Perbandingan SVM dan *Perceptron* dengan Optimasi *Heuristik*

Muchamad Kurniawan[#1], Maftahatul Hakimah[*2], Siti Agustini[#3]

[*]*Program Studi Teknik Informatika, Fakultas Teknik Elektro dan Teknologi Informasi,*

[#]*Program Studi Sistem Komputer, Fakultas Teknik Elektro dan Teknologi Informasi,*

Institut Teknologi Adhi Tama Surabaya

[1]muchamad.kurniawan@itats.ac.id

[2]hakimah.mafta@itats.ac.id

[3]sitiagustini@itats.ac.id

Abstract*— Support Vector Machine (SVM) and Perceptron are methods used in machine learning to determine classification. Both methods have the same motivation, namely to get the dividing line (hyperplane). Hyperplane can be obtained by using the optimization method Gradient Descent (GD), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). This study compares machine learning methods (Support Vector Machine and Perceptron) to optimization methods (Gradient Descent, Genetic Algorithm, and Particle Swarm Optimization) to find hyperplane. The dataset used is Iris Flower obtained from the UCI Machine Learning Repository. The test parameter on the Perceptron is the learning rate, while the optimization algorithm (GA and PSO) is the number of individuals. The results showed that the most suitable optimization method for Perceptron and SVM is PSO, with an accuracy value of 93%.

***Keywords*— support vector machine, perceptron, gradient descent, genetic algorithm, particle swarm optimization.**

Abstrak*— *Support Vector Machine* (SVM) dan *Perceptron* merupakan metode yang digunakan dalam *machine learning* untuk penentuan klasifikasi. Kedua metode tersebut memiliki motivasi yang sama, yaitu untuk mendapatkan garis pemisah (*hyperplane*). *Hyperplane* bisa didapatkan dengan metode optimasi *Gradient Descent* (GD), *Genetic Algorithm* (GA), dan *Particle Swarm Optimization* (PSO). Penelitian ini membandingkan metode *machine learning* (*Support Vector Machine* dan Perceptron) terhadap metode optimasi (*Gradient Descent*, *Genetic Algorithm*, dan *Particle Swarm Optimization*) untuk menemukan *hyperplane*. *Dataset* yang digunakan adalah *Iris Flower* yang diperoleh dari UCI *Machine Learning Repository*. Parameter pengujian pada *Perceptron* adalah *learning rate*, sedangkan pada algoritme optimasi (GA dan PSO) adalah jumlah individu. Hasil penelitian menunjukkan bahwa metode optimasi yang paling cocok untuk *Perceptron* dan SVM adalah PSO, dengan nilai akurasi 93%.

Kata Kunci— ***support vector machine*, *perceptron*, *gradient descent*, *genetic algorithm*, *particle swarm optimization*.**

I. PENDAHULUAN

Klasifikasi sebagai teknik mesin pembelajaran untuk memprediksi keanggotaan grup (kelas) dari sekumpulan data dapat digunakan untuk menganalisis dan mendeskripsikan data penting berdasarkan grup, atau untuk memprediksi suatu tren. *Support Vector Machine* (SVM) dan *Perceptron* merupakan contoh teknik klasifikasi. *Support Vector Machine* (SVM) pertama kali diperkenalkan oleh Vapnik dan Cortes pada tahun 1992 [1]. SVM merupakan metode klasifikasi yang telah banyak digunakan pada berbagai bidang dan *dataset*, seperti: pembentukan model data gas dan minyak [2], diagnosis potensi kegagalan pembangkitan listrik [3], uji coba dengan *dataset* yang besar [4], pengklasifikasian bebatuan dan minyak [5], implementasi pada dinamik heteroskedastisitas, *Least Square* SVM digunakan untuk predeksi kelongsoran tanah [6], dan lain-lain.

Inti dari metode klasifikasi adalah bagaimana memisahkan data-data yang berbeda kelas. Motivasi SVM adalah menemukan garis pemisah (*hyperplane*) untuk memisahkan data positif dan negatif. Motivasi yang sama telah dikembangkan terlebih dahulu pada metode *Perceptron*. *Perceptron* merupakan dasar metode *Backpropogation*, *Deep Learning*, *Convolution Neural Network,* dan juga merupakan dasar dari metode SVM. Dengan *hyperplane* yang sama, maka persamaan pembentuknya juga pasti sama. Formula yang digunakan adalah persamaan garis lurus, seperti pada persamaan 1. Pada persamaan 1 terdapat variabel bobot *w,* data *x*, dan bias *b*, di mana *w* dan *b* adalah variabel bebas yang nilainya harus dicari untuk mendapatkan hasil yang optimal. Vapnik telah mengembangkan metode ini dari formulasi 1 dengan optimasi Lagragian sehingga bobot dan bias ditransformasikan untuk mencari *alpha* (α) setiap data.

