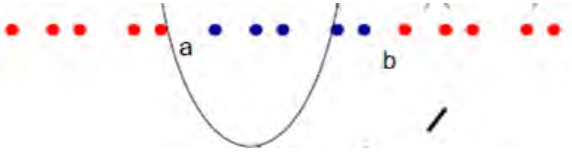
Pada dunia nyata, kasus *linearly separable* atau kasus data dapat dipisahkan secara linier seperti pada Gambar 2.2 jarang terjadi. Kasus yang terjadi pada umumnya bersifat nonlinier. Untuk menyelesaikan permasalahan nonlinier SVM dimodifikasi dengan memasukkan fungsi *kernel*.

Trik dalam mengerjakan nonlinier SVM adalah mentransformasi data dari ruang koordinat awal x menjadi ruang-ruang baru dengan fungsi $\emptyset(x)$ sehingga membentuk sebuah batasan linier yang dapat digunakan untuk memisahkan data-data yang diinginkan. Hal ini diterapkan agar selanjutnya dapat dilakukan metode pencarian batas bidang seperti pada proses linier SVM sebelumnya.



Gambar 2.2 Decision Boundary Pada Kasus Linearly Separable.

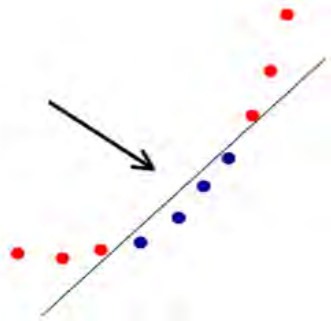
Misalnya terdapat data seperti pada Gambar 2.3. Penyelesaian yang dapat memisahkan kedua jenis data tersebut bukan penyelesaian linier melainkan penyelesaian kuadratik.



Gambar 2.3 Permasalahan Nonlinier dengan Penyelesaian Kuadratik [4].

Namun dengan menerapkan fungsi transformasi atau *mapping* dengan Persamaan 2.12 ke dalam *dataset*, kita akan mendapatkan pola data yang dapat dipisahkan dengan linier seperti Gambar 2.4.

$$\emptyset: (x) \rightarrow \{x^2, x\} \quad (2.12)$$

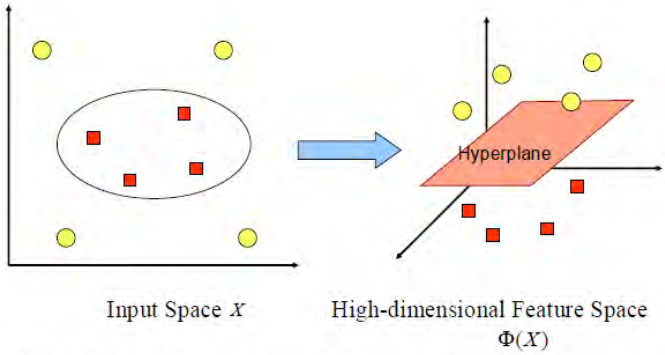


Gambar 2.4 Permasalahan Nonlinier dengan Fungsi Transformasi [4].

Selain itu, permasalahan ini dapat diselesaikan dengan memetakan data masukan ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini, *hyperplane* yang memisahkan kedua kelas tersebut dapat dibangun. Hal ini sejalan dengan Teori Cover yang menyatakan:
"Jika suatu transformasi bersifat nonlinier dan dimensi dari ruang fitur cukup tinggi, maka data pada ruang input dapat dipetakan ke ruang fitur yang baru, di mana titik-titik data tersebut pada probabilitas tinggi dapat dipisahkan secara linier"

Permasalahan transformasi ke ruang dimensi yang lebih tinggi ini dapat diilustrasikan pada Gambar 2.5. Data pada Gambar 2.5(a) berada pada ruang masukan berdimensi dua dan tidak dapat dipisahkan secara linier. Fungsi $\emptyset$ memetakan tiap data pada ruang masukan yang berdimensi dua tersebut ke ruang vektor baru yang berdimensi tiga. Pada ruang vektor yang berdimensi tiga, kedua *class* tersebut dapat dipisahkan secara linier oleh sebuah *hyperplane*. Notasi matematika dari pemetaan ini dijabarkan pada Persamaan 2.13.

$$\emptyset: \mathrm{R}^d \rightarrow \mathrm{R}^q, d < q \qquad (2.13)$$



Gambar 2.5 Fungsi $\emptyset$ memetakan data ke ruang vektor yang berdimensi lebih tinggi sehingga kedua kelas dapat dipisahkan secara linier oleh sebuah hyperplane [2].

Semakin rumit pola data yang tersedia, maka akan lebih sulit untuk menentukan jenis transformasi yang harus diaplikasikan terhadap pola data tersebut. Permasalahan bisa saja diselesaikan dengan cara mentransformasikan data ke dalam ruang dimensi tak hingga. Namun akan menimbulkan biaya komputasi yang besar dan berisiko terjebak dalam permasalahan *curse of dimensionality*.

Untuk menghindari permasalahan pemilihan fungsi transformasi, pembelajaran SVM selanjutnya difokuskan terhadap titik-titik *support vector*. Pembelajaran ini hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi yaitu $\emptyset(x_i).\emptyset(x_j)$. Karena umumnya transformasi $\emptyset$ ini tidak diketahui, dan sangat sulit untuk dipahami, maka sesuai dengan Teori Mercer, perhitungan *dot product* tersebut dapat digantikan dengan fungsi *kernel* (x_i, x_j) yang mendefinisikan secara implisit transformasi $\emptyset$.

Fungsi pengganti transformasi tersebut kemudian lebih sering disebut dengan *kernel trick*. *Kernel trick* dapat disebut juga sebagai metode untuk menghitung kesamaan dari ruang yang ditransformasi menggunakan atribut set awal dan nilai *dot*

product dari data. Secara umum, *kernel trick* dinotasikan dengan Persamaan 2.14.

$$K(x_i x_j) = \emptyset(x_i).\emptyset(x_j) \tag{2.14}$$

Kemudahan *kernel trick* ialah proses pembelajaran SVM tidak perlu mengetahui wujud dari fungsi nonlinier $\emptyset$ melainkan hanya cukup dengan mengetahui fungsi *kernel* yang dipakai. Ada beberapa jenis fungsi *kernel* yang sering digunakan seperti pada Tabel 2.1. Selanjutnya hasil klasifikasi dari data x akan diperoleh dengan Persamaan 2.15, 2.16, dan 2.17.

Selanjutnya hasil klasifikasi dari data x akan diperoleh dengan Persamaan 2.15, 2.16, dan 2.17. Variabel *SV* pada Persamaan 2.16 dan 2.17 merupakan subset dari data pelatihan yang terpilih sebagai *support vector* atau x_i yang berkorespondensi pada $\alpha_i \geq 0$.

Tabel 2.1 Kernel yang Biasa Digunakan dalam SVM

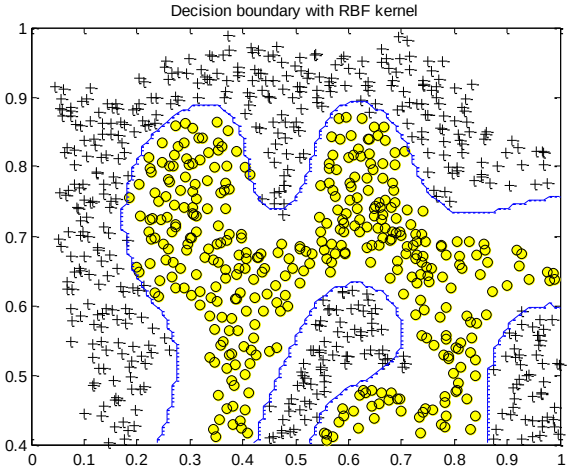
Nama Kernel	**Inner Product Kernel**
Polinomial	$(x^T x_i + 1)^p$ dengan nilai p bebas, ditentukan oleh pengguna
Radial-basis function (RBF/Gaussian)	$exp\left(-\frac{\|\|x-x'\|\|_2^2}{2\sigma^2}\right)$ dengan nilai σ bebas, ditentukan oleh pengguna
Two Layer Perceptron	$\tan h(\beta_0 x^T x_i + \beta_1)$

$$f(\emptyset(x)) = w.\emptyset(x) + \mathrm{b} \tag{2.15}$$

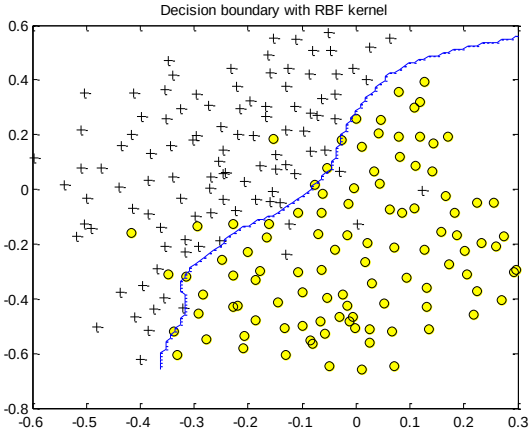
$$f(\emptyset(x)) = \sum_{i=1,x_i \in SV}^{l} \alpha_i y_i \emptyset(x).\emptyset(x_i) + b \tag{2.16}$$

$$f(\emptyset(x)) = \sum_{i=1,x_i \in SV}^{l} \alpha_i y_i K(x.x_i) + b \tag{2.17}$$

Hasil dari pembentukan *decision boundary* dengan menggunakan *kernel* RBF dapat diilustrasikan dengan Gambar 2.6 dan Gambar 2.7.



Gambar 2.6 Contoh 1 Decision Boundary dengan Kernel RBF.



Gambar 2.7 Contoh 2 Decision Boundary dengan Kernel RBF.

2.2.4 Kelebihan dan Kekurangan SVM

Dalam memilih solusi untuk menyelesaikan suatu masalah, kelebihan dan kelemahan masing-masing metode harus diperhatikan. Selanjutnya metode yang tepat dipilih dengan memperhatikan karakteristik data yang diolah. Dalam hal SVM, walaupun berbagai studi telah menunjukkan kelebihan metode SVM dibandingkan metode konvensional lain, SVM juga memiliki berbagai kelemahan. Kelebihan SVM antara lain sebagai berikut :

1. Generalisasi

Generalisasi didefinisikan sebagai kemampuan suatu metode (SVM, Neural Network, dan sebagainya) untuk mengklasifikasikan suatu data yang tidak termasuk sampel data yang dipakai dalam fase pembelajaran metode itu. Vapnik menjelaskan bahwa *generalization error* dipengaruhi oleh dua factor, yaitu *error* terhadap *training set*, dan satu faktor lagi yang dipengaruhi oleh dimensi VC (Vapnik-Chervokinensis).

Strategi pembelajaran pada *neural network* dan umumnya metode *learning machine* difokuskan pada usaha untuk meminimimalkan *error* pada *training set*. Strategi ini disebut *Empirical Risk Minimization* (ERM). Adapun SVM selain meminimalkan *error* pada *training set*, juga meminimalkan faktor kedua. Strategi ini disebut *Structural Risk Minimization* (SRM), dan dalam SVM diwujudkan dengan memilih *hyperplane* dengan margin terbesar.

Berbagai studi empiris menunjukkan bahwa pendekatan SRM pada SVM memberikan *error* generalisasi yang lebih kecil daripada yang diperoleh dari strategi ERM pada *neural network* maupun metode yang lain [2].

2. *Curse of Dimensionality*

Curse of dimensionality didefinisikan sebagai masalah yang dihadapi suatu metode pengenalan pola dalam mengestimasikan parameter (misalnya jumlah *hidden neuron* pada *neural network*, *stopping criteria* dalam proses pembelajaran, dsb.) dikarenakan jumlah sampel data yang relatif

sedikit dibandingkan dimensi ruang vektor data tersebut. Semakin tinggi dimensi dari ruang vektor dari informasi yang diolah, membawa konsekuensi dibutuhkannya jumlah data dalam proses pembelajaran.

Pada kenyataannya, sering terjadi data yang diolah berjumlah terbatas, dan untuk mengumpulkan data yang lebih banyak tidak mungkin dilakukan karena kendala biaya dan kesulitan teknis. Dalam kondisi tersebut, jika metode itu dipaksa harus bekerja pada data yang berjumlah relatif sedikit dibandingkan dimensinya, akan membuat proses estimasi parameter metode menjadi sangat sulit.

Curse of dimensionality sering dialami dalam aplikasi di bidang *biomedical engineering*, karena biasanya data biologi yang tersedia sangat terbatas, dan penyediaannya memerlukan biaya tinggi. Vapnik membuktikan bahwa tingkat generalisasi yang diperoleh oleh SVM tidak dipengaruhi oleh dimensi dari vektor masukan. Hal ini merupakan alasan mengapa SVM merupakan salah satu metode yang tepat dipakai untuk memecahkan masalah berdimensi tinggi, dalam keterbatasan sampel data yang ada [2].

3. Landasan Teori

Sebagai metode yang berbasis statistik, SVM memiliki landasan teori yang dapat dianalisis dengan jelas dan tidak bersifat *black box* [2].

4. Feasibility

SVM dapat diimplementasikan relatif mudah, karena proses penentuan *support vector* dapat dirumuskan dalam permasalahan *quadratic programming*. Dengan demikian jika kita memiliki *library* yang sesuai, dengan sendirinya SVM dapat diimplementasikan dengan mudah [2].

Disamping kelebihannya, SVM memiliki kelemahan atau keterbatasan, antara lain:

1. Sulit dipakai dalam permasalahan berskala besar.

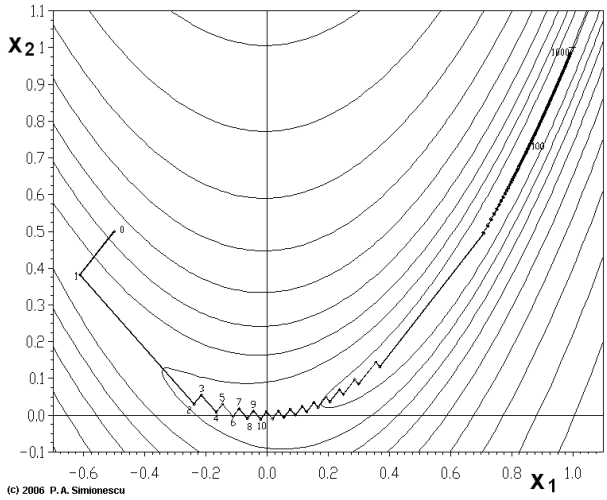
Skala besar dalam hal ini dimaksudkan dengan jumlah *sample* yang diolah.

2. SVM secara teoritik dikembangkan untuk problem klasifikasi dengan dua kelas.

Dewasa ini SVM telah dimodifikasi agar dapat menyelesaikan masalah dengan kelas lebih dari dua, antara lain strategi *One versus rest* dan strategi *Tree Structure*. Namun demikian, masing-masing strategi ini memiliki kelemahan, sehingga dapat dikatakan penelitian dan pengembangan SVM pada *multiclass-problem* masih merupakan tema penelitian yang masih terbuka [2].

2.3 Pembobotan Kernel dengan Metode Gradient Descent

Gradient Descent merupakan algoritma optimasi untuk menemukan *local minimum* atau nilai yang paling minimum dari sejumlah jangkauan nilai dari suatu fungsi dengan menggunakan langkah yang proporsional terhadap suatu nilai negatif dari gradien atau pendekatan gradien fungsi tersebut. Algoritma ini dapat diilustrasikan pada Gambar 2.8.



Gambar 2.8 Proses Gradient Descent Menemukan Local Minimum.

Pada metode Gradient Descent ada 2 parameter yang menentukan hasil akhir dari pembobotan, yaitu *e* sebagai *searching speed* dan *n* sebagai jumlah iterasi maksimal. Paramater *searching speed* menentukan kecepatan pencarian nilai *local minimum* dengan menjadi faktor pengali penurunan gradien pada tiap iterasi. Semakin besar nilai *searching speed* semakin besar pula penurunan gradien pada tiap iterasi tetapi jika nilai *searching speed* terlalu besar maka nilai *local minimum* akan terlewati sehingga tidak mendapatkan nilai yang diharapkan. Parameter *n* menentukan batas iterasi maksimal yang mungkin dilakukan untuk mencari nilai *local minimum*. Jika sebelum mencapai batas iterasi nilai *local minimum* sudah ditemukan maka iterasi akan berhenti [5].

Berdasarkan teori SVM, turunan margin γ dapat dituliskan dalam Persamaan (2.18).

$$\gamma = \frac{1}{||w||} \tag{2.18}$$

Pada Persamaan 2.18, w adalah vektor berdimensi *n* yang tegak lurus terhadap *hyperplane* pemisah yang dapat dapat direpresentasikan pada Persamaan 2.19.

$$w = \sum_{i=1}^{l} \alpha_i y_i \emptyset(x_i) \tag{2.19}$$

Dari Persamaan 2.18, dapat dilihat bahwa dengan memberikan ${\alpha_i}^0$ yang tetap, penurunan $||w||^2$ dapat dirumuskan dengan Persamaan 2.20

$$\frac{\partial ||w||^2}{\partial S_p} = \sum_{i,j=1}^{l} \alpha_i^0 \alpha_j^0 y_i y_j \frac{\partial K_{SW}(x_i, x_j)}{\partial S_p} \tag{2.20}$$

Nilai S_p adalah nilai vektor dari *dataset*. Berdasarkan Persamaan 2.20, maka estimasi bobot untuk setiap percobaan dapat diimplementasikan dengan Persamaan 2.21.

$$S_p(n+1) = S_p(n) - e\frac{\partial||w||^2}{\partial S_p} \quad (2.21)$$

Nilai *e* mengontrol kecepatan pencarian dan *n* adalah jumlah iterasi. Pada Persamaan 2.21, pembobotan akan diperbaharui setiap iterasi dan *local minimum* $||w||^2$ akan ditemukan setelah sejumlah iterasi [5]. Dan perumusan pembobotan pada *kernel* pada SVM dapat diimplementasikan seperti pada Persamaan 2.22

$$K_{SW}(x_i, x_j) = exp\left\{\frac{|S(x_i - x_j)|^2}{2\sigma^2}\right\} \quad (2.22)$$

2.4 Optimasi Parameter dengan Simmulated Annealing

Simulated Annealing adalah algoritma pencarian nilai optimal global yang pertama kali dikenalkan sebagai adaptasi dari algoritma Metropolis-Hasting dan kemudian dipopulerkan oleh Kirkpatrick. SA merupakan algoritma yang menggunakan konsep dasar metalurgi. Molekul pada besi secara bertahap akan mengkristal menjadi besi yang berenergi rendah saat suhunya perlahan turun. Semua bentuk kristal akan mencapai titik energi terendah selama besi itu dipanaskan pada suhu awal yang cukup tinggi dan laju pendinginan atau *cooling rate* yang cukup lambat. Algoritma ini diajukan oleh Metropolis sebagai algoritma yang tidak hanya meningkatkan hasil pada setiap iterasi, namun juga untuk menghindari nilai optimal lokal. Selain itu, proses pendinginan yang dianalogikan dengan pendinginan besi membuat algoritma SA dapat memusatkan hasil secara bertahap hingga menghasilkan penyelesaian global [6].

Algoritma ini menyimulasikan perpindahan acak dengan jangkauan yang kecil yang menghasilkan perubahan energi. Apabila perubahan energi terjadi negatif, maka keadaan energi tersebut lebih kecil dari sebelumnya dan konfigurasi atas keadaan itu menjadi solusi. Apabila perubahan energi tersebut positif, konfigurasi yang membentuknya menghasilkan energi yang lebih besar. Namun keadaan tersebut masih dapat diterima sesuai

dengan probabilitas Boltzman yang dirumuskan dengan Persamaan 2.23

$$P = exp\left(\frac{-\Delta E}{k_b T}\right) = e^{\left(-\Delta E / k_b T\right)} \quad (2.23)$$

b adalah konstanta Boltzman dan T adalah suhu. Suhu (T) pada Persamaan 2.23 ini diinisialisasi dengan nilai T>0 pada saat implementasi kode dan nilainya terus naik sesuai dengan iterasi. Dari Persamaan 2.23 disimpulkan bahwa probabilitas berbanding lurus dengan suhu, apabila suhu menurun maka probabilitas juga mengecil. Hal ini juga menunjukkan bahwa ketika suhu menurun, kebutuhan akan energi membesar sehingga probabilitas diterimanya konfigurasi saat itu menjadi kecil.