$$f(x) = w.x + b \tag{1}$$

Untuk mendapatkan nilai bobot dan bias, maka diperlukan optimasi. *Perceptron* menggunakan *Gradient Descent* sebagai standar metode optimasinya. SVM menggunakan *Quadratic Programming* (*QP*) sebagai pencari solusi dari pengembangan formula 1 yang telah dioptimasi dengan *Lagrange Multipliers*. Teknik optimasi untuk menyelesaikan QP menggunakan algoritma *"chunking"* [7]. Hasil yang diperoleh oleh SVM adalah solusi terbagus, atau biasa disebut dengan *global optimal*. Namun, algoritme ini mempunyai biaya komputasi yang mahal karena makin banyak data yang diolah, maka semakin tinggi kompleksitasnya, yaitu $O(m^2)$ untuk setiap iterasinya [8].

Pada penelitian sebelumnya telah dilakukan teknik klasifikasi tanpa menggunakan metode optimasi di antaranya adalah Teran, dkk. [9] yang menerapkan pembelajaran *Perceptron* untuk prediksi. Penelitian ini memberikan hasil akurasi hanya sebesar 54% dan masih memiliki kelemahan dalam peningkatan *overhead* ketika fitur lain digabungkan. *Perceptron* juga memiliki kompleksitas sebesar $O = \left(\frac{1}{\gamma^2}\right)$ di mana γ adalah *margin*. Penelitian lain melakukan komparasi akurasi Naïve Bayes, SVM, dan K-NN. Hasil komparasi ketiga teknik klasifikasi tersebut menunjukkan SVM memiliki akurasi terendah sebesar 66% [10] dan 64% [11].

Hasil penelitian sebelumnya juga menunjukkan bahwa SVM dan *Perceptron* masih memiliki akurasi yang cukup rendah. Berdasarkan permasalahan tersebut, penelitian ini memberikan kontribusi dengan menambahkan metode optimasi untuk meningkatkan akurasi. Algoritme optimasi yang digunakan untuk SVM dan *Perceptron* adalah *Gradient Descent* (GD), *Genetic Algorithm* (GA), dan *Particle Swarm Optimization* (PSO). Penelitian ini akan membandingkan akurasi gabungan teknik klasifikasi dan metode optimasi mana yang paling tinggi. Selain itu, disajikan juga hasil identifikasi *misclass* untuk setiap parameter *learning rate* agar memberikan gambaran lebih jelas untuk setiap metode optimasi.

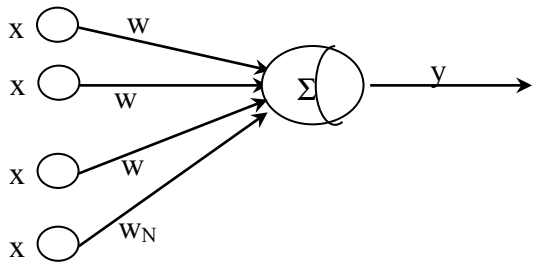
II. METODOLOGI

A. Perceptron

Perceptron adalah bentuk paling sederhana dari Jaringan Syaraf Tiruan (JST) yang digunakan untuk klasifikasi pola data yang terpisah secara linier (*linearly separable*). *Perceptron* terbatas hanya untuk pengelompokkan dalam dua kelas saja [12]. *Perceptron* terdiri atas sejumlah masukan N *node* yang disebut neuron, dengan nilai x_j = {1, -1}, j = 1, 2, …, N yang menghasilkan keluaran tunggal neuron y. Setiap *neuron input* terhubung ke *neuron output* dengan kekuatannya sesuai dengan besarnya parameter bobot $w_j \in$ [-1,1]. Ilustrasi dari model *Perceptron* diberikan pada Gambar 1.

Relasi antara *input-output* diatur oleh fungsi aktivasi berikut ini [13]:

$$\hat{y} = f\left(\sum_{j=1}^{N} w_j x_j - \theta\right) \quad (2)$$



Gambar 1 Ilustrasi model *Perceptron*

Bias $\theta \in$ [-1,1] dan $f(.)$ adalah fungsi aktivasi.

Algoritme pembelajaran *Perceptron* adalah:

1. Inisialisasi bobot w_j dan bias θ dengan bilangan yang sangat kecil.
2. Keluaran $\hat{y}$ diperoleh berdasarkan Persamaan (1).
3. *Update* bobot berdasarkan aturan:
   $w_k(t+1) = w_k(t) + \eta(y - \hat{y})x$
   dengan η adalah *learning rate*.
4. Jika $\hat{y} = y$, *Perceptron* tidak berubah. Jika $\hat{y} \neq y$, ulangi langkah 3 untuk melanjutkan *training* dan *learning*.

B. Support Vector Machine

Diberikan vektor masukan $\mathbf{x}_i \in \mathrm{R}^N$ dengan i = 1, 2, …, N dan berkaitan dengan label $y_i \in$ {+1, -1} dengan i = 1, 2, …, N. Nilai dari label +1 dan -1 masing-masing mewakili kelas positif dan negatif. Fungsi keputusan f berikut ini menyatakan hubungan *input-output* dalam model SVM [14],[15],[16]:

$$f(x_i) = \mathbf{w}^T \varphi(x_i) + b \; ; \; i = 1, 2, \ldots, N \; ; \; b \in \mathrm{R} \quad (3)$$

Di mana $\mathbf{w}$ menyatakan vektor bobot, b adalah parameter bias, f merupakan kelas prediksi, dan jika $f(\mathbf{x}) = 0$ maka merupakan *hyperplane* pemisah. Untuk data yang terpisah secara linier, maka fungsi transformasi $\varphi(x_i) = \mathbf{x}$. Karena itu, Persamaan (3) bisa ditulis menjadi Persamaan (4) berikut ini [17]:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \quad (4)$$

Untuk i = 1, 2, …, N berlaku:

$$\mathbf{w}^T \mathbf{x}_i + b \begin{cases} > 0 \text{ untuk } y_i = 1 \\ < 0 \text{ untuk } y_i = -1 \end{cases} \quad (5)$$

Data *training* terpisah secara linier sehingga tidak ada data training yang memenuhi $\mathbf{w}^T \mathbf{x}_i + b = 0$. Untuk mengatur pemisahan, pertidaksamaan (5) diganti menjadi pertidaksamaan berikut ini:

$$\mathbf{w}^T \mathbf{x}_i + b \begin{cases} \geq 1 \text{ untuk } y_i = 1 \\ \leq -1 \text{ untuk } y_i = -1 \end{cases} \quad (6)$$

Bentuk sederhana dari pertidaksamaan (6) adalah,

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad (7)$$

dengan i = 1, 2, …, N.

Bentuk umum dari:

$$\mathbf{w}^T \mathbf{x}_i + b = c \; ; \; -1 < c < 1 \quad (8)$$

merupakan persamaan *hyperplane* yang memisahkan $\mathbf{x}_i$ ke dalam kelas yang berbeda. *Hyperplane* yang memisahkan dua kelas ini dicari sedemikian hingga *hyperplane* tersebut memaksimalkan *margin* (*optimal hyperplane*). *Margin* adalah jarak antara *hyperplane* dan titik data terdekat dari setiap kelas. Untuk mendapatkan optimal *hyperplane*, maka parameter model SVM yang harus dioptimalkan adalah vektor bobot $\boldsymbol{w}$ dan b.

Bentuk formulasi model optimasi untuk mendapatkan optimal *hyperplane* adalah:

$$\text{Minimasi } Q(\mathbf{w},b) = \frac{1}{2}\|w\|^2 \tag{9}$$

Dengan fungsi kendala seperti Persamaan (7), maka *optimal hyperplane* diperoleh dengan menyelesaikan Persamaan (9) tersebut. Solusi Persamaan (9) dapat diperoleh dengan mentransformasikannya ke dalam persamaan *Lagrange* dan mencari titik *saddle*-nya dengan menurunkan secara parsial dan mengaplikasikannya dalam teorema *dual*-nya. Bentuk akhir transformasi masalah optimasi (9) diberikan berikut ini:

$$\text{Maksimasi } Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j \tag{10}$$

Dengan kendala $\sum_{i=1}^{N} y_i\alpha_i = 0 \;; \alpha_i \geq 0$ untuk i = 1, 2, …, N, di mana $\alpha_i\alpha_j$ merupakan pengali *Lagrange*. Solusi dari Persamaan (11) adalah pengali *Lagrange* yang memaksimalkan fungsi *Lagrange* Q. Dengan demikian, fungsi keputusan bisa dinyatakan dengan:

$$f(x) = \sum_{i\in S}\alpha_i y_i x_i^T x + b \tag{11}$$

C. Genetic Algorithm (GA)

Genetic Algorithm (GA) merupakan algoritme yang termasuk dalam kategori algoritme evolusi dan pertama kali diperkenalkan pada tahun 1975 oleh Holland. Metode ini digunakan sebagai solusi dari suatu pencarian. Menurut [17], algoritme ini juga menganut konsep Carles Darwin bahwa individu yang kuatlah yang akan bertahan di populasinya sesuai dengan teori evolusi. Prosedur pencarian GA didasarkan pada nilai fungsi tujuan saja dan tidak menggunakan teknik gradien maupun kalkulus.