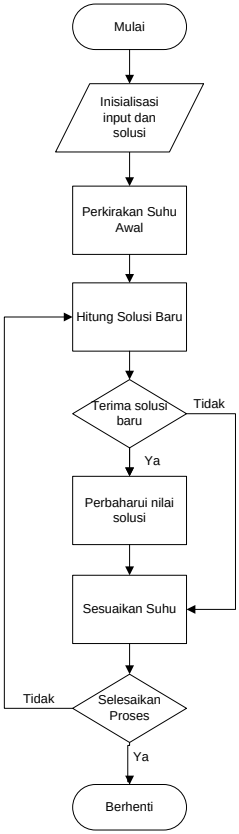
Implementasi dari algoritma Simulated Annealing berlangsung secara berturut-turut seperti digambarkan pada Gambar 2.9. Komponen terpenting dari algoritma Simulated Annealing ini adalah representasi yang dipilih untuk solusi, fungsi objektif, dan operator atau variabel yang menentukan konfigurasi dari proses yang dioptimasi. Fungsi objektif yang dinyatakan sebagai variabel ΔE pada Persamaan 2.23 direpresentasikan sebagai nilai akurasi dari SVM pada sistem gabungan SVM dan SA. Sedangkan nilai operator atau variabel adalah variabel *C* dan σ [6].

2.4.1 Kelebihan SA

Simulated Annealing dapat mengatasi model permasalahan nonlinier yang memiliki tingkat tinggi, data yang kacau, data memiliki banyak *noise*, dan data yang memiliki banyak batasan. Simulated Annealing adalah teknik umum yang dapat digunakan pada banyak permasalahan sehingga fleksibilitas algoritma ini juga menjadi salah satu kelebihan utama. Algoritma ini cukup serba guna dan tidak bergantung pada ketentuan atau model tertentu.

2.4.2 Kekurangan SA

Simulated Annealing adalah salah satu algoritma metaheuristik dan banyak yang harus dilakukan untuk menggunakan algoritma ini. Ada banyak cara untuk mencapai kualitas solusi yang diinginkan. Presisi dari angka-angka yang digunakan dalam algoritma ini dapat menentukan kualitas dari hasil proses. Selain itu menambahkan algoritma ini akan menambah waktu komputasi.



Gambar 2.9 Struktur Algoritma Simulated Annealing.

BAB III
DESAIN DAN PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dibahas mengenai perancangan dan pembuatan sistem perangkat lunak (*software*). Sistem perangkat lunak yang dikembangkan dalam tugas akhir ini adalah implementasi klasifikasi data dengan metode Support Vector Machine dan Simulated Annealing serta pembobotan *kernel*. SVM yang diaplikasikan pada Tugas Akhir ini merupakan aplikasi dari *library* SVM dari Steve R. Gunn untuk Matlab.

Perancangan sistem pada bagian ini meliputi tiga bagian penting yaitu penjelasan data masukan, penjelasan data keluaran, dan algoritma yang digunakan dalam sistem yang juga digambarkan dengan diagram alir.

3.1 Data Masukan

Data masukan pada sistem ini ada dua. *Dataset* Hepatitis dan Breast Cancer yang keduanya didapatkan dari halaman *web* UC Irvine Machine Learning Repository. Selain itu masukan lain yang dibutuhkan adalah jumlah *folds* untuk proses *cross-validation*.

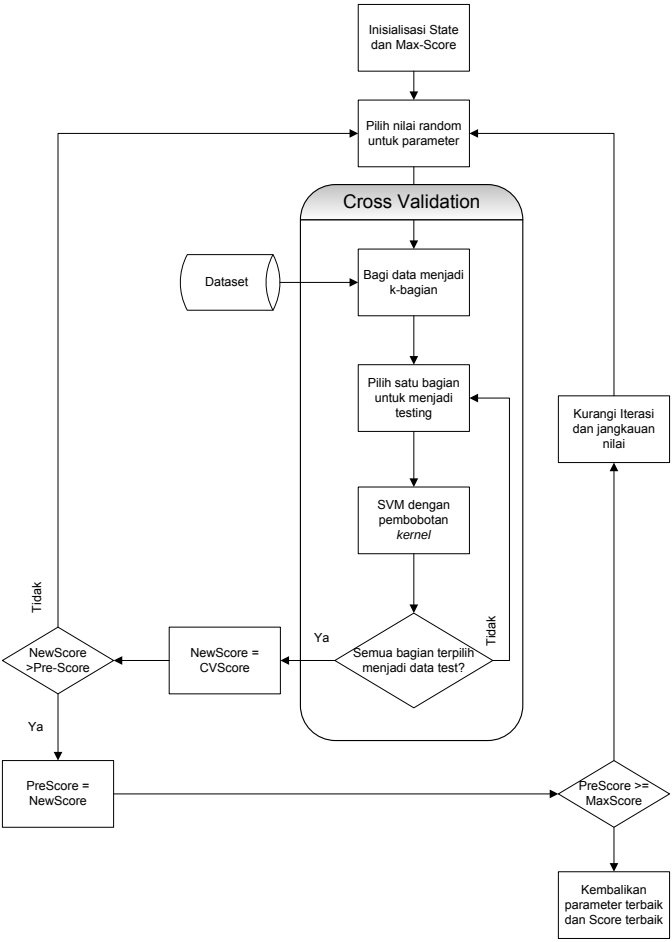
3.2 Data Keluaran

Program ini akan menghasilkan keluaran utama berupa nilai iterasi, nilai parameter C, parameter σ, dan akurasi proses klasifikasi. Keluaran akhir pada sistem ini adalah akurasi terbaik dari proses beserta parameter yang membentuknya.

3.3 Algoritma dan Diagram Alir

3.3.1 Alur Sistem Secara Umum

Sistem menerapkan 3 algoritma yaitu SVM, SA, dan pembobotan *kernel*. Ketiga algoritma tersebut dapat direpresentasikan dalam diagram alir seperti pada Gambar 3.1.

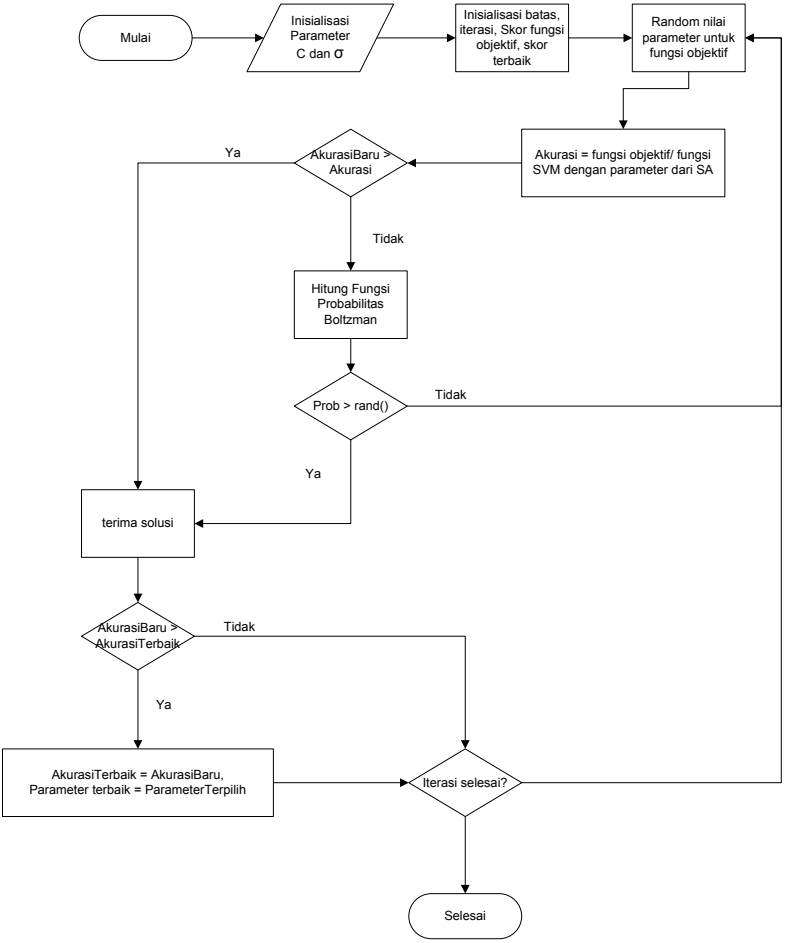


Gambar 3.1 Diagram Alir SVM-SA.

3.3.2 Alur Sistem Pemilihan Parameter dengan Simulated Annealing

Algoritma Simulated Annealing merupakan salah satu metode yang diterapkan untuk mengoptimasi parameter *C* dan σ. Parameter *C* merupakan parameter yang menentukan *upperbound*

dalam pembentukan margin pada SVM dan parameter σ merupakan parameter yang menentukan nilai fungsi *kernel* dalam SVM. Algoritma ini diimplementasikan sesuai dengan Gambar 3.2.



Gambar 3.2 Diagram Alir Algoritma Simulated Annealing.

Sesuai dengan alur yang telah diilustrasikan pada Gambar 3.2, maka kode Simulated Annealing diimplementasikan dengan menggunakan variabel-variabel yang ada pada Tabel 3.1.

Tabel 3.1 Variabel yang Digunakan Pada Implementasi Algoritma Simulated Annealing.

No	Variabel	Tipe	Penjelasan
1.	x_start	double	Matriks yang berisi nilai awal parameter *C* dan σ
2.	N	Int	Jumlah iterasi yang akan dilakukan
3.	Xi	double	Nilai x pada setiap iterasi ke-i
4.	Xc	double	Nilai x terbaik
5.	Fs	double	Skor fungsi objektif setiap iterasi ke-i
6.	Fc	double	Skor terbaik dari fungsi objektif
7.	ObjFunct	double	Hasil fungsi objektif pada iterasi ke-i
8.	P	double	Fungsi probabilitas dari penerimaan solusi

Variabel-variabel pada Tabel 3.1 diimplementasikan dalam bentuk *pseudocode* seperti pada Gambar 3.3 danGambar 3.4 sebelum akhirnya diimplementasikan dalam kode Matlab.

Masukan	Matriks X dengan nilai Parameter C dan σ
Keluaran	Parameter terbaik, skor terbaik

```
1. x_start = [xi(1) xi(2)]
2. n = 10
3. for i = 1:n
4.     Random nilai parameter xi(1), xi(2)
5. fs =ObjFunct(xi(1), xi(2))
```

Gambar 3.3 Pseudocode Implementasi Simulated Annealing (Bagian Pertama).

```
6.  if (fs<fc)
7.         if (rand()<p)
8.            accept = true;
9.  else
10. accept = false;
11.    else
12.         accept = true;
13. if(accept = true)
14.  update fs, xc
```

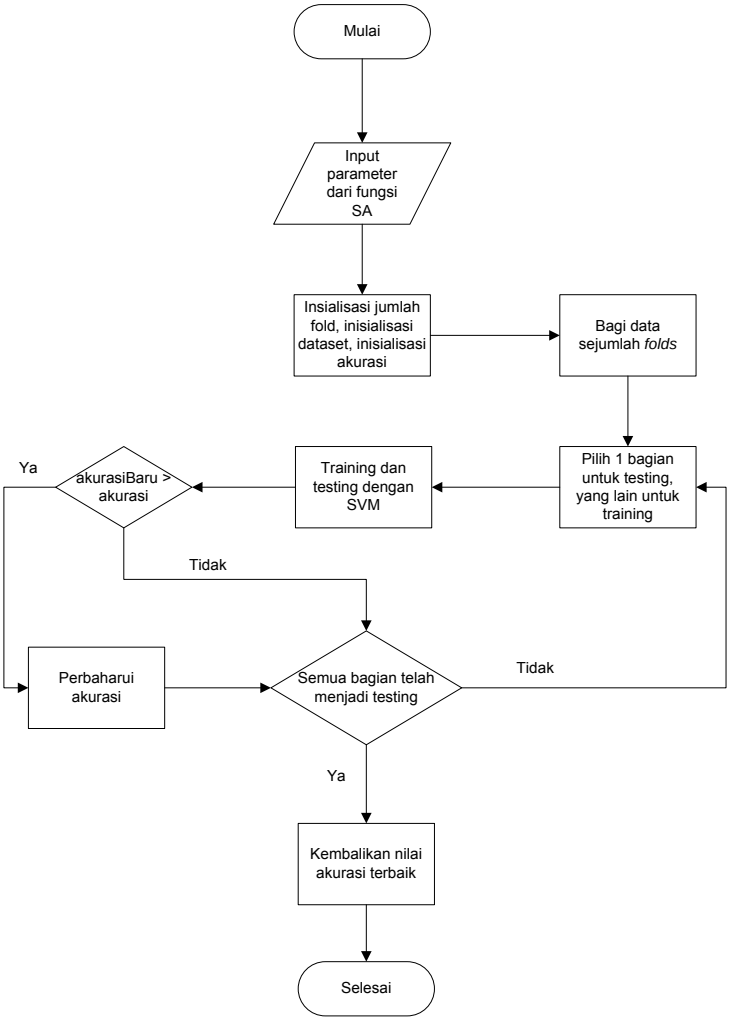
Gambar 3.4 Pseudocode Implementasi Simulated Annealing (Bagian Kedua).

3.3.3 Cross Validation dan SVM

Cross-validation adalah sebuah model untuk membagi *dataset* menjadi bagian-bagian yang independen. Biasanya cara ini digunakan untuk mengatur data untuk mendapatkan prediksi dan mendapatkan akurasi atas prediksi tersebut. Pada kasus tertentu, sering terjadi *overfitting* pada hasil uji coba. Hal ini terjadi karena data pelatihan yang jumlahnya tidak mencukupi, jumlah parameter yang digunakan besar, atau persebaran data setiap *class* tidak seimbang. Dan metode *cross-validation* merupakan salah satu cara yang biasa digunakan untuk mengurangi resiko terjadinya *overfitting* ini.

Cross-validation pada sistem ini dilakukan setelah parameter berhasil didapatkan dari implementasi algoritma SA. *Cross-validation* akan berhubungan langsung dengan implementasi SVM seperti diilustrasikan pada Gambar 3.5

Sesuai dengan diagram alir pada Gambar 3.5, metode *cross-validation* yang digunakan untuk membagi data latih dan data uji. Data latih dan data uji inilah yang nantinya akan diklasifikasikan dengan SVM. *Cross-validation* dapat diimplementasikan dengan variabel-variabel seperti pada Tabel 3.2.



Gambar 3.5 Diagram Alir Implementasi Cross-Validation Terhadap SVM.

Tabel 3.2 Variabel yang Digunakan Pada Implementasi Algoritma Cross-Validation.

No	Variabel	Tipe	Penjelasan
1.	akurasi	Double	Akurasi terbaik dari SVM
2.	C	double	Parameter *C* untuk SVM
3.	p1	double	Parameter σ untuk SVM
4.	K	int	Jumlah *folds*
5.	groups	double	*Dataset* dengan penanda jenis *class*
6.	cvFolds	int	Penanda nomor bagian ke-*k folds*
7.	index	int	Bagian untuk pelatihan pada iterasi ke-*i*
8.	index2	int	Bagian untuk pengujian pada iterasi ke-*i*
9.	trainSet	double	Matriks data pelatihan
10.	trainLabel	double	Matriks label data pelatihan
11.	testSet	double	Matriks data uji
12.	testLabel	double	Matriks label data uji
13.	nsv	double	Nilai kembalian fungsi *svc* berisi *support vector* hasil pelatihan
14.	alpha	double	Nilai kembalian fungsi *svc* berisi nilai parameter alpha hasil pelatihan
15.	b0	double	Nilai kembalian fungsi *svc* berisi nilai parameter bias hasil pelatihan

Variabel-variabel pada Tabel 3.2 diimplementasikan dalam bentuk *pseudocode* seperti pada Gambar 3.6 dan 3.7 sebelum akhirnya diimplementasikan dalam kode Matlab.

`Masukan`	`Parameter Hasil SA dan` *`dataset`*
`Keluaran`	`Akurasi terbaik`

Gambar 3.6 Pseudocode Implementasi Cross-Validation (Bagian Pertama).

```
1. C, p1, k,
2. cvFolds(k)
3. for i = 1:k
1.      index = (groups~=i)
2.      trainSet  = groups(index)
3.      trainLabel = groups(index)
4.      index2 = (groups==i)
5.      testSet = groups(index2)
6.      testLabel = groups(index2)
7. //pelatihan
8. [nsv alpha b0] =
9.           svc(trainSet,trainLabel,C,p1)
10.   //pengujian
11.    akur = svcinfo(svc, testSet, testLabel)
12.    if(akur>akurasi)
13.        akurasi = akur
```

Gambar 3.7 Pseudocode Implementasi Cross-Validation (Bagian Kedua).

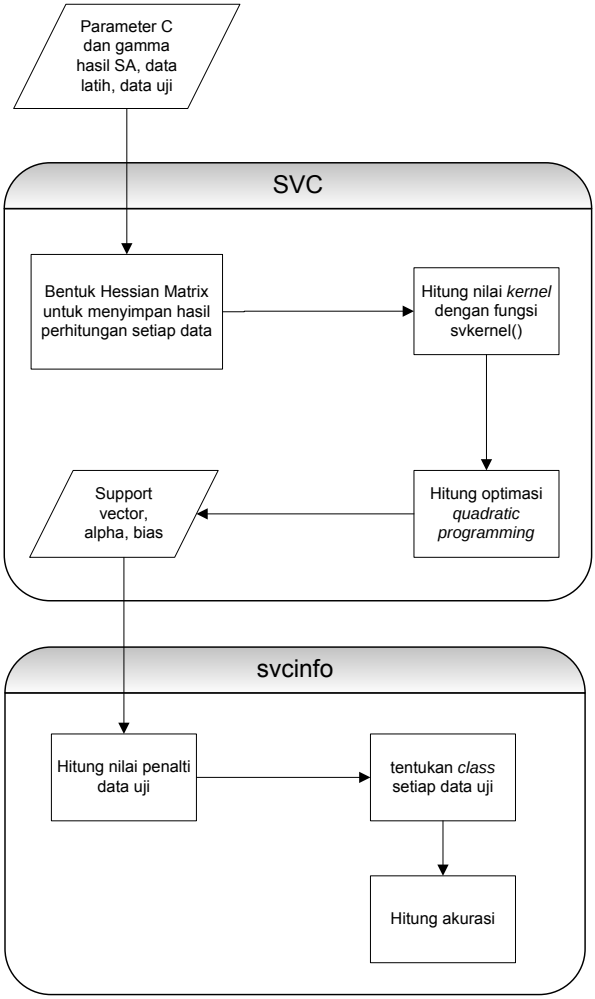
SVM merupakan algoritma yang cukup rumit untuk diimplementasikan. Oleh karena itu, implementasi SVM biasanya terbagi atas beberapa fungsi sesuai dengan kebutuhan fungsional SVM. Pada sistem ini, implementasi SVM dapat diilustrasikan dalam Gambar 3.8. Gambar 3.8 menunjukkan bahwa implementasi SVM terdiri atas banyak fungsi dan saling berkaitan satu sama lain dan setiap fungsi dibutuhkan oleh beberapa kebutuhan seperti pelatihan atau pengujian.

Secara garis besar, implementasi SVM terdiri atas fungsi *svc*, fungsi *svkernel*, dan *svinfo*. Fungsi *svc* digunakan untuk pelatihan, fungsi *svkernel* digunakan untuk menghitung nilai jarak terhadap margin sesuai dengan *kernel* yang digunakan, dan fungsi *svinfo* digunakan untuk pengujian.

3.3.3.1 Fungsi Pelatihan

Proses pelatihan ini dilakukan dengan menggunakan metode SVM nonlinier dengan *kernel* Gaussian ERBF

(Exponential Radiant Basis Function) dan diimplementasikan dalam fungsi *svc*.



Gambar 3.8 Diagram Alir Implementasi SVM.

Langkah-langkah dalam proses pelatihan ini dijelaskan dalam tahap-tahap sebagai berikut:

- Membuat matriks Hessian *H* yang berukuran *n*x*n*.
  Variabel n merupakan jumlah sampel data pelatihan. Matriks Hessian ini terbentuk dengan menerapkan Persamaan (3.1)
  $$H(i,j) = y_i y_j K(x_i, x_j) \quad (3.1)$$
  Variabel y merupakan label yang diberikan pada data latih x.
- Melakukan optimasi *quadratic programming* dengan menggunakan parameter α (Lagrange Multiplier) sesuai dengan Persamaan 2.8 dan 2.9.
- Menghitung nilai margin $||w||^2$
  Fungsi ini membutuhkan data dan beberapa parameter. Data dan parameter ini dapat diimplementasikan sebagai variabel saat masuk proses pengodean. Variabel-variabel ini dijelaskan dalam Tabel 3.3 dan Tabel 3.4.

Tabel 3.3 Variabel yang Digunakan Pada Implementasi Fungsi svc (Bagian Pertama).

No	Variabel	Tipe	Penjelasan
1.	X	double	Matriks masukan data latih
2.	Y	double	Matriks masukan target data latih (*class*)
3.	Ker	String	Jenis *kernel*
4.	C	double	Parameter *upper-bound*
5.	nsv	double	Nilai *support vector* hasil klasifikasi
6.	Alpha	double	Parameter Lagrange Multiplier
7.	b0	double	Parameter bias
8.	H	double	Matriks Hessian untuk menyimpan hasil perhitungan margin setiap data

Tabel 3.4 Variabel yang Digunakan Pada Implementasi Fungsi svc (Bagian Kedua).

9.	Vlb	double	Parameter untuk optimasi *quadratic programming*, kondisi alpha >= 0
10.	Vub	double	Parameter untuk optimasi *quadratic programming*, kondisi alpha <= *C*
11.	w2	double	Nilai margin

Variabel-variabel pada Tabel 3.3 dan Tabel 3.4 diimplementasikan dalam bentuk *pseudocode* seperti Gambar 3.9 sebelum akhirnya diimplementasikan dalam kode Matlab.

Masukan	Data pelatihan dan parameter C hasil SA
Keluaran	*Support vector* (nsv), parameter hasil proses data pelatihan dengan SVM (alpha dan bias)

```
1. Bentuk matriks H(n,n)
2. for i=1:n
3.     for j=1:n
4.        H(i,j)=Y(i)*Y(j)*
5.               svkernel(ker,X(i,:),Y(i,:))
6. vlb = zeros(n,1)
7. vub = C*ones(n,1)
8. [alpha] = qp(H, C, vlb, vub)
9. w2 = alpha'*H*alpha
10.  compute nsv dan b0
```

Gambar 3.9 Pseudocode Implementasi Fungsi svc.

3.3.3.2 Fungsi Kernel Trick

Pada Gambar 3.8, fungsi *kernel trick* ini digunakan saat proses penghitungan nilai jarak terhadap *hyperplane* dengan perhitungan jenis *kernel* tertentu. Fungsi ini diimplementasikan dengan kode fungsi *svkernel*. *Kernel* yang digunakan pada sistem

ini adalah *kernel* ERBF (Exponential Radiant Basis Function) yang dirumuskan dengan Persamaan 3.2.

$$k = exp\left(-\frac{||x-x\prime||_2^2}{2\sigma^2}\right) \quad (3.2)$$

Data dan parameter pada fungsi ini dapat diimplementasikan sebagai variabel saat masuk proses pengodean. Variabel x pada Persamaan 3.2 merupakan data pelatihan dan diimplementasikan sebagai *kernel argument u* sedangkan x' diimplementasikan sebagai *kernel argument v* pada Tabel 3.5. Variabel σ merupakan variabel khusus untuk *kernel* ERBF yang diimplementasikan sebagai *p1* pada Tabel 3.5.

Tabel 3.5 Variabel yang Digunakan Pada Implementasi Fungsi svkernel (Bagian Pertama).

No	Variabel	Tipe	Penjelasan
1.	Ker	String	Jenis *kernel* yang digunakan
2.	U	double	*Kernel* argument, didapatkan dari matriks data pelatihan
3.	V	double	*Kernel* argument, didapatkan dari matriks data pelatihan
4.	p1	double	Parameter *kernel trick*
5.	K	double	Hasil perhitungan *kernel*

Variabel-variabel pada Tabel 3.5 diimplementasikan dalam bentuk *pseudocode* seperti Gambar 3.10 dan Gambar 3.11 sebelum akhirnya diimplementasikan dalam kode Matlab.

`Masukan`	`Matriks data pelatihan dan parameter` σ `hasil SA`
`Keluaran`	`Nilai` k `sebagai hasil perhitungan nilai` *kernel* `terhadap setiap data`

Gambar 3.10 Pseudocode Implementasi Fungsi svkernel (Bagian Pertama).

```
1. //pilih jenis kernel
2. switch 'ker'
3.     //hitung kernel
4.     case 'erbf'
5.     K = exp(-(u-v)*(u-v)'/(2*p1^2));
```

Gambar 3.11 Pseudocode Implementasi Fungsi svkernel (Bagian Pertama).

Persamaan 3.2 diimplementasikan dalam *pseudocode* pada Gambar 3.11 pada baris 5.

3.3.3.3 Fungsi Pengujian

Fungsi pengujian ini dikodekan dengan fungsi *svinfo*. Data dan parameter pada fungsi ini dapat diimplementasikan sebagai variabel saat masuk proses pengodean. Variabel-variabel ini dijelaskan dalam Tabel 3.6 dan Tabel 3.7.

Tabel 3.6 Variabel yang Digunakan Pada Implementasi Fungsi svinfo (Bagian Pertama).

No	Variabel	Tipe	Penjelasan
1.	trnX	double	Matriks masukan data latih
2.	trnY	double	Matriks masukan target data latih (*class*)
3.	tstX	Double	Matriks masukan data uji
4.	tstY	Double	Matriks masukan target data uji (*class*)
5.	Ker	String	Jenis *kernel* yang digunakan
6.	Alpha	Double	Parameter Lagrange Multiplier
7.	Bias	Double	Parameter bias
8.	Correct	Double	Jumlah data yang diklasifikasikan dengan benar, dicocokkan dengan target data uji
9.	Z	Double	Nilai penalti data uji terhadap *support vector*

Tabel 3.7 Variabel yang Digunakan Pada Implementasi Fungsi svinfo (Bagian Kedua).

10.	A	Double	*Class* yang dihasilkan untuk data uji
11.	Akur	Double	Akurasi atas data yang diklasifikasi

Variabel-variabel pada Tabel 3.6 dan Tabel 3.7 diimplementasikan dalam bentuk *pseudocode* seperti Gambar 3.12 sebelum akhirnya diimplementasikan dalam kode Matlab.

`Masukan`	`Matriks data pelatihan dan parameter hasil pelatihan`
`Keluaran`	`Akurasi klasifikasi data`

```
1.  for i=1:(length(tstY))
2.    for j=1:(length(trnY))
3.      //hitung penalty
4.      z = z + trnY(i)*alpha(i)*
5.          svkernel(ker,tstX(j,:),trnX(i,:),S)
6.      //putuskan a dari nilai penalty
7.    if a == tstY(j)
8.  Correct = correct+1
9.  Hitung akur
```

Gambar 3.12 Pseudocode Implementasi Fungsi svinfo (Bagian Pertama).

3.3.4 Pembobotan *Kernel* Gradient Descent

Langkah pertama untuk mencari bobot *S* menggunakan algoritma *Gradient Descent* adalah dengan menghitung nilai Persamaan 3.3 berdasarkan Persamaan 2.20.

$$\frac{\partial \|w\|^2}{\partial S_p} \tag{3.3}$$

Setelah Persamaan 3.3 didapatkan, selanjutnya dilakukan iterasi sampai menemukan nilai *local minimum* S_p menggunakan

Persamaan 2.21. Masukan dari proses ini adalah data latih x_i yang sudah diberi label y_i dan i adalah jumlah sampel data latih.

Setelah proses iterasi berhenti akan didapatkan bobot S yang memenuhi *local minimum* atau yang sudah mencapai batas iterasi yang sudah ditentukan. Keluaran akhir dari proses ini adalah bobot yang sudah dinormalisasi. Variabel–variabel untuk implementasi metode ini dapat dilihat pada Tabel 3.8

Tabel 3.8 Variabel yang Digunakan Pada Implementasi Metode Gradient Descent.

No	Variabel	Tipe	Penjelasan
1.	trainData	double	Matriks masukan data uji
2.	trainLabel	double	Matriks masukan target data uji (*class*)
5.	E	double	Parameter *searching speed*
6.	alpha	double	Parameter Lagrange Multiplier
7.	p1	Double	Parameter *kernel* RBF
8.	Iter	Int	Jumlah iterasi yang akan dilakukan
9.	w2Sp	Double	Nilai bobot yang dihitung dengan rumus pembobotan *gradient descent*
9.	b0	double	Parameter bias
10.	S	Double	Nilai bobot sebagai kembalian fungsi

Variabel-variabel tersebut diimplementasikan dalam *pseudocode* untuk mempermudah pembangunan kode dengan Matlab. *Pseudocode* tersebut dapat dilihat pada Gambar 3.13 dan Gambar 3.14.

`Masukan`	`Matriks data pelatihan dan parameter hasil pelatihan`

Gambar 3.13 Pseudocode Implementasi Metode Gradient Descent (Bagian Pertama).

Keluaran	Nilai bobot yang sudah dinormalisasi
1.	trainData = normalize(trainData)
2.	for iterasi=1:iter
3.	//hitung bobot per data
4.	for i=1:length(trainData)
5.	w2Sp=w2Sp+alpha(i)*alpha(j)*
6.	trainLabel(i)*trainLabel(j).* (-
7.	svkernel('erbfsw',trainData(i,:),
8.	trainData(j,:),S) .*
9.	(S.*double(trainData(i,:)-
10.	trainData(j,:)).^2/p1^2))
11.	End
12.	S = S - e.*w2Sp;

Gambar 3.14 Pseudocode Implementasi Metode Gradient Descent (Bagian Kedua).

BAB IV
IMPLEMENTASI

Setelah pada Bab III dijelaskan mengenai desain dan perancangan perangkat lunak klasifikasi Support Vector Machine (SVM) dengan menggunakan algoritma optimasi Simulated Annealing (SA) serta pembobotan *kernel*, maka pada Bab IV ini akan dijelaskan mengenai tahap implementasi yang meliputi realisasi *pseudocode* berupa kode program yang terdapat dalam perangkat lunak.

4.1 Lingkungan Implementasi

Implementasi dilakukan pada lingkungan sebagai berikut:

- Perangkat keras
  Uji coba dilakukan pada sebuah PC dengan spesifikasi Processor Intel Core i-3 3240 (3.40 GHz, 4.00 GB RAM), 64-bit Operating System.
- Perangkat lunak
  Perangkat lunak ini dikembangkan pada sistem operasi Windows 7 Ultimate dengan menggunakan Matlab 2008a.

4.2 Penjelasan Implementasi

Seperti telah dijelaskan pada bagian desain dan perancangan perangkat lunak, sistem dieksekusi dalam beberapa tahap penting. Setiap tahap mengandung fungsi-fungsi utama. Fungsi-fungsi utama tersebut antara lain:

- Fungsi implementasi Simulated Annealing.
- Fungsi implementasi *cross-validation* yang sekaligus mengandung fungsi pelatihan serta fungsi uji coba atau klasifikasi data dengan menggunakan SVM. Fungsi pada SVM sendiri terdiri atas tiga fungsi penting yaitu fungsi *svc*, *svkernel*, dan *svinfo*.
- Fungsi pembobotan *kernel* dengan metode Gradient Descent.
- Fungsi pembentukan grafik.

4.3 Implementasi Algoritma Simulated Annealing

Algoritma Simulated Annealing membutuhkan masukan jumlah iterasi yang harus dijalankan. Fungsi implementasi algoritma ini merupakan fungsi utama yang nantinya akan dapat menampilkan nilai akhir berupa akurasi setiap iterasi beserta parameternya dan akurasi terbaik secara keseluruhan beserta parameter yang membentuknya pula. Kode program ini dapat dilihat pada Kode Sumber 4.1,Kode Sumber 4.2, 4.3, dan 4.4.

Program ini mengaplikasikan konsep probabilitas Boltzman yang dinotasikan pada Persamaan 2.22 pada baris ke 51. Sesuai dengan alur dari algoritma ini, nilai T tidak akan bernilai 0 dan negatif karena nilai akan terus naik sesuai dengan iterasi.

```
1.  %% Simulated Annealing
2.  Function SA(k, weight)
3.  %Initial Point
4.  x_start= [0.01 0.01];
5.  na = 0.0;% Number of accepted solutions
6.  n = 100;% Jumlah iterasi
7.  % Probability of accepting worse solution at
    the start
8.  p1 = 1;
9.  % Probability of accepting worse solution at
    the end
10. p50 = 0.001;
11. % Initial temperature
12. t1 = -1.0/log(p1);
13. % Final temperature
14. t50 = -1.0/log(p50);
15. % Fractional reduction every cycle
16. frac = (t50/t1)^(1.0/(n-1.0));
```

Kode Sumber 4.1 Program Simulated Annealing (Bagian Pertama).

```
% Initialize x
x = zeros(n+1,2);
x(1,:) = x_start;
xi = x_start;
na = na + 1.0;
% Current best results so far
xc = x(1,:);
fc=crossvalHepatitis(xi(1),xi(2),k,weight);
fs = zeros(n+1,1);
fs(1,:) = fc;
% Current temperature
t = t1;
% DeltaE Average
DeltaE_avg = 0.0;
        fprintf('--------------------------
-----------------------------------------
-----------------\n');
        fprintf('| Iterasi ke |  Cycle |
Function | Nilai C | Nilai Gamma | Folds  |
Best Accuracy  \n');
        fprintf('--------------------------
-----------------------------------------
-----------------------------------------
-\n');
for i=1:n
        % Generate new trial points
        xi(1) = xc(1) + rand();
        xi(2) = xc(2) + rand - 0.5;
% Clip to upper and lower bounds
        xi(1) = max(min(xi(1),32.0),0.1);
        xi(2) = max(min(xi(2),32.0),0.1);
        fprintf('|     %d      |    %d   |
Initial  ', i,j);
```

Kode Sumber 4.2 Program Simulated Annealing (Bagian Kedua).

```
42.         ObjFunct = crossvalHepatitis(xi(1), xi(2),k,weight);
43.         DeltaE = abs(ObjFunct-fc);
44.  if (ObjFunct<fc)
45.  %Initialize DeltaE_avg if a worse solution was foundon the first iteration
46.  if (i==1 && j==1)
47.                  DeltaE_avg = DeltaE;
48.  end
49.  % objective function is worse
50.  % generate probability of acceptance
51.             p = exp(-DeltaE/(DeltaE_avg * t));
52.             % determine whether to accept worse point
53.  if (rand()<p)
54.  % accept the worse solution
55.                  accept = true;
56.  else
57.                  accept = false;
58.                  fprintf('|     %d      | %d   | Solution | Solution NOT accepted because p (%1.1f) < rand\n', i,j', p);
59.  end
60.  if (accept==true)
61.  % update currently accepted solution
62.              xc(1) = xi(1);
63.              xc(2) = xi(2);
64.              fprintf('|     %d      |     %d | Solution ', i,j);
65.              fc = ObjFunct;
66.  % increment number of accepted solutions
67.              na = na + 1.0;
```

Kode Sumber 4.3 Program Simulated Annealing (Bagian Ketiga).

```
68. % update DeltaE_avg
69.             DeltaE_avg = (DeltaE_avg * (na-1.0) +  DeltaE) / na;
70. end
71. % Record the best x values at the end of every cycle
72.     x(i+1,1) = xc(1);
73.     x(i+1,2) = xc(2);
74.     fs(i+1) = fc;
75. % Lower the temperature for next cycle
76.     t = frac * t;
77. end
78. % print solution
79. disp(['Best solution: ',num2str(xc)])
80. disp(['Best objective: ',num2str(fc)])
```

Kode Sumber 4.4 Program Simulated Annealing (Bagian Keempat).

4.4 Implementasi Metode Cross Validation dan SVM

Implementasi dari metode *cross-validation* ini penting karena menentukan pembagian data mana saja yang menjadi data latih dan data uji. Pada fungsi ini pula dilakukan pengenalan *dataset* terhadap sistem. Kode program ini dapat dilihat pada Kode Sumber 4.5 dan 4.6.

```
1. function akurasi = crossvalHepatitis(y,z,k,weight)
2. load hepatitis.data
3. addpath('support vector machines');
4. S = ones(1,19);
5. C = y;
6. global p1;
7. p1= z;
```

Kode Sumber 4.5 Program Cross Validation (Bagian Pertama).

```
8.  bestAcc = 0;
9.  groups = ismember(hepatitis(:,1),1);
10  cvFolds = crossvalind('Kfold', groups, k);
11. RandomSet =[hepatitis cvFolds];
12. fprintf('|    %1.1f    ', C);
13. fprintf('|      %1.1f      ', p1);
14. fprintf('|    %1.1f   ', k);
15. for i= 1:k
16. index = (RandomSet(:,21)~=i);
17. trainSet = RandomSet(index,2:20);
18. trainLabel = RandomSet(index,1);
19. index2 = (RandomSet(:,21)==i);
20. testSet = RandomSet(index2,2:20);
21. testLabel = RandomSet(index2, 1);
22.     [m n] = size(testSet);
23. [nsv, alpha, b0] = svc(trainSet, trainLabel,
    'erbf', C);
24.     akur = svcinfo(trainSet, trainLabel,
    testSet, testLabel, 'erbfsw', alpha, b0, S);
25. if akur>=bestAcc
26.         bestAcc = akur;
27. end
28. end
29. akurasi = bestAcc;
30. fprintf('Best Accuracy = %.2f\n', akurasi)
```

Kode Sumber 4.6 Program Cross Validation (Bagian Kedua).

4.4.1 Implementasi Fungsi *svc*

Seperti telah dijelaskan pada desain dan perancangan perangkat lunak, kode pelatihan yang diimplementasikan dengan fungsi *svc* dieksekusi setelah eksekusi metode *cross-validation.* Implementasi kode *svc* ini dapat dilihat pada Kode Sumber 4.7 dan 4.8.

Program ini mengaplikasikan Persamaan 3.1 pada baris 11 sampai dengan 17. Program ini juga mengaplikasikan

Persamaan 2.8 dan 2.9 dengan mengeksekusi fungsi *qp* pada baris 35.

```
1.  function [nsv, alpha, b0] = svc(X,Y,ker,C)
2.  if (nargin <2 | nargin>4) % check correct
    number of arguments
3.      help svc
4.  else
5.      n = size(X,1);
6.  if (nargin<4) C=Inf;, end
7.  if (nargin<3) ker='linear';, end
8.      S = ones(1,19);
9.  % tolerance for Support Vector Detection
10.     epsilon = svtolI;
11. % Construct the Kernel matrix
12.     H = zeros(n,n);
13. for i=1:n
14. for j=1:n
15.             H(I,j) = Y(i)*Y(j)
    *svkernel(ker,X(I,☺),X(j,☺));
16. end
17. end
18.     c = -ones(n,1);
19. % Add small amount of zero order
    49egularization to avoid problems when
    Hessian is badly conditioned
20.     H = H+1e-10*eye(size(H));
21. % Set up the parameters for the Optimisation
    problem
22. % Set the bounds: alphas >= 0
23.     vlb = zeros(n,1);
24. % Set the bounds:alphas <= C
25.     vub = C*ones(n,1);
```

Kode Sumber 4.7 Program Fungsi svc (Bagian Pertama).

```
26.     x0 = zeros(n,1);
27.     neqcstr = nobias(ker);
28. if neqcstr
29.         A = Y';, b = 0;
30. else
31.         A = [];, b = [];
32. end
33. % Solve the Optimisation Problem
34.     st = cputime;
35.     [alpha lambda how] = qp(H, c, A, b,
    vlb, vub, x0, neqcstr);
36.     w2 = alpha'*H*alpha;
37.     svi = find( alpha > epsilon);
38.     nsv = length(svi);
39.     b0 = 0;
40. if nobias(ker) ~= 0
41. % find b0 from average of support vectors
    on margin
42.       svii = find( alpha > epsilon & alpha
    < (C - epsilon));
43. if length(svii) > 0
44.         b0 =  (1/length(svii))*sum(Y(svii)
    - H(svii,svi)*alpha(svi).*Y(svii));
45. else
46.         fprintf('No support vectors on
    margin - cannot compute bias.\n');
47. end
48 end
49. end
```

Kode Sumber 4.8 Program Fungsi svc (Bagian Kedua).

4.4.2 Implementasi Fungsi *svkernel*

Eksekusi fungsi *svkernel* dilakukan di dalam eksekusi *cross validation* dan dieksekusi setiap perhitungan *kernel* dibutuhkan. Perhitungan *kernel* dilakukan saat pelatihan,

pembobotan, dan pengujian. Kode program fungsi ini dapat dilihat pada Kode Sumber 4.9.

Program ini mengaplikasikan konsep *kernel trick* dengan *kernel* RBF Persamaan 3.2 pada baris 12.

```
1.  function k = svkernel(ker,u,v,S)
2.  if (nargin < 1) % check correct number of
    arguments
3.       help svkernel
4.  else
5.      u = double(u);
6.     v = double(v);
7.  global p1 p2;
8.  switch lower(ker)
9.  case'erbf'
10.          k = exp(-sqrt((u-v)*(u-
    v)')./(2*p1^2));
11. case'erbfsw'
12.         k = exp(-sqrt((S.*(u-v))*(S.*(u-
    v))')/(2*p1^2));
13. end
```

Kode Sumber 4.9 Program Fungsi svkernel.

4.4.3 Implementasi Fungsi *svinfo*

Program fungsi ini selain bergantung pada eksekusi *cross validation*, juga bergantung pada fungsi pembobotan. Nilai S pada fungsi ini menunjukkan bobot yang harus dibebankan pada data uji terhadap data latih dan *hyperplane*. Kode program untuk fungsi ini dapat dilihat pada Kode Sumber 4.10 dan Kode Sumber 4.11. Program ini mengaplikasikan penerapan penentuan nilai hasil klasifikasi sesuai dengan Persamaan 2.17 pada baris 15.

```
1.  function akur = svcinfo(trnX, trnY, tstX,
    tstY, ker, alpha, bias,S)
```

Kode Sumber 4.10 Program Fungsi svinfo (Bagian Pertama).

```
2.  if (nargin ~= 8) % check correct number of arguments
3.      help svcinfo
4.  else
5.      trnX = svdatanorm(trnX,ker);
6.      tstX = svdatanorm(tstX,ker);
7.      epsilon = 1e-10;
8.      n = length(trnY);
9.      m = length(tstY);
10.     correct = n;
11. for j = 1 : m
12.       z = bias;
13. for i = 1 : n
14. if (abs(alpha(i)) > epsilon)
15.             z = z + trnY(i)*alpha(i)*svkernel(ker,tstX(j,:),trnX(i,:),S);
16. end
17. end
18. if (z < 3.285752e-002)
19.             a = 2;
20. else
21.             a = 1;
22. end
23. if (a == tstY(j))
24.             correct = correct + 1;
25. end
26. end
27.     totalPositive = correct;
28.     totalData = n+m;
29.     akur = totalPositive/totalData*100;
30. end
```

Kode Sumber 4.11 Program Fungsi svinfo (Bagian Kedua).

4.5 Implementasi Pembobotan Kernel Gradient Descent

Implementasi pembobotan *kernel* ini dilakukan dalam eksekusi *cross validation* juga namun sifatnya opsional. Metode ini diimplementasikan dalam fungsi Gradient Descent dan kode fungsi ini ditunjukkan pada Kode Sumber 4.12 dan Kode Sumber 4.13. Baris 5 sampai 9 pada Kode Sumber 4.12 merupakan aplikasi dari Persamaan 2.20 dan baris 12 merupakan aplikasi dari Persamaan 2.21.

```
1.  function S = gradientDescent(trainData,trainLabel,alpha,e,p1,iter)
2.      S = ones(1,19);
3.      n = size(trainData,1);
4.      trainData = normalizeX(trainData);
5.  for iterasi=1:iter
6.          w2Sp = zeros(1,19);
7.  for i=1:n
8.  for j=1:n
9.                  w2Sp = w2Sp + alpha(i)*alpha(j)*trainLabel(i)*trainLabel(j) .* (-svkernel('erbfsw',trainData(i,:), trainData(j,:),S) .*(S.*double (trainData(i,:)- trainData(j,:)).^2/p1^2));
10. end
11. end
12.         S = S - e.*w2Sp;
13. end
14. function Xnorm = normalizeX(trainData)
15.     n = size(trainData,2);
16.     Xnorm = double(zeros(size(trainData,1), n));
17. for i=1:n
```

Kode Sumber 4.12 Program Metode Gradient Descent (Bagian Pertama).

```
18.          minX = double(min(trainData(:,i)));
19.          maxX = double(max(trainData(:,i)));
20.          scaleX = double(maxX - minX);
21.  if scaleX
22.              Xnorm(:,i) =
     double(trainData(:,i) - minX) / scaleX;
23.  end
24.  end
```

Kode Sumber 4.13 Program Metode Gradient Descent (Bagian Kedua).

4.6 Implementasi Pembentukan Grafik

Program pembentukan grafik ini terpisah dari program yang lain. Pembentukan grafik dilakukan setelah hasil dari proses SVM-SA dan pembobotan kernel selesai. Pembentukan grafik ini dilakukan untuk mengetahui hubungan antara parameter dan akurasi yang dihasilkan.

Program pembentukan grafik ini ditunjukkan pada Kode Sumber 4.14 dan Kode Sumber 4.15.

```
1.  clc
2.  clear all
3.  data = xlsread('2-10AccBasedWeight.xls');
4.  x = data(:,1);
5.  z = data(:,3);
6.  y = data(:,2);
7.  zbar = mean(z);
8.  xgrid = linspace (-3, 13, N);
9.  ygrid = linspace (-0.5, 1.5, N);
10. [XI,YI] = meshgrid(xgrid,ygrid);
11. ZI = griddata(x,y,z,XI,YI);
12. for I = 1:N;
```

Kode Sumber 4.14 Program Pembentukan Grafik (Bagian Pertama).

```
13.     for J = 1:N;
14.         if isnan(ZI(I,J)) == 1
15.             ZI(I,J) = zbar;
16.         end
17.     end
18. end
19. surf(XI,YI,ZI)
20. grid on;
21. light;
22. lighting phong;
23. camlight('left');
```

Kode Sumber 4.15 Program Pembentukan Grafik (Bagian Kedua).

4.7 Program Inisialisasi

Program ini bertujuan untuk mempermudah pemakaian perangkat lunak. Program ini merupakan representasi fungsi *main* yang dijalankan pada awal uji coba. Program ini ditunjukkan pada Kode Sumber 4.16.

```
1.  function main(name, k, weight)
2.  Data = name;
3.  switch lower(Data)
4.      case 'hepatitis'
5.          SAhepatitis(k, weight);
6.      case 'breastcancer'
7.          SAbreastcancer(k, weight);
8.  end
```

Kode Sumber 4.16 Implementasi Fungsi Inisialisasi.

BAB V
UJI COBA DAN ANALISIS HASIL

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada lingkungan sebagai berikut:

- Perangkat keras
  Uji coba dilakukan pada sebuah PC dengan spesifikasi Processor Intel Core i-3 3240 (3.40 GHz, 4.00 GB RAM), 64-bit Operating System.
- Perangkat lunak
  Perangkat lunak ini dikembangkan pada sistem operasi Windows 7 Ultimate dengan menggunakan Matlab 2008a.

5.2 Skenario Pengujian

Pengujian dilakukan dengan beberapa metode sesuai dengan algoritma yang telah dijelaskan pada bab-bab sebelumnya. Skenario pengujian yang paling utama adalah aplikasi metode Support Vector Machine (SVM) dengan penggabungan algoritma Simulated Annealing (SA) sebagai metode pencarian parameter. Selain penggabungan SVM dengan SA, dilakukan juga pengujian dengan pembobotan *kernel* Gradient Descent.

Pengujian diterapkan pada 2 *dataset* yaitu *dataset* Hepatitis dan Breast Cancer. Jumlah data pada *dataset* Hepatitis adalah 80 setelah dilakukan penghapusan data yang memiliki *missing value. Dataset* Hepatitis memiliki 21 kolom. Atribut pada *dataset* Hepatitis ini dijelaskan dalam Tabel 5.1.

Dataset Breast Cancer juga tidak jauh berbeda dengan *dataset* Hepatitis. Data yang ada pada *dataset* Breast Cancer adalah 683 setelah dilakukan proses eliminasi pada data yang memiliki *missing value. Dataset* Breast Cancer memiliki sebelas kolom dengan satu kolomnya merupakan kolom *class* dan satu kolom lainnya merupakan *code number* dari data. Atribut pada *dataset* Breast Cancer ini dijelaskan pada Tabel 5.2 dan 5.3.

Tabel 5.1 Informasi Atribut Dataset Hepatitis.

Nomor	Atribut	Keterangan
1	Class	Die, Live
2	Age	10, 20, 30, 40, 50, 60, 70, 80
3	Sex	Male, Female
4	Steroid	No, Yes
5	Antivirals	No, Yes
6	Fatique	No, Yes
7	Malaise	No, Yes
8	Anorexia	No, Yes
9	Liver Big	No, Yes
10	Liver Firm	No, Yes
11	Spleen Palpable	No, Yes
12	Spiders	No, Yes
13	Ascites	No, Yes
14	Varices	No, Yes
15	Bilirubin	0.39, 0.80, 1.20, 2.00, 3.00, 4.00
16	Alk Phosphate	33, 80, 120,160, 200, 150
17	Sgot	13, 100, 200, 300, 400, 500
18	Albumin	2.1, 3.0, 3.8, 4.5, 5.0, 6.0
19	Protime	10, 20, 30, 40, 50, 60, 70, 80, 90
20	Histology	No, Yes

Tabel 5.2. Informasi Atribut Dataset Breast Cancer (Bagian Pertama).

Nomor	Atribut	Keterangan
1	Sample Code Number	Nomor *id*
2	Clump Thickness	1 – 10
3	Uniformity of Cell Size	1 – 10
4	Uniformity of Cell Shape	1 – 10
5	Marginal Adhesion	1 – 10
6	Single Ephitelial Cell Size	1 – 10
7	Bare Nuclei	1 – 10
8	Bland Chromatin	1 – 10

Tabel 5.3 Informasi Atribut Dataset Breast Cancer (Bagian Kedua).

9	Normal Nucleoli	1 – 10
10	Mitoses	1 – 10
11	Class	2 untuk Benign, 4 untuk Malignant

5.2.1 Uji Coba Awal Pada *Dataset* Hepatitis

Uji coba awal merupakan skenario untuk mengetahui performa SVM terhadap *dataset* yang digunakan. Uji coba ini dilakukan tanpa menggunakan algoritma SA karena parameter yang dihasilkan telah ditetapkan sebagai acuan untuk percobaan berikutnya. Pada uji coba ini digunakan parameter terbaik yang tertera pada literatur utama. Percobaan ini juga menggunakan 2 jenis *cross validation* yaitu *cross validation* dengan 5 *folds* dan 10 *folds* yang ditunjukkan dalam Tabel 5.4, Tabel 5.5, dan 5.6.

Tabel 5.4 Uji Coba SVM dengan Parameter Awal 5 Folds Cross Validation Dataset Hepatitis.

Validasi ke	Data Latih	Data Uji	Akurasi
1	63	17	96,25
2	64	16	96,25
3	64	16	96,25
4	64	16	97,50
5	65	15	97,50
K = 5; C = 16,6; σ = 0.31342			

Tabel 5.5 Uji Coba SVM dengan Parameter Awal 10 Folds Cross Validation Dataset Hepatitis (Bagian Pertama).

Validasi ke	Data Latih	Data Uji	Akurasi
1	72	8	98,75
2	72	8	98,75
3	72	8	98,75
4	73	7	98,75
5	72	8	98,75

Tabel 5.6 Uji Coba SVM dengan Parameter Awal 10 Folds Cross Validation Dataset Hepatitis (Bagian Kedua).

6	72	8	97,50
7	71	9	97,50
8	72	8	97,50
9	72	8	97,50
10	72	8	97,50
K = 10; C = 16,6; σ = 0.31342			

Hasil dari proses klasifikasi dengan menggunakan SVM menunjukkan bahwa SVM yang digunakan dalam percobaan ini lebih akurat 1.25% daripada SVM yang digunakan dalam literatur utama yang memiliki akurasi 96.25% dengan percobaan 5 *folds cross validation*.

Sedangkan dengan 10 *folds cross validation,* percobaan pada Tugas Akhir ini lebih akurat 2.5%. Percobaan ini juga menunjukkan bahwa jumlah *folds* pada *cross validation* mempengaruhi hasil klasifikasi.

5.2.2 Uji Coba Awal Pada Dataset Breast Cancer

Percobaan kedua dilakukan pada *dataset* Breast Cancer. Percobaan ini dilakukan tanpa mengaplikasikan algoritma SA untuk mengetahui performa awal SVM terhadap *dataset* Breast Cancer. Parameter awal yang digunakan ditentukan oleh pengguna dengan C = 10 dan $\sigma = 0.4$.

Sama dengan percobaan awal pada *dataset* Hepatitis, dilakukan variasi percobaan dengan 2 jenis *cross validation*, 5 *folds* dan 10 *folds.* Hasil untuk percobaan ini ditunjukkan pada Tabel 5.7 untuk hasil uji coba SVM dengan parameter awal 5 *folds cross validation* dan Tabel 5.8 untuk hasil uji coba SVM dengan parameter awal 10 *folds cross validation*.

Tabel 5.7 Uji Coba SVM dengan Parameter Awal 5 Folds Cross Validation.

Validasi Ke	Data Latih	Data Uji	Akurasi
1	546	137	98,98
2	546	137	99,41
3	547	136	99,12
4	547	136	99,27
5	546	137	99,27
K = 5, C = 10, σ = 0,4			

Tabel 5.8 Uji Coba SVM dengan Parameter Awal 10 Folds Cross Validation.

Validasi Ke	Data Latih	Data Uji	Akurasi
1	615	68	99,71
2	615	68	98,98
3	615	68	99,71
4	615	68	99,71
5	615	68	99,56
6	614	69	99,27
7	615	68	99,27
8	615	68	99,41
9	614	69	99,71
10	614	69	99,56
K = 10; C = 10; σ = 0,4			

Hasil untuk percobaan pada *dataset* Breast Cancer menunjukkan bahwa jumlah *folds* pada *cross validation* mempengaruhi hasil klasifikasi. Hasil klasifikasi ini akan dibandingkan dengan skenario percobaan lain untuk melihat performa klasifikasi SVM.

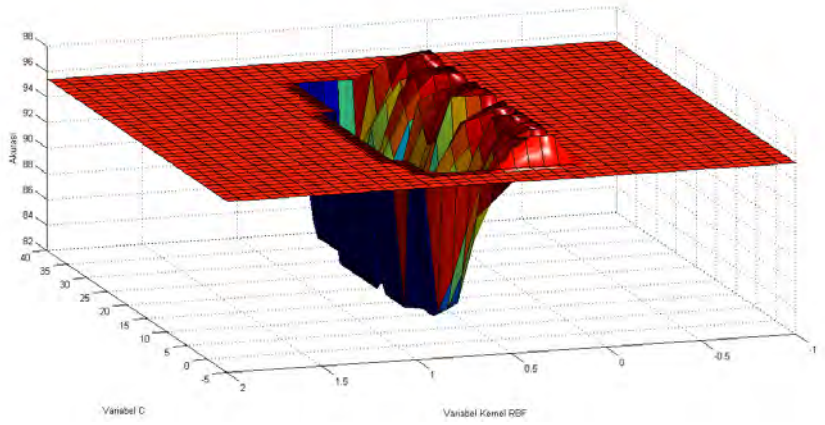
5.2.3 Uji Coba SVM-SA Pada Dataset Hepatitis

Uji coba SVM-SA pada *dataset* Hepatitis dilakukan dengan variasi *cross validation* 5 *folds* dan 10 *folds.* Parameter ditentukan dengan algoritma Simulated Annealing seperti pada

Gambar 3.1. Percobaan dilakukan sebanyak 5 kali dan hasil parameter serta akurasi terbaik ditunjukkan pada Tabel 5.9 untuk 5 dan 10 *folds cross validation*.

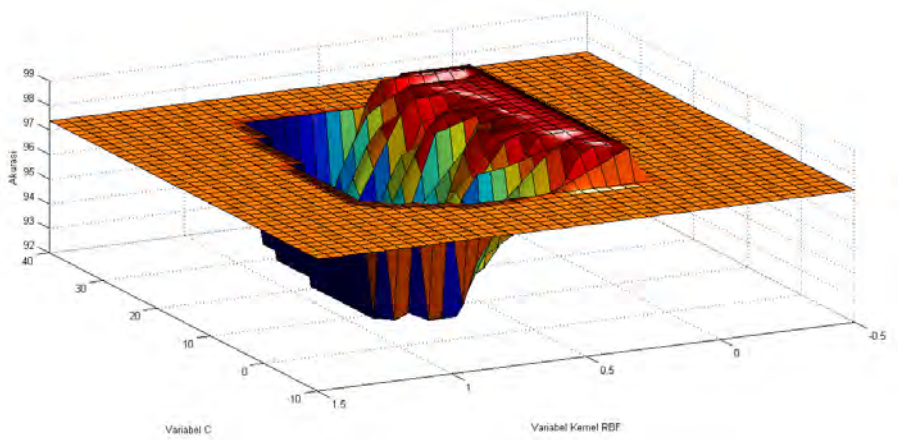
Tabel 5.9 Hasil Uji Coba SVM-SA pada Dataset Hepatitis.

Nomor	C	σ	*Folds*	Akurasi
1	32	0,1179	5	97,5
2	32	0,1	5	97,5
3	32	0,36	5	97,5
4	32	0,1	5	97,5
5	32	0,1467	5	97,5
6	32	0,2392	10	98,75
7	32	0,1	10	98,75
8	32	0,1	10	98,75
9	32	0,1	10	98,75
10	3,2	0,3782	10	98,75



Gambar 5.1 Grafik Hasil Uji Coba SVM-SA 5 Folds Pada Dataset Hepatitis dengan Berbagai Variasi Parameter.

Pada Gambar 5.1 terlihat bahwa kombinasi parameter mempengaruhi akurasi. Tampak bahwa wilayah puncak yang merupakan representasi dari akurasi yang baik berada di wilayah parameter σ yang sempit (antara 0–0,5) untuk berbagai variasi nilai parameter C. Hal ini sesuai dengan hasil akurasi pada Tabel 5.9. Hal ini sama dengan hasil akurasi dengan percobaan 10 *folds* yang diilustrasikan dalam grafik pada Gambar 5.2.



Gambar 5.2 Grafik Hasil Uji Coba SVM-SA 10 Folds Pada Dataset Hepatitis dengan Berbagai Variasi Parameter.

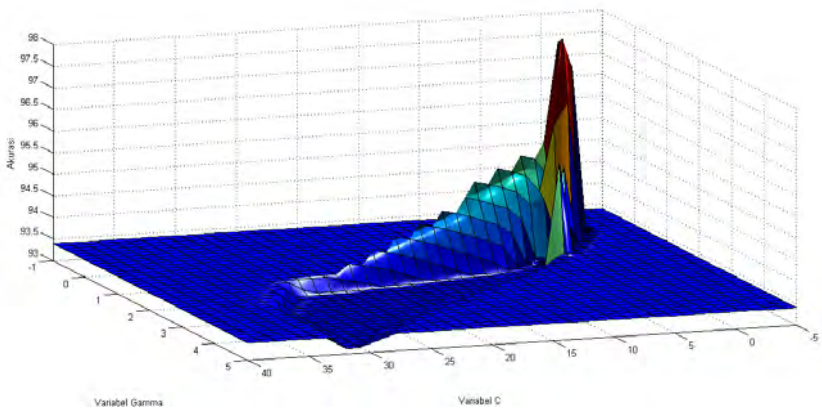
5.2.4 Uji Coba SVM-SA Pada Dataset Breast Cancer

Uji coba SVM-SA pada *dataset* Breast Cancer ini dilakukan sama dengan uji coba SVM-SA pada *dataset* Hepatitis. Parameter ditentukan dengan algoritma Simulated Annealing, menggunakan *5 folds* dan 10 *folds cross validation*, dan menghasilkan akurasi terbaik untuk hasil akhir proses klasifikasi. Percobaan ini dilakukan satu kali untuk masing-masing variasi *folds* dan hasil klasifikasi ditunjukkan pada Tabel 5.10. Satu kali proses SA, akan dilakukan seratus iterasi dan didapatkan satu

akurasi terbaik diantara seratus iterasi tersebut. Berdasarkan konsep ini, grafik seperti pada Gambar 5.3 dan Gambar 5.4 dapat terbentuk. Grafik tersebut merepresentasikan akurasi dari berbagai macam variasi parameter C dan σ. Tabel 5.10 menunjukkan bahwa akurasi maksimum untuk 5 *folds cross validation* sama dengan uji coba awal namun didapatkan dari pasangan parameter yang berbeda. Sementara untuk 10 *folds cross validation*, akurasi yang dihasilkan 0,29% lebih baik daripada percobaan awal.

Tabel 5.10 Akurasi Terbaik Uji Coba SVM-SA pada Dataset Breast Cancer.

Nomor	C	σ	*Folds*	Akurasi
1	0,9	0,4	5	99,41
2	1,5	0,4	10	100

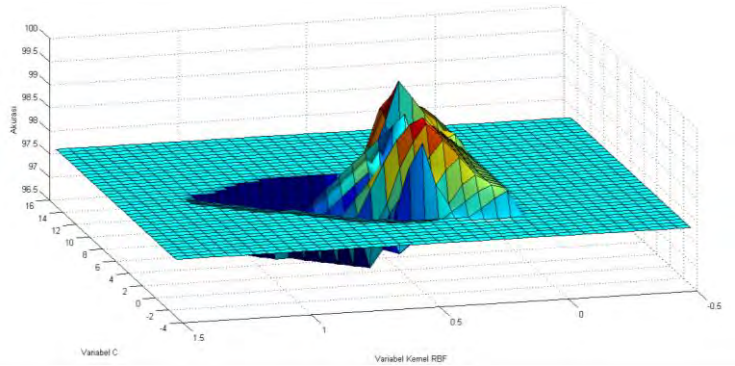


Gambar 5.3 Grafik Hasil Uji Coba SVM-SA 5 Folds Pada Dataset Breast Cancer dengan Berbagai Variasi Parameter.

Pada Gambar 5.3 terlihat bahwa hanya beberapa pasangan parameter saja yang bisa mencapai titik puncak atau akurasi terbaik. Bisa dicocokkan dengan Tabel 5.10 bahwa parameter yang menghasilkan akurasi terbaik berada pada

wilayah parameter C = 0,9 dan parameter σ = 0,4. Selanjutnya akurasi akan menurun seiring dengan bertambahnya nilai parameter baik parameter C maupun parameter σ. Pada uji coba ini terdapat 8 dari 100 pasangan parameter yang menghasilkan akurasi di atas rata-rata.

Pada Gambar 5.4 menunjukkan bahwa akurasi yang tinggi dapat dicapai dengan parameter kernel RBF atau parameter σ antara 0 sampai 0,5 dan parameter C antara 0 sampai 15. Pada uji coba ini, 43 dari 80 pasangan parameter menghasilkan akurasi di atas rata-rata.



Gambar 5.4 Grafik Hasil Uji Coba SVM-SA 10 Folds Pada Dataset Breast Cancer dengan Berbagai Variasi Parameter.

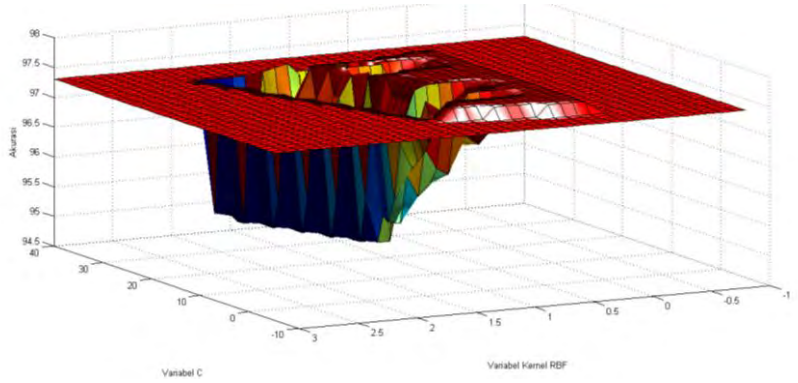
5.2.5 Uji Coba SVM-SA dengan Pembobotan Kernel Gradient Descent Pada Dataset Hepatitis

Uji coba SVM-SA dengan pembobotan Gradient Descent pada *dataset Hepatitis* dilakukan dengan variasi *cross validation* 5 *folds* dan 10 *folds.* Parameter ditentukan dengan algoritma Simulated Annealing. Percobaan dilakukan sebanyak 5 kali dan hasil parameter serta akurasi terbaik ditunjukkan pada Tabel 5.11. Pada Tabel 5.11 akan terlihat bahwa akurasi tertinggi yang dapat dicapai pada uji coba 5 *folds cross validation* masih sebesar 97,50% dan pada uji coba 10 *folds cross validation* sebesar 98,76%.

Tabel 5.11 Uji Coba SVM-SA dengan Pembobotan Gradient Descent Pada Dataset Hepatitis.

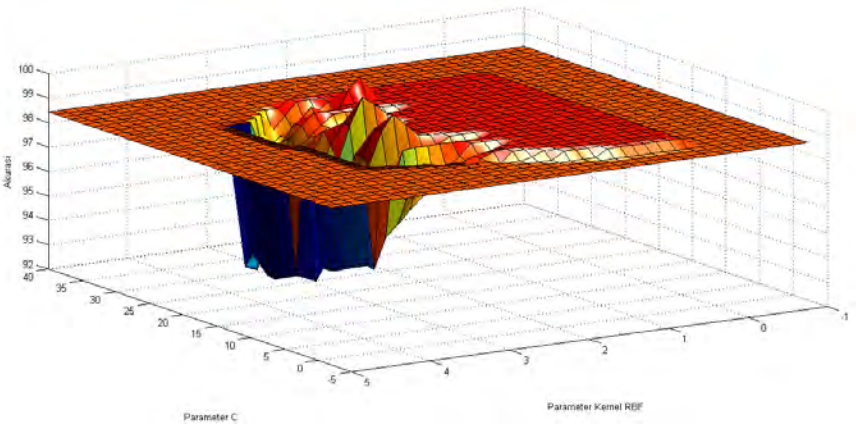
Nomor	C	σ	*Folds*	Akurasi
1	32	1,1989	5	97,50
2	32	0,7975	5	97,50
3	32	0,1	5	97,50
4	32	0,5028	5	97,50
5	32	0,3810	5	97,50
6	31,2915	2,1294	10	100
7	32	2,7103	10	98,75
8	32	0,1	10	98,75
9	32	1,6308	10	98,75
10	32	0,2758	10	98,75

Uji coba kali ini merupakan metode untuk menambahkan bobot pada *kernel* SVM. Hasil percobaan menunjukkan bahwa dengan penentuan parameter menggunakan algoritma Simulated Annealing dan pembobotan Gradient Descent, hasil akurasi menjadi lebih akurat 1.25% daripada literatur utama.



Gambar 5.5 Grafik Hasil Uji Coba SVM-SA dan Pembobotan Kernel 5 Folds Pada Dataset Hepatitis dengan Berbagai Variasi Parameter.

Pada Gambar 5.5 tampak bahwa jangkauan nilai parameter yang bisa menghasilkan akurasi maksimal melebar daripada sebelum pembobotan *kernel* diimplementasikan. Kondisi pada Gambar 5.1 menunjukkan bahwa parameter *kernel* RBF yang dapat menghasilkan akurasi baik berada diantara nilai 0 sampai dengan 0,5. Sementara setelah pembobotan *kernel* dengan metode Gradient Descent diimplementasikan, jangkauan parameter *kernel* RBF yang menghasilkan akurasi maksimum menjadi antara 0 sampai 1,5 dengan berbagai nilai parameter *C*.



Gambar 5.6 Grafik Hasil Uji Coba SVM-SA dan Pembobotan Kernel 10 Folds Pada Dataset Hepatitis dengan Berbagai Variasi Parameter.

Begitu juga dengan uji coba dengan 10 *folds cross validation*. Tidak hanya meningkatkan akurasi, jangkauan nilai yang menghasilkan akurasi maksimum juga melebar. Pada Gambar 5.2 pada kondisi pembobotan *kernel* belum diimplementasikan terlihat bahwa parameter *kernel* RBF yang dibutuhkan untuk menghasilkan akurasi maksimum berada antara 0 sampai 0,5. Sedangkan pada Gambar 5.6 dengan kondisi

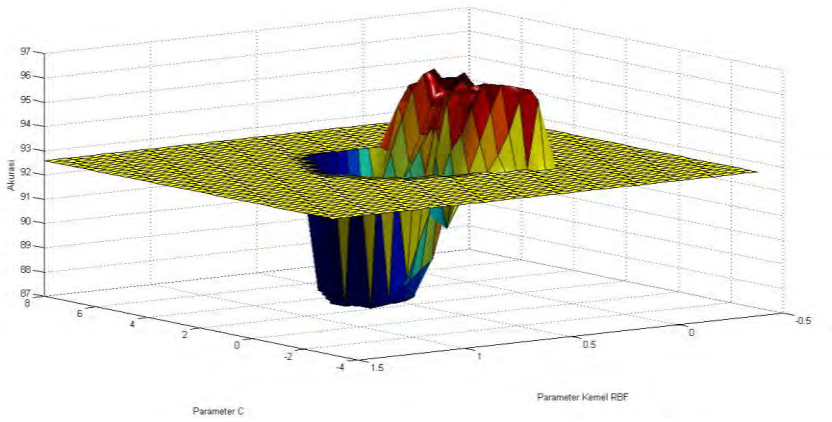
pembobotan *kernel* telah diimplementasikan, jangkauan tersebut melebar sampai nilai 3 dan diantaranya mencapai akurasi terbaik.

5.2.6 Uji Coba SVM-SA dengan Pembobotan *Kernel* Gradient Descent Pada *Dataset* Breast Cancer

Pada data Breast Cancer, dilakukan percobaan yang sama dengan menerapkan algoritma SVM dengan optimasi parameter SA dan pembobotan *kernel* Gradient Descent. Uji coba ini dilakukan satu kali dengan variasi *cross validation* 5 *folds* dan 10 *folds*. Hasil klasifikasi ditunjukkan dengan akurasi terbaik dan parameter yang membentuknya dapat dilihat pada Tabel 5.12

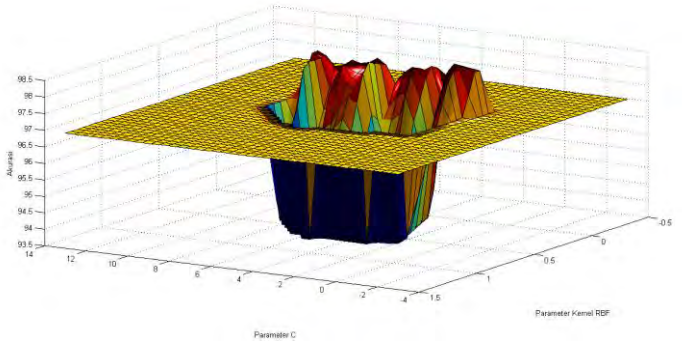
Tabel 5.12 Uji Coba SVM-SA dengan Pembobotan *Kernel* Gradient Descent Pada *Dataset* Breast Cancer

No	C	σ	*Folds*	Akurasi
1	4,2	0,1	5	96,49
2	7,921	0,3544	10	98,54



Gambar 5.7 Grafik Hasil Uji Coba SVM-SA dengan Pembobotan Kernel 5 Folds Pada Dataset Breast Cancer dengan Berbagai Pasangan Parameter.

Seperti percobaan pada *dataset* Hepatitis, terlihat pada Gambar 5.7 bahwa jangkauan dari pasangan parameter yang dapat menghasilkan akurasi tinggi terletak pada jangkauan parameter σ antara 0 sampai dengan 0,5 dan parameter C antara 0 sampai dengan 6. Pasangan parameter yang dihasilkan memang lebih terbatas daripada uji coba tanpa pembobotan *kernel*. Namun uji coba ini membuktikan bahwa dengan parameter yang terbatas dapat menghasilkan akurasi yang tinggi. Hal ini dibuktikan dengan 56 dari 85 data yang dihasilkan mencapai akurasi di atas rata-rata.



Gambar 5.8 Grafik Hasil Uji Coba SVM-SA dengan Pembobotan Kernel 10 Folds Pada Dataset Breast Cancer dengan Berbagai Pasangan Parameter.

Begitu juga pada Gambar 5.8. Terlihat bahwa akurasi tinggi dapat dicapai dengan jangkauan parameter σ antara 0 sampai dengan 0,5 dan parameter C antara 0 sampai 12. Serupa dengan uji coba SVM-SA dengan pembobotan *kernel* 5 *folds*, pasangan parameter yang menghasilkan akurasi di atas rata-rata akan lebih merata daripada tanpa menggunakan pembobotan *kernel*. Hal ini ditunjukkan dengan 74 dari 100 data menghasilkan akurasi di atas rata-rata.

5.3 Analisis Hasil Uji Coba

Uji coba dilakukan sebanyak enam skenario dan dilakukan terhadap dua *dataset*. Pada enam skenario ini didapatkan beberapa kesimpulan atas hasil percobaan.

Secara umum, SVM yang digunakan pada percobaan Tugas Akhir ini lebih akurat 1,25% daripada SVM yang digunakan pada literatur utama yang menjadi acuan percobaan Tugas Akhir ini. Penggunaan algoritma Simulated Annealing membantu penemuan pasangan parameter C dan σ sehingga menghasilkan akurasi yang tinggi. Pada uji coba SVM-SA pada *dataset* Hepatitis terlihat bahwa akurasi tertinggi 97,50% dapat dicapai sejumlah pasangan parameter. Dan penerapan pembobotan *kernel* pada proses akan memperlebar jangkauan pasangan parameter hingga 2 kali jangkauan parameter pada uji coba tanpa pembobotan.

Sedangkan pada *dataset* Breast Cancer, hasil uji coba tidak jauh berbeda dari uji coba pada dataset Hepatitis. Algoritma Simulated Annealing menemukan pasangan-pasangan yang dapat menghasilkan akurasi terbaik dan pembobotan *kernel* dapat memperlebar jangkauan parameter hingga 2 kali jangkauan parameter tanpa pembobotan. Jangkauan nilai parameter ini penting agar sistem tidak mudah terjebak pada parameter-parameter yang akan menghasilkan akurasi rendah.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas kesimpulan yang dapat diambil dari hasil pembuatan perangkat lunak dan hasil uji cobanya. Selain itu, terdapat saran untuk pengembangan perangkat lunak lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Kinerja Support Vector Machine (SVM) dipengaruhi oleh banyak parameter. Algoritma Simulated Annealing (SA) dapat diimplementasikan untuk mengoptimasi parameter SVM
2. Kinerja SVM juga ditentukan oleh *kernel* yang digunakan Metode pembobotan *kernel* Gradient Descent dapat diaplikasikan pada SVM untuk memperbaiki hasil klasifikasi.
3. Akurasi yang dihasilkan sistem SVM dengan optimasi parameter menggunakan SA lebih akurat hingga 1,25% lebih baik daripada tanpa optimasi parameter dengan menggunakan SA.
4. Jangkauan nilai parameter dapat diperluas hingga 2 kali lipat dengan menerapkan metode pembobotan *kernel*. Metode pembobotan *kernel* Gradient Descent akan membantu menghasilkan akurasi yang baik pada jangkauan nilai parameter yang luas.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan aplikasi ini antara lain:

1. Perangkat lunak ini akan membutuhkan waktu yang lebih lama karena menerapkan beberapa metode dalam satu kali proses. Pengembangan yang dapat dilakukan adalah memperbaiki sistem hingga dapat menghasilkan akurasi yang

baik dengan waktu yang lebih singkat.

2. Metode pembelajaran SA-SVM pada Tugas Akhir ini menghasilkan akurasi yang dipengaruhi oleh banyak parameter. Oleh karena itu perlu dilakukan perancangan percobaan secara ortogonal untuk hasil yang lebih maksimal.

DAFTAR PUSTAKA

[1] J. S. Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, 1st Edition ed., Cambrige: Cambrige Univeristy Press, 2004.

[2] A. S. Nugroho, A. B. Witarto and D. Handoko, "Support Vector Machine - Teori dan Applikasinya dalam Bioinformatika," 2003. [Online]. Available: http://www.ilmukomputer.com.

[3] J. S. Sartakhti, M. H. Zangooei and K. Mozafari, "Hepatitis Disease Diagnosis Using A Novel Hybrid Method Based On Support Vector Machine and Simulated Annealing (SVM-SA)," *Computer Methods And Programs In Biomedicine*, vol. I, no. 08, pp. 570-579, August 2011.

[4] S. Theodoridis and K. Koutroumbas, Pattern Recognition, 4th Edition ed., Elsevier, 2003.

[5] B. Guo, S. R. Gunn, R. Damper and J. D. B. Nelson, "Customizing Kernel Function for SVM-Based Hyperspectral Image Classification," *IEEE Transactions On Image Processing*, vol. 17, pp. 622-629, April 2008.

[6] F. Busetti, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/.

[7] W. L. Shih, J. L. Zne, C. C. Shih and Y. T. Tsung, "Parameter Determination of Support Vector Machine and Feature Selection Using Simulated Annealing Approach," *Applied Soft Computing*, vol. 8, pp. 1505-1512, October 2007.

BIODATA PENULIS

Astris Dyah Perwita, biasa dipanggil Astris, lahir di Surabaya pada tanggal 25 September 1991, merupakan anak kedua dari dua bersaudara. Penulis telah menempuh pendidikan mulai dari MIN Malang I (1998-2004), SMP Negeri 3 Malang (2004-2007), SMA Negeri 3 Malang (2007-2010), dan pada tahun 2010 penulis meneruskan pendidikannya di Teknik Informatika ITS.

Penulis aktif di organisasi Himpunan Mahasiswa Teknik Computer - Informatika (HMTC) ITS. Penulis juga aktif dalam Unit Kegiatan Mahasiswa Tari dan Karawitan Rara Kananta sebagai koodinator tari tradisional.

Dalam perkuliahan, penulis mengambil bidang minat Komputasi Cerdas dan Visualisasi (KCV) dan tertarik pada hal yang berhubungan dengan kecerdasan buatan dan *machine learning*.

Penulis dapat dihubungi melalui email di astrisdp@gmail.com.

LAMPIRAN

Pada bagian lampiran ini berisi hasil-hasil percobaan secara keseluruhan. Tabel A.22 sampai Tabel A.7 berisi hasil percobaan SVM, SA, dan tanpa pembobotan kernel Gradient Descent 5 *folds cross validation*. Percobaan ini dilakukan 5 kali kemudian hasil dari kelima percobaan ini digabungkan dan hasil yang serupa dieliminasi sehingga tidak ada data yang identik.

Tabel A.1 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	0.1	0.1	97.5	19	10.2	0.5	92.5
2	0.1	0.4	97.5	20	10.2	0.1	97.5
3	0.3	0.1	97.5	21	10.3	0.1	97.5
4	0.5	0.8	83.75	22	10.4	0.4	97.5
5	0.6	0.5	93.75	23	10.5	0.1	97.5
6	0.7	0.1	97.5	24	10.6	0.2	97.5
7	0.8	0.1	97.5	25	10.6	0.1	97.5
8	0.8	0.4	96.25	26	10.7	0.9	83.75
9	1	0.1	97.5	27	10.8	0.1	97.5
10	1.1	0.6	86.25	28	10.8	0.2	97.5
11	1.4	0.6	83.75	29	10.9	0.5	90
12	1.4	0.1	97.5	30	11	0.1	97.5
13	1.6	0.5	93.75	31	11.1	0.1	97.5
14	1.6	0.1	97.5	32	11.3	0.9	83.75
15	1.6	0.6	86.25	33	11.4	0.4	97.5
16	1.8	0.1	97.5	34	11.4	0.1	97.5
17	1.9	0.1	97.5	35	11.5	0.1	97.5
18	10	0.1	97.5	36	11.6	0.2	97.5

Tabel A.2 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
39	12.2	0.1	97.5	65	16	0.2	97.5
40	12.3	0.1	97.5	66	16	0.5	95
41	12.3	0.4	96.25	67	16.1	0.6	85
42	12.6	0.1	97.5	68	16.1	0.3	97.5
43	12.8	0.1	97.5	69	16.2	0.1	97.5
44	12.8	0.2	97.5	70	16.4	0.1	97.5
45	13.2	0.4	96.25	71	16.5	0.2	97.5
46	13.4	0.2	97.5	72	16.6	0.1	97.5
47	13.4	0.1	97.5	73	16.8	0.1	97.5
48	13.5	0.1	97.5	74	16.8	0.4	96.25
49	13.7	0.1	97.5	75	17	0.4	96.25
50	13.8	0.1	97.5	76	17.1	0.3	97.5
51	14.2	0.2	97.5	77	17.1	0.4	96.25
52	14.3	0.2	97.5	78	17.1	0.1	97.5
53	14.3	0.1	97.5	79	17.2	0.7	83.75
54	14.3	0.5	93.75	80	17.2	0.4	96.25
55	14.6	0.1	97.5	81	17.3	0.1	97.5
56	14.7	0.7	83.75	82	17.3	0.6	86.25
57	14.8	0.1	97.5	83	17.4	0.1	97.5
58	15	0.4	96.25	84	17.4	0.2	97.5
59	15.1	0.1	97.5	85	17.6	0.4	97.5
60	15.1	0.4	95	86	17.6	0.1	97.5
61	15.3	0.1	97.5	87	17.8	0.3	97.5
62	15.6	0.4	97.5	88	17.9	0.1	97.5
63	15.7	0.1	97.5	89	17.9	0.4	96.25
64	15.9	0.1	97.5	90	18	0.4	96.25

Tabel A.3 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Ketiga).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
91	18.2	0.3	97.5	117	2.9	0.1	97.5
92	18.4	0.1	97.5	118	20	0.5	91.25
93	18.4	0.3	97.5	119	20	0.1	97.5
94	18.5	0.1	97.5	120	20.2	0.1	97.5
95	18.6	0.1	97.5	121	20.2	0.5	95
96	18.7	0.5	96.25	122	20.4	0.2	97.5
97	18.9	0.6	86.25	123	20.4	0.1	97.5
98	19	0.1	97.5	124	20.5	0.1	97.5
99	19	0.5	92.5	125	20.6	0.1	97.5
100	19.1	0.1	97.5	126	20.7	0.1	97.5
101	19.4	0.5	96.25	127	20.9	0.1	97.5
102	19.4	0.4	96.25	128	20.9	0.3	97.5
103	19.4	0.1	97.5	129	20.9	0.1	97.5
104	19.5	0.5	93.75	130	21	0.1	97.5
105	19.6	0.1	97.5	131	21.4	0.1	97.5
106	19.8	0.1	97.5	132	21.5	0.1	97.5
107	19.8	0.4	96.25	133	21.6	0.1	97.5
108	19.9	0.1	97.5	134	21.7	0.6	83.75
109	2	0.1	97.5	135	21.7	0.1	97.5
110	2.1	0.1	97.5	136	21.9	0.1	97.5
111	2.2	0.6	86.25	137	22.1	0.2	97.5
112	2.3	0.3	97.5	138	22.1	0.1	97.5
113	2.3	0.1	97.5	139	22.4	0.3	97.5
114	2.7	0.6	86.25	140	22.5	0.4	97.5
115	2.8	0.6	85	141	22.6	0.1	97.5
116	2.8	0.3	97.5	142	22.6	0.2	97.5

Tabel A.4 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Keempat).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
143	22.6	0.5	92.5	169	25.7	0.5	92.5
144	22.7	0.5	93.75	170	25.7	0.1	97.5
145	22.7	0.1	97.5	171	26	0.1	97.5
146	23	0.1	97.5	172	26.2	0.4	97.5
147	23.1	0.3	97.5	173	26.4	0.6	85
148	23.1	0.2	97.5	174	26.4	0.1	97.5
149	23.2	0.5	92.5	175	26.5	0.1	97.5
150	23.3	0.8	83.75	176	26.6	0.1	97.5
151	23.4	0.5	92.5	177	26.7	0.3	97.5
152	23.4	0.1	97.5	178	26.8	0.3	97.5
153	23.5	0.1	97.5	179	26.9	0.2	97.5
154	23.6	0.5	96.25	180	26.9	0.1	97.5
155	23.7	0.2	97.5	181	26.9	0.5	93.75
156	23.7	0.3	97.5	182	27.1	0.5	93.75
157	24	0.1	97.5	183	27.1	0.4	93.75
158	24.2	0.1	97.5	184	27.4	0.1	97.5
159	24.4	0.4	97.5	185	27.5	0.1	97.5
160	24.4	0.3	97.5	186	27.5	0.6	86.25
161	24.7	0.5	96.25	187	27.7	0.2	97.5
162	25	0.4	97.5	188	27.7	0.4	97.5
163	25	0.3	96.25	189	27.9	0.1	97.5
164	25.1	0.3	97.5	190	27.9	0.2	97.5
165	25.1	0.1	97.5	191	28	0.4	97.5
166	25.2	0.1	97.5	192	28	0.1	97.5
167	25.4	0.1	97.5	193	28.1	0.4	97.5
168	25.4	0.4	97.5	194	28.2	0.1	97.5

Tabel A.5 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Kelima).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
195	28.2	0.5	90	221	30.1	0.1	97.5
196	28.3	0.5	96.25	222	30.4	0.1	97.5
197	28.5	0.1	97.5	223	30.5	0.1	97.5
198	28.7	0.5	92.5	224	30.7	0.3	97.5
199	28.7	0.3	97.5	225	30.7	0.6	86.25
200	28.8	0.3	96.25	226	30.7	0.1	97.5
201	28.9	0.1	97.5	227	30.9	0.3	97.5
202	29	0.7	83.75	228	30.9	0.6	85
203	29	0.1	97.5	229	30.9	0.1	97.5
204	29.2	0.2	97.5	230	30.9	0.5	92.5
205	29.2	0.1	97.5	231	31	0.3	97.5
206	29.5	0.2	97.5	232	31.2	0.2	97.5
207	29.6	0.1	97.5	233	31.3	0.4	97.5
208	29.7	0.1	97.5	234	31.4	0.2	97.5
209	29.9	0.5	92.5	235	31.4	0.1	97.5
210	29.9	0.1	97.5	236	31.5	0.1	97.5
211	29.9	0.4	96.25	237	31.6	0.1	97.5
212	3.1	0.2	97.5	238	31.6	0.4	96.25
213	3.1	0.3	97.5	239	31.8	0.4	96.25
214	3.3	0.2	97.5	240	31.8	0.2	97.5
215	3.3	0.4	97.5	241	31.9	0.6	86.25
216	3.6	0.1	97.5	242	31.9	0.1	97.5
217	3.7	0.5	91.25	243	32	0.1	97.5
218	3.8	0.6	83.75	244	32	0.2	97.5
219	3.9	0.1	97.5	245	32	0.3	97.5
220	30	0.1	97.5	246	32	0.4	97.5

Tabel A.6 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Keenam).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
247	32	0.5	96.25	273	6.3	0.3	97.5
248	32	0.6	91.25	274	6.3	0.7	83.75
249	32	0.8	83.75	275	6.3	0.1	97.5
250	4	0.1	97.5	276	6.4	0.4	96.25
251	4	0.2	97.5	277	6.7	0.1	97.5
252	4.3	0.3	97.5	278	7	0.1	97.5
253	4.3	0.1	97.5	279	7	0.2	97.5
254	4.4	0.2	97.5	280	7.1	0.1	97.5
255	4.5	0.2	97.5	281	7.4	0.4	96.25
256	4.6	0.6	87.5	282	7.4	0.1	97.5
257	4.8	0.6	83.75	283	7.5	0.4	96.25
258	4.9	0.1	97.5	284	7.6	0.2	97.5
259	5.1	0.1	97.5	285	7.6	0.5	90
260	5.2	0.1	97.5	286	7.6	0.1	97.5
261	5.3	0.2	97.5	287	7.8	0.6	87.5
262	5.3	0.4	97.5	288	7.9	0.1	97.5
263	5.5	0.1	97.5	289	8	0.4	97.5
264	5.7	0.6	86.25	290	8	0.1	97.5
265	5.9	0.2	97.5	291	8.2	0.5	95
266	5.9	0.4	97.5	292	8.2	0.1	97.5
267	5.9	0.1	97.5	293	8.3	0.1	97.5
268	6	0.5	92.5	294	8.5	0.2	97.5
269	6	0.7	83.75	295	8.7	0.3	97.5
270	6	0.1	97.5	296	8.7	0.7	83.75
271	6.1	0.2	97.5	297	8.7	0.6	87.5
272	6.2	0.1	97.5	298	8.8	0.6	86.25

Tabel A.7 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Ketujuh).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
299	8.8	0.3	97.5	308	9.3	0.3	97.5
300	8.8	0.7	83.75	309	9.3	0.4	97.5
301	8.9	0.6	88.75	310	9.5	0.4	96.25
302	8.9	0.1	97.5	311	9.6	0.6	83.75
303	9	0.6	86.25	312	9.6	0.8	83.75
304	9	0.8	83.75	313	9.6	0.1	97.5
305	9	0.5	95	314	9.6	0.4	97.5
306	9	0.6	83.75	315	9.9	0.2	97.5
307	9.2	0.4	96.25				

Pada bagian lampiran ini berisi hasil-hasil percobaan secara keseluruhan. Tabel A.8 sampai Tabel A.16 berisi hasil percobaan SVM, SA, dan tanpa pembobotan kernel Gradient Descent 10 *folds cross validation*. Pada awalnya percobaan ini dilakukan 5 kali kemudian hasil dari kelima percobaan ini digabungkan dan hasil yang serupa dieliminasi sehingga tidak ada data yang identik.

Tabel A.8 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	0.2	0.3	98.75	7	1	0.1	98.75
2	0.7	0.4	98.75	8	1.1	0.5	98.75
3	0.7	0.1	98.75	9	1.1	0.6	92.5
4	0.7	0.2	98.75	10	1.3	0.3	98.75
5	0.9	0.4	98.75	11	1.3	0.5	98.75
6	1	0.3	98.75	12	1.4	0.6	96.25

Tabel A.9 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
13	1.4	0.1	98.75	39	4	0.1	98.75
14	1.5	0.1	98.75	40	4	0.9	92.5
15	1.7	0.8	92.5	41	4	0.3	98.75
16	1.7	0.1	98.75	42	4.1	0.1	98.75
17	2	0.1	98.75	43	4.1	0.2	98.75
18	2	0.4	98.75	44	4.2	0.2	98.75
19	2	0.5	97.5	45	4.3	0.2	98.75
20	2	0.8	92.5	46	4.3	0.1	98.75
21	2.1	0.1	98.75	47	4.4	0.9	92.5
22	2.1	0.4	98.75	48	4.5	1	92.5
23	2.2	0.5	98.75	49	4.5	0.5	98.75
24	2.3	0.1	98.75	50	4.5	0.2	98.75
25	2.4	0.1	98.75	51	4.9	0.1	98.75
26	2.6	0.2	98.75	52	5	0.1	98.75
27	3	0.1	98.75	53	5.1	0.1	98.75
28	3	0.3	98.75	54	5.2	0.1	98.75
29	3.1	0.1	98.75	55	5.3	0.1	98.75
30	3.2	0.7	92.5	56	5.4	0.1	98.75
31	3.2	0.1	98.75	57	5.4	0.6	93.75
32	3.3	0.1	98.75	58	5.5	0.4	98.75
33	3.5	0.5	98.75	59	5.7	0.3	98.75
34	3.5	0.6	92.5	60	5.7	0.2	98.75
35	3.6	0.1	98.75	61	5.9	0.6	96.25
36	3.7	0.1	98.75	62	5.9	0.1	98.75
37	3.7	0.5	98.75	63	5.9	0.3	98.75
38	3.8	0.9	92.5	64	6	0.1	98.75

Tabel A.10 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Ketiga).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
65	6	0.3	98.75	91	8.2	0.3	98.75
66	6.1	0.1	98.75	92	8.2	0.6	96.25
67	6.1	0.6	93.75	93	8.2	0.5	98.75
68	6.2	0.8	92.5	94	8.3	0.5	98.75
69	6.2	0.1	98.75	95	8.5	0.8	92.5
70	6.2	0.2	98.75	96	8.5	0.7	92.5
71	6.3	0.7	92.5	97	8.7	0.3	98.75
72	6.5	0.3	98.75	98	8.7	0.1	98.75
73	6.6	0.6	97.5	99	8.8	0.7	92.5
74	6.7	0.1	98.75	100	8.9	0.2	98.75
75	6.8	0.1	98.75	101	8.9	0.1	98.75
76	6.9	0.1	98.75	102	9	0.1	98.75
77	7	0.1	98.75	103	9	0.7	92.5
78	7.2	0.5	98.75	104	9	0.9	92.5
79	7.3	0.1	98.75	105	9.1	0.8	92.5
80	7.4	0.3	98.75	106	9.2	0.4	98.75
81	7.5	0.1	98.75	107	9.4	0.1	98.75
82	7.6	0.1	98.75	108	9.6	0.1	98.75
83	7.7	0.1	98.75	109	9.6	0.3	98.75
84	7.7	0.2	98.75	110	9.6	0.4	98.75
85	7.8	0.1	98.75	111	9.6	0.9	92.5
86	7.9	0.7	92.5	112	9.7	0.4	98.75
87	7.9	0.8	92.5	113	9.7	0.1	98.75
88	8.1	0.1	98.75	114	9.7	0.5	97.5
89	8.1	0.4	98.75	115	9.8	0.1	98.75
90	8.2	0.1	98.75	116	9.9	0.7	92.5

Tabel A.11 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Keempat).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
117	10.1	0.2	98.75	143	12.2	0.8	92.5
118	10.1	0.3	98.75	144	12.2	0.7	92.5
119	10.1	0.8	92.5	145	12.3	0.1	98.75
120	10.3	0.9	92.5	146	12.3	0.8	92.5
121	10.3	0.1	98.75	147	12.3	0.4	98.75
122	10.4	0.4	98.75	148	12.5	0.2	98.75
123	10.5	0.1	98.75	149	12.6	0.1	98.75
124	10.5	0.3	98.75	150	12.6	0.5	98.75
125	10.5	0.9	92.5	151	12.9	0.1	98.75
126	10.7	0.2	98.75	152	12.9	0.7	92.5
127	11	0.1	98.75	153	13.1	0.1	98.75
128	11	0.6	96.25	154	13.1	0.3	98.75
129	11	0.5	98.75	155	13.3	0.1	98.75
130	11.1	0.1	98.75	156	13.3	0.8	92.5
131	11.3	0.6	93.75	157	13.4	0.1	98.75
132	11.4	0.2	98.75	158	13.5	0.1	98.75
133	11.4	0.3	98.75	159	13.5	0.4	98.75
134	11.5	0.7	92.5	160	13.6	0.4	98.75
135	11.5	0.1	98.75	161	13.6	0.7	92.5
136	11.7	0.7	92.5	162	13.6	0.1	98.75
137	11.7	0.5	98.75	163	13.7	0.3	98.75
138	11.8	0.3	98.75	164	13.8	0.3	98.75
139	11.8	0.1	98.75	165	13.8	0.2	98.75
140	12	0.5	98.75	166	13.9	0.6	92.5
141	12.1	0.8	92.5	167	13.9	0.1	98.75
142	12.1	0.3	98.75	168	14	0.1	98.75

Tabel A.12 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Kelima).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
169	14	0.6	97.5	195	16.7	0.6	96.25
170	14.1	0.3	98.75	196	16.7	0.3	98.75
171	14.2	0.1	98.75	197	16.7	0.2	98.75
172	14.2	0.8	92.5	198	16.9	0.3	98.75
173	14.4	0.4	98.75	199	17	0.1	98.75
174	14.5	0.4	98.75	200	17	0.5	98.75
175	14.6	0.5	98.75	201	17	0.7	92.5
176	14.6	0.1	98.75	202	17.1	0.6	92.5
177	14.8	0.6	93.75	203	17.2	0.2	98.75
178	15	1	92.5	204	17.2	0.8	92.5
179	15	0.1	98.75	205	17.4	0.7	92.5
180	15	0.2	98.75	206	17.4	0.6	92.5
181	15.1	0.9	92.5	207	17.4	0.1	98.75
182	15.2	0.4	98.75	208	17.5	0.3	98.75
183	15.2	0.6	92.5	209	17.5	0.7	92.5
184	15.3	0.1	98.75	210	17.7	0.1	98.75
185	15.5	0.1	98.75	211	17.7	0.2	98.75
186	15.6	0.1	98.75	212	17.8	0.1	98.75
187	15.9	0.1	98.75	213	17.8	0.7	92.5
188	15.9	0.2	98.75	214	17.9	0.6	93.75
189	15.9	0.3	98.75	215	17.9	0.4	98.75
190	16	0.3	98.75	216	17.9	0.8	92.5
191	16.2	0.6	96.25	217	17.9	0.5	98.75
192	16.3	0.1	98.75	218	18.2	0.5	98.75
193	16.4	0.2	98.75	219	18.3	0.4	98.75
194	16.5	0.2	98.75	220	18.3	0.3	98.75

Tabel A.13 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Keenam).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
221	18	0.5	98.75	247	21	0.1	98.75
222	19	0.3	98.75	248	21	0.6	96.25
223	19	0.1	98.75	249	21	0.1	98.75
224	19	0.4	98.75	250	21	0.3	98.75
225	19	0.8	92.5	251	21	0.2	98.75
226	19	0.6	93.75	252	21	0.1	98.75
227	19	0.2	98.75	253	21	0.1	98.75
228	19	0.1	98.75	254	21	0.1	98.75
229	19	0.2	98.75	255	21	0.4	98.75
230	19	0.2	98.75	256	21	0.5	98.75
231	19	0.1	98.75	257	21	0.4	98.75
232	20	0.3	98.75	258	21	0.1	98.75
233	20	0.1	98.75	259	22	0.4	98.75
234	20	0.6	96.25	260	22	0.4	98.75
235	20	0.1	98.75	261	22	0.2	98.75
236	20	0.7	92.5	262	22	0.8	92.5
237	20	0.1	98.75	263	22	0.5	98.75
238	20	0.7	92.5	264	22	0.3	98.75
239	20	0.1	98.75	265	22	0.8	92.5
240	20	0.3	98.75	266	22	0.1	98.75
241	20	0.1	98.75	267	22	0.6	92.5
242	20	0.6	92.5	268	22	0.7	92.5
243	20	0.1	98.75	269	22	0.5	97.5
244	20	0.2	98.75	270	22	0.5	98.75
245	20	0.7	92.5	271	22	0.4	98.75
246	20	0.7	92.5	272	22	0.8	92.5

Tabel A.14 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Ketujuh).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
273	23	0.4	98.75	299	25	0.6	97.5
274	23	0.7	92.5	300	25	0.2	98.75
275	23	0.4	98.75	301	25	0.3	98.75
276	23	0.8	92.5	302	25	0.1	98.75
277	23	0.1	98.75	303	25	0.5	97.5
278	23	0.4	98.75	304	25	0.4	98.75
279	23	0.2	98.75	305	25	0.1	98.75
280	23	0.1	98.75	306	25	0.9	92.5
281	23	0.1	98.75	307	25	0.6	92.5
282	23	0.5	98.75	308	25	0.2	98.75
283	23	0.4	98.75	309	25	0.2	98.75
284	23	0.1	98.75	310	26	0.1	98.75
285	23	0.5	98.75	311	26	0.7	92.5
286	23	0.1	98.75	312	26	0.4	98.75
287	24	0.8	92.5	313	26	0.6	96.25
288	24	0.1	98.75	314	26	0.1	98.75
289	24	0.5	98.75	315	26	0.1	98.75
290	24	0.6	92.5	316	26	0.3	98.75
291	24	0.7	92.5	317	26	0.2	98.75
292	24	0.1	98.75	318	26	0.4	98.75
293	24	0.2	98.75	319	26	0.4	98.75
294	24	0.1	98.75	320	26	0.3	98.75
295	24	0.1	98.75	321	26	0.1	98.75
296	24	0.5	98.75	322	26	0.2	98.75
297	25	0.4	98.75	323	27	0.4	98.75
298	25	0.1	98.75	324	27	0.1	98.75

Tabel A.15 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Kedelapan).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
325	26.8	0.4	98.75	351	28.7	0.7	92.5
326	26.9	0.1	98.75	352	28.7	0.3	98.75
327	26.9	0.2	98.75	353	28.8	0.1	98.75
328	27.1	0.4	98.75	354	28.8	0.4	98.75
329	27.1	0.6	97.5	355	28.9	0.1	98.75
330	27.3	0.1	98.75	356	29.2	0.4	98.75
331	27.4	0.8	92.5	357	29.5	0.1	98.75
332	27.4	0.5	98.75	358	29.6	0.1	98.75
333	27.4	0.1	98.75	359	29.6	0.7	92.5
334	27.5	0.5	98.75	360	29.6	0.1	98.75
335	27.5	0.3	98.75	361	29.7	0.5	98.75
336	27.6	0.1	98.75	362	29.8	0.5	98.75
337	27.7	0.1	98.75	363	30.1	0.1	98.75
338	27.8	0.1	98.75	364	30.4	0.1	98.75
339	27.8	0.4	98.75	365	30.4	0.3	98.75
340	28	0.2	98.75	366	30.5	0.1	98.75
341	28	0.1	98.75	367	30.5	0.4	98.75
342	28.2	0.5	98.75	368	30.6	0.2	98.75
343	28.2	0.1	98.75	369	30.6	0.1	98.75
344	28.4	0.1	98.75	370	30.7	0.4	98.75
345	28.5	0.5	97.5	371	31	0.2	98.75
346	28.5	0.8	92.5	372	31	0.4	98.75
347	28.6	0.1	98.75	373	31.1	0.1	98.75
348	28.6	0.5	98.75	374	31.2	0.6	93.75
349	28.6	0.2	98.75	375	31.4	0.3	98.75
350	28.7	0.6	97.5	376	31.4	0.5	98.75

Tabel A.16 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Kesembilan).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
377	31.6	0.6	95	385	32	0.3	98.75
378	31.7	0.3	98.75	386	32	0.4	98.75
379	31.7	0.8	92.5	387	32	0.5	98.75
380	31.7	0.1	98.75	388	32	0.6	95
381	31.9	0.1	98.75	389	32	0.7	92.5
382	31.9	0.2	98.75	390	32	0.8	92.5
383	32	0.1	98.75	391	32	0.9	92.5
384	32	0.2	98.75				

Pada Tabel A.17 sampai Tabel A.19 ditunjukkan hasil klasifikasi dari percobaan implementasi SVM, SA, dan pembobotan *kernel* pada data Breast Cancer. Percobaan ini dilakukan satu kali. Dari percobaan satu kali ini didapatkan 100 hasil klasifikikasi. Percobaan ini dilakukan pada skenario 5 *folds cross validation*

Tabel A.17 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Breast Cancer 5 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	0.9	0.4	99.41	9	2.9	0.6	93.12
2	2.1	0.3	98.98	10	3.9	1	93.12
3	0.6	0.1	96.19	11	4.2	0.8	93.12
4	1	0.2	96.05	12	4.9	0.8	93.12
5	1.6	0.1	95.75	13	5.2	0.9	93.12
6	0.3	0.1	95.61	14	5.3	1.2	93.12
7	0.6	0.1	95.31	15	5.3	1.1	93.12
8	0.8	0.1	95.31	16	5.8	1.4	93.12

Tabel A.18 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Breast Cancer 5 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
17	5.9	1.5	93.12	43	19.9	2.5	93.12
18	6.2	1.6	93.12	44	20.2	2.4	93.12
19	6.4	1.2	93.12	45	20.8	2.2	93.12
20	6.8	1.1	93.12	46	21.4	1.8	93.12
21	7.5	1.4	93.12	47	21.9	2.2	93.12
22	8.3	1.9	93.12	48	22.6	2.7	93.12
23	8.5	1.4	93.12	49	23.1	3	93.12
24	9.4	1.8	93.12	50	23.6	3.4	93.12
25	9.8	1.7	93.12	51	24.3	3.4	93.12
26	10.6	1.4	93.12	52	24.5	3.3	93.12
27	11.3	1.6	93.12	53	25	3.3	93.12
28	11.6	1.8	93.12	54	26	3	93.12
29	11.6	2	93.12	55	26.3	3	93.12
30	12.1	1.9	93.12	56	26.6	2.5	93.12
31	12.7	1.9	93.12	57	27.4	2.5	93.12
32	13.3	1.9	93.12	58	28.1	2.2	93.12
33	14.3	1.9	93.12	59	28.2	2.6	93.12
34	15.3	2.3	93.12	60	28.4	2.7	93.12
35	15.7	2.4	93.12	61	28.6	2.4	93.12
36	15.9	2	93.12	62	28.7	2	93.12
37	16	2.4	93.12	63	29.6	2.3	93.12
38	16.4	2.8	93.12	64	30	2.6	93.12
39	17	2.4	93.12	65	30.4	2.8	93.12
40	17.9	2.6	93.12	66	31.3	2.7	93.12
41	18.7	2.2	93.12	67	31.3	3.2	93.12
42	19.4	2.5	93.12	68	31.8	3.5	93.12

Tabel A.19 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Breast Cancer 5 Folds (Bagian Ketiga).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
69	32	4	93.12	85	32	3.2	93.12
70	32	4.3	93.12	86	32	3.2	93.12
71	32	4.2	93.12	87	32	3.6	93.12
72	32	4.1	93.12	88	32	4	93.12
73	32	3.8	93.12	89	32	3.8	93.12
74	32	3.7	93.12	90	32	4.2	93.12
75	32	3.8	93.12	91	32	4.5	93.12
76	32	4	93.12	92	32	4.9	93.12
77	32	4.2	93.12	93	32	5	93.12
78	32	4	93.12	94	32	5.2	93.12
79	32	3.9	93.12	95	32	4.8	93.12
80	32	3.7	93.12	96	32	4.5	93.12
81	32	3.7	93.12	97	32	4.8	93.12
82	32	3.4	93.12	98	32	4.9	93.12
83	32	2.9	93.12	99	32	4.6	93.12
84	32	2.9	93.12	100	32	4.8	93.12

Pada Tabel A.20 sampai Tabel A.21 ditunjukkan hasil klasifikasi dari percobaan implementasi SVM, SA, dan pembobotan *kernel* pada data Breast Cancer. Percobaan ini dilakukan satu kali. Dari percobaan satu kali ini didapatkan 79 hasil klasifikikasi. Percobaan ini dilakukan pada skenario 10 *folds cross validation*

Tabel A.20 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Breast Cancer 10 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	1.5	0.4	100	27	8.1	0.1	98.24
2	6.5	0.4	100	28	7.5	0.1	98.1
3	7.1	0.4	100	29	7.5	0.1	98.1
4	7.4	0.4	100	30	7.4	0.1	97.95
5	6.5	0.3	99.85	31	7.2	0.1	97.95
6	7.4	0.4	99.85	32	7.5	0.1	97.95
7	7.6	0.4	99.85	33	7.4	0.1	97.95
8	7.7	0.4	99.85	34	8.4	0.1	97.95
9	8.3	0.4	99.85	35	7.9	0.5	97.95
10	3.3	0.4	99.71	36	6.6	0.5	97.8
11	7.4	0.3	99.71	37	8.2	0.5	97.8
12	7.4	0.3	99.71	38	7.4	0.1	97.8
13	8	0.4	99.56	39	8.2	0.5	97.8
14	7.5	0.3	99.41	40	8.2	0.1	97.8
15	7.6	0.4	98.98	41	8.2	0.5	97.66
16	7.7	0.2	98.98	42	1.1	0.5	97.07
17	7.9	0.2	98.83	43	6.7	0.5	96.93
18	7.5	0.2	98.68	44	7.2	0.5	96.93
19	1.3	0.2	98.39	45	1.3	0.6	96.63
20	8.2	0.1	98.39	46	2.2	0.6	96.63
21	7.6	0.1	98.39	47	2.8	0.6	96.63
22	8	0.2	98.39	48	4	0.8	96.63
23	0.5	0.1	98.24	49	5	0.7	96.63
24	7.3	0.1	98.24	50	5.9	1.2	96.63
25	6.8	0.1	98.24	51	6.2	0.9	96.63
26	7.6	0.1	98.24	52	6.4	0.7	96.63

Tabel A.21 Hasil Percobaan SVM-SA Tanpa Pembobotan Kernel Data Breast Cancer 10 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
53	6.5	0.6	96.63	67	7.9	0.6	96.63
54	7.5	0.8	96.63	68	7.5	0.6	96.63
55	6.6	0.8	96.63	69	8.3	0.7	96.63
56	7.2	0.7	96.63	70	7.5	0.7	96.63
57	6.6	0.7	96.63	71	7.6	0.7	96.63
58	7.5	0.8	96.63	72	7.5	0.6	96.63
59	7.3	0.7	96.63	73	7.8	0.7	96.63
60	8	0.6	96.63	74	8.2	0.8	96.63
61	8	0.6	96.63	75	7.9	0.6	96.63
62	8	0.9	96.63	76	7.4	0.7	96.63
63	8.2	0.7	96.63	77	7.8	0.8	96.63
64	7.6	0.8	96.63	78	8.2	0.6	96.63
65	8.1	0.8	96.63	79	8.2	0.7	96.63
66	7.6	0.6	96.63				

Pada bagian lampiran ini berisi hasil-hasil percobaan secara keseluruhan. Tabel A.22 sampai Tabel A.28 berisi hasil percobaan SVM, SA, dan pembobotan kernel Gradient Descent 5 *folds cross validation*. Pada awalnya percobaan ini dilakukan 5 kali kemudian hasil dari kelima percobaan ini digabungkan dan hasil yang serupa dieliminasi sehingga tidak ada data yang identik.

Tabel A.22 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	0.1	0.2	97.5	2	0.1	0.1	97.5

Tabel A.23 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
3	2	0.9	97.5	29	13.2	0.1	97.5
4	2.3	0.1	97.5	30	13.3	0.1	97.5
5	2.8	1.2	97.5	31	13.4	0.8	96.25
6	3.5	0.3	97.5	32	13.4	0.9	97.5
7	4.6	0.1	97.5	33	13.5	0.3	97.5
8	5.8	0.8	97.5	34	13.6	0.1	97.5
9	6.7	0.4	97.5	35	13.7	1.2	95
10	7.5	1.1	97.5	36	13.7	1.1	97.5
11	8.2	0.9	96.25	37	13.8	0.3	97.5
12	9.5	0.7	97.5	38	13.8	1.2	97.5
13	9.8	0.4	97.5	39	13.8	0.6	97.5
14	10.7	0.8	97.5	40	13.9	0.5	96.25
15	10.9	0.2	97.5	41	13.9	0.1	97.5
16	11.3	0.2	97.5	42	13.9	0.6	97.5
17	12.3	1.3	96.25	43	13.9	0.9	97.5
18	12.3	1.4	95	44	14	0.7	97.5
19	12.4	0.7	97.5	45	14.3	1.3	95
20	12.5	0.2	97.5	46	14.4	0.3	97.5
21	12.6	0.4	97.5	47	14.5	0.2	97.5
22	12.7	0.4	97.5	48	14.5	1	96.25
23	12.8	1	97.5	49	14.7	0.9	96.25
24	13	0.8	97.5	50	14.8	0.1	97.5
25	13	0.5	97.5	51	14.8	0.4	97.5
26	13	1.1	97.5	52	14.8	0.5	97.5
27	13.1	0.3	97.5	53	14.9	1	96.25
28	13.1	0.6	97.5	54	14.9	1	97.5

Tabel A.24 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Ketiga).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
55	15	0.3	97.5	81	17	0.2	97.5
56	15	0.4	96.25	82	17.1	0.6	97.5
57	15	0.1	97.5	83	17.2	0.5	97.5
58	15	0.9	97.5	84	17.3	0.5	97.5
59	15	0.5	97.5	85	17.3	0.9	97.5
60	15	0.4	97.5	86	17.3	0.1	97.5
61	15	0.7	97.5	87	17.4	0.3	97.5
62	15	0.4	97.5	88	17.5	0.1	97.5
63	16	0.1	97.5	89	17.6	0.1	97.5
64	16	0.2	97.5	90	17.7	0.1	97.5
65	16	0.4	97.5	91	17.9	0.7	97.5
66	16	0.6	97.5	92	18	0.6	97.5
67	16	1	97.5	93	18	0.3	97.5
68	16	0.2	97.5	94	18.1	0.1	97.5
69	16	0.2	97.5	95	18.3	0.5	97.5
70	16	0.8	97.5	96	18.5	0.6	97.5
71	16	0.1	97.5	94	18.1	0.1	97.5
72	16	0.7	97.5	95	18.3	0.5	97.5
73	16	0.7	96.25	96	18.5	0.6	97.5
74	16	0.3	97.5	97	18.6	0.6	97.5
75	16	0.6	97.5	98	18.6	0.3	97.5
76	16	0.1	97.5	99	18.8	0.7	97.5
77	16	0.7	97.5	100	18.8	0.5	97.5
78	17	0.3	97.5	101	18.8	0.1	97.5
79	17	0.6	97.5	102	18.9	0.3	97.5
80	17	0.2	97.5	103	19	0.1	97.5

Tabel A.25 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Keempat).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
104	19.1	0.3	97.5	130	21.8	0.6	97.5
105	19.2	0.2	97.5	131	21.9	0.4	97.5
106	19.3	0.1	97.5	132	21.9	0.1	97.5
107	19.5	0.6	97.5	133	22	0.1	97.5
108	19.5	0.1	97.5	134	22.2	0.1	97.5
109	19.9	0.1	97.5	135	22.2	0.4	96.25
110	20	0.4	97.5	136	22.3	0.6	97.5
111	20	0.1	97.5	137	22.4	0.3	97.5
112	20.1	0.4	97.5	138	22.5	0.6	97.5
113	20.1	0.1	97.5	139	22.5	0.3	97.5
114	20.4	0.9	97.5	140	22.6	0.3	97.5
115	20.4	0.1	97.5	141	22.8	0.7	97.5
116	20.5	0.5	97.5	142	22.8	0.3	97.5
117	20.6	0.1	97.5	143	23	0.2	97.5
118	20.7	0.1	97.5	144	23	0.6	97.5
119	20.8	0.1	97.5	145	23.1	0.4	97.5
120	20.8	0.5	97.5	146	23.2	1.1	97.5
121	20.9	0.2	97.5	147	23.2	1	97.5
122	21	0.4	97.5	148	23.5	1.3	95
123	21	0.1	97.5	149	23.5	0.7	97.5
124	21	0.6	97.5	150	23.5	0.1	97.5
125	21.2	0.3	97.5	151	23.6	0.8	97.5
126	21.2	0.1	97.5	152	23.7	0.4	97.5
127	21.4	0.4	97.5	153	23.8	1	96.25
128	21.4	0.5	97.5	154	23.9	0.1	97.5
129	21.7	0.3	97.5	155	23.9	0.3	97.5

Tabel A.26 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Kelima).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
156	24	0.1	97.5	182	26	0.3	97.5
157	24	0.4	97.5	183	26	0.3	97.5
158	24.1	0.7	97.5	184	26	0.1	97.5
159	24.2	0.8	97.5	185	26	0.7	96.25
160	24.2	0.1	97.5	186	26	0.5	97.5
161	24.5	1.3	97.5	187	26	1.1	97.5
162	24.6	1.3	95	188	26	0.9	97.5
163	24.6	0.9	97.5	189	27	0.1	97.5
164	24.7	0.5	97.5	190	27	0.5	97.5
165	24.8	1	97.5	191	27	0.3	97.5
166	24.9	0.1	97.5	192	27	0.5	97.5
167	25	0.9	96.25	193	27	0.9	96.25
168	25.1	0.3	97.5	194	27	0.9	97.5
169	25.1	0.5	96.25	195	27	0.6	97.5
170	25.1	0.1	97.5	196	27	0.9	97.5
171	25.2	0.9	97.5	197	27	0.8	97.5
172	25.3	0.5	97.5	198	27	0.2	97.5
173	25.4	0.1	97.5	199	27	0.5	96.25
174	25.5	0.1	97.5	200	27	0.4	97.5
175	25.6	0.1	97.5	201	27	0.3	97.5
176	25.6	0.5	97.5	202	27	0.7	97.5
177	25.7	1.4	95	203	28	0.7	97.5
178	25.7	0.3	97.5	204	28	0.2	97.5
179	25.8	0.9	97.5	205	28	0.5	96.25
180	25.8	0.2	97.5	206	28	0.4	97.5
181	25.8	0.1	97.5	207	28	1.1	97.5

Tabel A.27 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Keenam).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
208	28.1	0.2	97.5	234	31	0.1	97.5
209	28.1	0.5	97.5	235	31	0.6	97.5
210	28.4	0.4	97.5	236	31	1.2	96.25
211	28.4	0.8	97.5	237	31	0.7	97.5
212	28.5	0.2	97.5	238	31	0.1	97.5
213	28.6	0.3	97.5	239	31	0.9	97.5
214	28.6	0.6	97.5	240	31	0.1	97.5
215	28.8	0.1	97.5	241	31	1.2	97.5
216	28.9	0.4	97.5	242	31	1.5	96.25
217	28.9	0.9	96.25	243	31	0.7	97.5
218	29.2	0.3	97.5	244	31	0.9	97.5
219	29.2	0.5	97.5	245	31	0.6	97.5
220	29.3	1	97.5	246	32	1.1	93.75
221	29.4	0.4	97.5	247	32	0.8	96.25
222	29.5	0.1	97.5	248	32	0.3	97.5
223	29.5	0.3	97.5	249	32	1.3	97.5
224	29.7	0.1	97.5	250	32	0.1	97.5
225	29.7	0.7	97.5	251	32	0.2	97.5
226	29.8	0.4	97.5	252	32	0.3	97.5
227	30	1.3	97.5	253	32	0.4	97.5
228	30.1	0.1	97.5	254	32	0.5	97.5
229	30.1	1	96.25	255	32	0.6	97.5
230	30.2	1	97.5	256	32	0.7	97.5
231	30.3	0.3	97.5	257	32	0.8	97.5
232	30.3	0.7	97.5	258	32	0.9	97.5
233	30.4	1.2	97.5	259	32	1	97.5

Tabel A.28 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 5 Folds (Bagian Ketujuh).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
260	32	1.1	97.5	264	32	1.5	97.5
261	32	1.2	97.5	265	32	1.6	97.5
262	32	1.3	97.5	266	32	1.7	95
263	32	1.4	97.5	267	32	2	95

Pada Tabel A.29 sampai Tabel A.35 ditunjukkan hasil klasifikasi dari percobaan implementasi SVM, SA, dan pembobotan *kernel* pada data Hepatitis dengan 10 *folds cross validation*. Prosedur yang dilakukan sama dengan percobaan dengan 5 *folds cross validation*. Hasil dari beberapa kali percobaan digabungkan kemudian data yang identik dihapus.

Tabel A.29 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	1.5	0.5	98.75	13	8.6	0.1	98.75
2	1.5	0.1	98.75	14	8.7	3.1	97.5
3	2.4	0.8	98.75	15	8.7	0.2	98.75
4	2.6	1.4	98.75	16	9.3	0.8	98.75
5	3.8	0.7	98.75	17	9.4	3.5	93.75
6	5	1.8	98.75	18	9.8	2.2	98.75
7	5.5	2.2	98.75	19	10.2	1.4	98.75
8	5.9	2.2	98.75	20	10.6	2.2	98.75
9	6.4	0.4	98.75	21	8.7	0.2	98.75
10	7.1	0.4	98.75	22	9.3	0.8	98.75
11	7.4	2.2	98.75	23	9.4	3.5	93.75
12	8.1	1.5	98.75	24	9.8	2.2	98.75

Tabel A.30 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
25	10.2	1.4	98.75	51	13	2.2	98.75
26	10.6	2.2	98.75	52	13.1	2.1	98.75
27	10.9	3	98.75	53	13.1	2.9	97.5
28	11.1	2.1	98.75	54	13.1	2.2	98.75
29	11.7	1.8	98.75	55	13.2	3	97.5
30	11.7	3.2	96.25	56	13.2	0.3	98.75
31	11.7	0.1	98.75	57	13.4	2.1	98.75
32	11.9	0.5	98.75	58	13.5	2.6	98.75
33	12	2.9	97.5	59	13.5	0.3	98.75
34	12	1.8	98.75	60	13.8	0.1	98.75
35	12.1	1.6	98.75	61	13.8	0.6	98.75
36	12.1	0.1	98.75	62	14	2.4	98.75
37	12.3	0.3	98.75	63	14	2.5	98.75
38	12.3	0.2	98.75	64	14	0.1	98.75
39	12.5	2.9	98.75	65	14.1	2.9	100
40	12.5	3.4	95	66	14.1	0.6	98.75
41	12.5	3.2	97.5	67	14.2	1.1	98.75
42	12.7	2.5	98.75	68	14.3	0.6	98.75
43	12.7	0.2	98.75	69	14.5	1	98.75
44	12.7	0.5	98.75	70	14.7	1.4	98.75
45	12.8	3.3	96.25	71	14.8	1.1	98.75
46	12.8	1.9	98.75	72	14.8	2.6	98.75
47	13	1.8	98.75	73	14.9	2.1	98.75
48	13	2.9	97.5	74	15	2.1	98.75
49	13	3.3	96.25	75	15	2.4	98.75
50	13	0.3	98.75	76	15	2.3	98.75

Tabel A.31 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Ketiga).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
77	15	0.3	98.75	103	17.4	2.5	98.75
78	15.1	1.2	98.75	104	17.4	3.3	96.25
79	15.2	2.5	98.75	105	17.5	2.5	98.75
80	15.2	2.6	97.5	106	17.7	3	96.25
81	15.3	0.1	98.75	107	17.8	3.4	95
82	15.4	2.3	98.75	108	17.8	2.6	98.75
83	15.4	2.8	98.75	109	17.9	2.9	98.75
84	15.4	0.1	98.75	110	17.9	1.1	98.75
85	15.5	0.1	98.75	111	17.9	3.4	95
86	15.9	3.1	98.75	112	17.9	0.5	98.75
87	16	2.4	98.75	113	18.2	2.3	98.75
88	16	2.3	98.75	114	18.3	1.8	98.75
89	16	1.1	98.75	115	18.3	3.5	92.5
90	16.2	2.1	98.75	116	18.4	2	98.75
91	16.2	0.6	98.75	117	18.4	2.6	98.75
92	16.3	2.1	98.75	118	18.5	2.6	98.75
93	16.4	0.9	98.75	119	18.5	1.8	98.75
94	16.5	2.5	98.75	120	18.7	2.2	98.75
95	16.6	3.1	97.5	121	18.7	2.5	97.5
96	16.6	0.9	98.75	122	18.7	1	98.75
97	16.9	3.1	97.5	123	18.8	2.2	98.75
98	17.1	2.3	98.75	124	18.8	0.8	98.75
99	17.2	0.9	98.75	125	18.9	2.5	98.75
100	17.2	0.5	98.75	126	18.9	0.6	98.75
101	17.3	2.3	98.75	127	19	2.3	97.5
102	17.3	3	97.5	128	19	2.2	98.75

Tabel A.32 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Keempat).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
129	19	1.3	98.75	155	20.5	1.9	98.75
130	19.1	1	98.75	156	20.5	3.1	96.25
131	19.2	2.3	98.75	157	20.5	0.5	98.75
132	19.2	0.6	98.75	158	20.9	0.1	98.75
133	19.4	2.3	98.75	159	20.9	0.4	98.75
134	19.6	2.6	97.5	160	21.1	2.3	98.75
135	19.7	2.2	98.75	161	21.3	1.9	98.75
136	19.7	2.1	98.75	162	21.4	0.2	98.75
137	19.9	2.3	98.75	163	21.5	2.9	98.75
138	19.9	0.2	98.75	164	21.5	0.5	98.75
139	20	1.9	98.75	165	21.5	0.8	98.75
140	20	2.4	98.75	166	21.8	2.8	98.75
141	20	2.2	98.75	167	21.8	2.6	98.75
142	20.1	2.5	97.5	168	22	0.1	98.75
143	20.1	0.2	98.75	169	22.2	2.9	98.75
144	20.2	2	98.75	170	22.2	1.9	98.75
145	20.2	2.1	98.75	171	22.4	1.1	98.75
146	20.2	2.5	98.75	172	22.5	1	98.75
147	20.2	2.3	98.75	173	22.6	3.1	97.5
148	20.2	2.7	97.5	174	22.8	3.1	98.75
149	20.3	2.7	98.75	175	22.9	0.1	98.75
150	20.4	2	98.75	176	23.1	3.2	97.5
151	20.4	2.1	98.75	177	23.1	1.7	98.75
152	20.4	0.4	98.75	178	23.1	2.1	98.75
153	20.5	2.3	98.75	179	23.3	1.2	98.75
154	20.5	2.7	100	180	23.5	0.1	98.75

Tabel A.33 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Kelima).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
181	23.7	2.9	98.75	207	26.6	2.1	98.75
182	24	1.2	98.75	208	26.6	2.2	98.75
183	24.1	2.7	97.5	209	26.7	0.9	98.75
184	24.1	1.7	98.75	210	26.7	2.4	98.75
185	24.1	0.4	98.75	211	26.8	1	98.75
186	24.3	1.3	98.75	212	26.9	1.5	98.75
187	24.5	1.2	98.75	213	26.9	3.2	98.75
188	24.5	0.6	98.75	214	26.9	3.7	92.5
189	24.5	2.8	98.75	215	26.9	3.4	92.5
190	24.6	3	97.5	216	27	1	98.75
191	24.6	1.4	98.75	217	27.2	3.5	92.5
192	24.9	2.7	98.75	218	27.4	1.7	98.75
193	24.9	1.4	98.75	219	27.4	2.9	97.5
194	25.1	3.2	97.5	220	27.5	2.5	98.75
195	25.2	0.9	98.75	221	27.6	1.5	98.75
196	25.3	2.4	98.75	222	27.6	3.3	96.25
197	25.3	0.7	98.75	223	27.7	1.2	98.75
198	25.3	0.6	98.75	224	27.7	3.2	98.75
199	25.4	3.3	96.25	225	27.7	2.9	97.5
200	25.8	0.5	98.75	226	27.8	2.9	97.5
201	25.8	1.9	98.75	227	27.8	2.6	98.75
202	26.1	2.2	98.75	228	27.9	1.1	98.75
203	26.1	0.9	98.75	229	27.9	2.5	98.75
204	26.3	3.3	95	230	28	0.9	98.75
205	26.4	3.1	98.75	231	28	3	97.5
206	26.5	3	98.75	232	28.1	2	98.75

Tabel A.34 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Keenam).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
233	28.1	2.1	100	259	30.5	1.6	98.75
234	28.1	2.5	97.5	260	30.5	2.2	98.75
235	28.2	2.7	98.75	261	30.7	0.4	98.75
236	28.2	2.3	98.75	262	30.8	2.1	98.75
237	28.3	1.8	98.75	263	30.8	0.4	98.75
238	28.4	1.8	98.75	264	30.9	0.4	98.75
239	28.4	3.4	96.25	265	31.1	0.1	98.75
240	28.6	1.4	98.75	266	31.1	0.2	98.75
241	28.6	2.7	97.5	267	31.1	2.3	98.75
242	28.6	2.5	97.5	268	31.2	2.1	100
243	28.7	2.3	98.75	269	31.2	0.6	98.75
244	28.8	2.3	98.75	270	31.3	2.1	98.75
245	29	0.5	98.75	271	31.3	2.3	98.75
246	29.1	2	98.75	272	31.3	1.9	98.75
247	29.4	1.1	98.75	273	31.4	1.8	98.75
248	29.4	2.4	98.75	274	31.4	2.4	98.75
249	29.5	2.3	98.75	275	31.5	2.2	98.75
250	29.6	0.2	98.75	276	31.5	1.7	98.75
251	29.8	2.6	98.75	277	31.5	2.3	98.75
252	30	0.3	98.75	278	31.5	2.6	98.75
253	30.1	1.9	98.75	279	31.5	0.7	98.75
254	30.1	2.6	98.75	280	31.6	2.6	97.5
255	30.2	0.7	98.75	281	31.6	2.1	98.75
256	30.3	1.8	98.75	282	31.6	1.9	98.75
257	30.3	2.7	98.75	283	31.7	1.7	98.75
258	30.4	2.3	98.75	284	31.8	2.1	98.75

Tabel A.35 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Hepatitis 10 Folds (Bagian Ketujuh).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
285	31.8	1.8	98.75	301	32	1	98.75
286	31.8	2.2	98.75	302	32	1.1	98.75
287	31.9	2.2	98.75	303	32	1.2	98.75
288	31.9	2.5	98.75	304	32	1.4	98.75
289	31.9	1.7	98.75	305	32	1.5	98.75
290	31.9	0.5	98.75	306	32	1.6	98.75
291	31.9	2.1	98.75	307	32	1.7	100
292	32	0.1	98.75	308	32	1.9	98.75
293	32	0.2	98.75	309	32	2	98.75
294	32	0.3	98.75	310	32	2.1	98.75
295	32	0.4	98.75	311	32	2.2	98.75
296	32	0.5	98.75	312	32	2.3	98.75
297	32	0.6	98.75	313	32	2.4	98.75
298	32	0.7	98.75	314	32	2.5	100
299	32	0.8	98.75	315	32	2.6	98.75
300	32	0.9	98.75	316	32	2.7	98.75

Pada Tabel A.36 sampai Tabel A.37 ditunjukkan hasil klasifikasi dari percobaan implementasi SVM, SA, dan pembobotan *kernel* pada data Breast Cancer. Percobaan ini dilakukan satu kali. Dari percobaan satu kali ini didapatkan 100 hasil klasifikikasi. Percobaan ini dilakukan pada skenario 10 *folds cross validation.*

Tabel A.36 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Breast Cancer 10 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	0.5	0.1	98.24	26	4.9	0.1	97.66
2	1.1	0.3	98.39	27	4.9	0.3	98.39
3	2.1	0.4	93.85	28	5.2	0.3	98.39
4	1.9	0.2	98.1	29	5.8	0.8	93.56
5	1.8	0.7	93.56	30	5.6	0.7	93.56
6	1.1	0.7	93.56	31	5.7	0.1	98.1
7	1.5	0.4	94.14	32	6.2	0.2	97.95
8	2.1	0.1	98.39	33	5.4	0.1	98.1
9	3	0.1	97.8	34	5.8	0.1	97.95
10	3.1	0.3	98.1	35	5.9	0.4	97.51
11	2.9	0.1	98.24	36	5.3	0.6	93.56
12	2.9	0.1	98.1	37	5.3	0.6	93.56
13	2.3	0.1	98.1	38	5.8	0.6	93.56
14	2.3	0.4	94.58	39	5.4	0.1	98.54
15	2.8	0.2	98.39	40	5.9	0.1	98.24
16	3.1	0.3	98.1	41	5.7	0.4	97.22
17	3.5	0.1	97.95	42	5.5	0.2	97.95
18	3.4	0.5	93.56	43	6.4	0.3	98.24
19	3	0.1	97.95	44	5.5	0.3	97.95
20	3.4	0.2	98.24	45	6.3	0.2	97.66
21	3.2	0.1	98.1	46	5.8	0.1	98.1
22	3.8	0.1	98.39	47	5.7	0.1	98.54
23	4.3	0.1	97.95	48	6.5	0.3	97.95
24	4.6	0.4	94.14	49	6.7	0.1	98.39
25	4.3	0.1	98.39	50	6.4	0.1	97.95

Tabel A.37 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Breast Cancer 10 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
51	6.1	0.1	98.1	76	7.3	0.5	93.56
52	6.4	0.1	97.95	77	7.5	0.5	93.56
53	6.3	0.5	93.85	78	6.7	0.1	98.1
54	6.5	0.1	97.95	79	7.3	0.1	98.24
55	6.4	0.1	97.95	80	6.6	0.3	98.39
56	6.5	0.1	98.54	81	7.2	0.4	94.73
57	6.6	0.2	97.8	82	7.2	0.1	98.1
58	7.1	0.1	97.95	83	7.5	0.4	94
59	6.5	0.2	98.1	84	7.4	0.1	98.54
60	6.6	0.1	97.95	85	8.1	0.3	98.24
61	6.6	0.4	96.93	86	7.4	0.6	93.56
62	6.6	0.1	98.24	87	7.8	0.4	94.44
63	7	0.3	98.24	88	8.2	0.4	94.73
64	7.5	0.1	97.95	89	8.1	0.3	97.95
65	6.9	0.1	97.8	90	7.8	0.1	97.95
66	6.8	0.3	97.51	91	8.3	0.2	98.24
67	7	0.1	98.39	92	7.7	0.5	93.85
68	6.8	0.1	98.24	93	7.5	0.1	98.39
69	6.9	0.1	98.24	94	8.1	0.1	98.1
70	6.8	0.5	93.7	95	8	0.1	98.24
71	6.9	0.1	98.24	96	8.3	0.5	93.7
72	7.4	0.1	97.95	97	8.3	0.5	93.56
73	7.2	0.1	97.66	98	7.9	0.4	98.54
74	7.1	0.1	98.1	99	8.7	0.5	93.7
75	7	0.5	93.7	100	8.8	0.1	98.39

Pada Tabel A.38 sampai Tabel A.39 ditunjukkan hasil klasifikasi dari percobaan implementasi SVM, SA, dan pembobotan *kernel* pada data Breast Cancer. Percobaan ini dilakukan satu kali. Dari percobaan satu kali ini didapatkan 85 hasil klasifikikasi. Percobaan ini dilakukan pada skenario 5 *folds cross validation.*

Tabel A.38 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Breast Cancer 5 Folds (Bagian Pertama).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
1	0.3	0.1	95.61	21	1.5	0.1	95.31
2	0.4	0.1	95.61	22	1.7	0.4	87.55
3	1.2	0.1	95.17	23	1.5	0.7	87.12
4	0.5	0.1	95.75	24	1.3	0.4	91.07
5	1.2	0.1	95.9	25	1.4	0.1	95.02
6	1.9	0.3	95.75	26	1.4	0.3	95.75
7	1.5	0.2	95.46	27	2.1	0.1	95.9
8	1.8	0.5	87.41	28	1.3	0.7	87.12
9	1.5	0.2	95.61	29	2.2	0.7	87.12
10	1.2	0.1	96.05	30	1.7	0.4	96.34
11	1.5	0.3	95.46	31	2.4	0.6	87.12
12	2	0.1	95.46	32	2	0.1	96.05
13	1.8	0.1	95.9	33	2.2	0.2	95.61
14	1.2	0.3	96.19	34	2.3	0.1	95.75
15	2.1	0.6	87.12	35	2.5	0.5	87.26
16	1.8	0.2	95.31	36	1.7	0.1	96.05
17	1.4	0.5	87.26	37	2.1	0.5	87.26
18	1.4	0.1	95.75	38	2.6	0.1	95.46
19	1.6	0.1	95.61	39	1.8	0.1	95.75
20	2.2	0.1	95.46	40	1.9	0.1	95.61

Tabel A.39 Hasil Percobaan SVM, SA, dan Pembobotan Kernel Data Breast Cancer 5 Folds (Bagian Kedua).

Nomor	C	σ	Akurasi	Nomor	C	σ	Akurasi
41	2.6	0.3	95.31	64	4.2	0.1	96.49
42	2.3	0.7	87.12	65	5	0.4	87.85
43	2.2	0.7	87.12	66	4.5	0.1	95.9
44	2.5	0.7	87.12	67	4.4	0.1	95.75
45	2.1	0.1	95.02	68	4.3	0.1	95.9
46	2	0.7	87.12	69	5.1	0.1	95.61
47	1.8	0.5	87.41	70	4.2	0.5	87.12
48	2.4	0.6	87.12	71	4.7	0.6	87.12
49	2	0.5	87.12	72	4.4	0.1	95.9
50	2.1	0.2	95.02	73	4.3	0.2	96.05
51	1.9	0.2	95.61	74	4.6	0.1	95.46
52	2.6	0.5	87.12	75	4.6	0.1	95.31
53	2.2	0.1	95.46	76	4.3	0.2	95.17
54	1.7	0.3	96.05	77	5	0.1	95.46
55	2.5	0.6	87.12	78	5	0.1	95.46
56	2.3	0.8	87.12	79	4.9	0.4	87.99
57	1.7	0.7	87.12	80	4.9	0.5	87.12
58	1.8	0.4	95.02	81	4.9	0.2	95.02
59	2	0.7	87.12	82	4.2	0.1	95.17
60	2.4	0.6	87.12	83	5.1	0.2	95.9
61	2.8	0.8	87.12	84	5.1	0.6	87.12
62	2.6	0.1	95.02	85	4.8	0.1	95
63	3.3	0.1	96.05				

Keterangan:

Akurasi yang disebutkan pada Bab Uji Coba dan Analisis Hasil