Dalam GA, sebuah individu disebut kromosom. Setiap kromosom mewakili setiap vektor solusi. Dari beberapa kromosom tersebut terbentuklah sebuah populasi di mana tiap kromosom mewakili satu vektor solusi. Dengan adanya pembangkitan populasi ini, maka kita memiliki beberapa atau banyak solusi pilihan. Terdapat beberapa istilah dalam GA, yaitu kromosom, fungsi *fitness*, dan *crossover*.

Untuk mengukur akurasi kesesuaian (*fitness*) suatu solusi terhadap suatu permasalahan digunakan suatu fungsi yaitu fungsi *fitness*. Fungsi *fitness* terhubung langsung dengan fungsi tujuan. Ketika suatu solusi dievaluasi menggunakan fungsi *fitness*, maka perlu dilakukan seleksi pada kromosom untuk memilih kromosom anggota populasi lainnya.

Dalam metode GA terdapat istilah *crossover*. *Crossover* terbagi menjadi 2 teknik, yaitu *crossover* sederhana dan *crossover* aritmatik. Konsep *crossover* sederhana adalah jika terdapat 2 induk P_1 dan P_2, maka akan menghasilkan 2 keturunan, C_1 dan C_2. Jika anggota induk $= [x_1, x_2, .., x_n]$, $y = [y_1, y_2, .., y_n]$, dan r adalah bilangan random diskrit dengan nilai 1 dan panjang vektor x, maka U dan V yang mewakili turunan C_1 dan C_2 dapat didefinisikan sebagai berikut:

$$u_i = \begin{cases} x_i & i < r \\ y_i & \text{sebaliknya} \end{cases} \tag{12}$$

$$v_i = \begin{cases} x_i & i < r \\ y_i & \text{sebaliknya} \end{cases} \tag{13}$$

Keturunan dalam *crossover* aritmatik dihasilkan dengan melakukan kombinasi linier dari vektor induk yang dapat dituliskan seperti persamaan (14) dan (15).

$$C_1 = \lambda_1 X + \lambda_2 Y \tag{14}$$

$$C_2 = \lambda_2 X + \lambda_1 Y \tag{15}$$

Persamaan (14) dan (15) memiliki syarat mutasi di mana $\lambda_1 + \lambda_2 = 1$. *Crossover* aritmatik banyak diterapkan untuk suatu masalah di mana variabel keputusan memiliki kontinu. Selain itu, parameter *crossover rate* juga memiliki peran penting. Jika *crossover rate* kecil, maka hanya sedikit kromosom yang mengalami *crossover*. Sebaliknya, jika nilai *crossover* besar, maka semakin banyak kromosom yang mengalami *crossover*. Misalkan *crossover rate* yang dihasilkan adalah 0,5, maka hanya 50% dari jumlah kromosom yang mengalami *crossover*.

D. Particle Swarm Optimization (PSO)

Pencarian solusi algoritme PSO dilakukan pada suatu populasi yang terdiri dari kumpulan partikel. Setiap partikel memiliki posisi dan lokasi dari permasalahan yang akan diselesaikan. Setiap partikel melakukan pencarian solusi dengan melakukan penyesuaian berdasarkan posisi terbaik dari partikel (*local best*) dan penyesuaian posisi partikel terbaik dari seluruh kawanan (*global best*). Pada setiap iterasi, solusi yang dihasilkan partikel akan dievalusi dengan memasukkan solusi tersebut ke dalam *fitness function*. Setiap partikel dimisalkan sebagai sebuah titik pada dimensi ruang tertentu, kemudian diberikan status partikel pada ruang pencarian, yaitu posisi X dan kecepatan Y.

Persamaan (16) dan (17) merupakan formula yang menggambarkan posisi (X) dan kecepatan partikel (V) pada dimensi ruang N, di mana i adalah indeks partikel dan t adalah iterasi.

$$X_i(t) = x_{i1}(t), x_{i2}(t), x_{i3}(t), \ldots, x_{iN}(t) \tag{16}$$

$$V_i(t) = v_{i1}(t), v_{i2}(t), v_{i3}(t), \ldots, v_{iN}(t) \tag{17}$$

Persamaan (18) dan (19) adalah persamaan yang merepresentasikan mekanisme perbaikan status partikel. Persamaan (18) digunakan untuk menentukan kecepatan partikel baru berdasarkan kecepatan sebelumnya, jarak antara posisi sekarang dengan posisi terbaik partikel (*local best*), dan jarak saat ini dengan posisi terbaik populasi (*global best*). Kemudian partikel berpindah menuju posisi yang didasarkan persamaan (19).

$$V_i(t) = V_i(t-1) + c_1 r_1 \left(X_i^L X_i(t-1) \right) + c_2 r_2 \left(X^G X_i(t-1) \right) \quad (18)$$

$$X_i(t) = V_i(t) + X_i(t-1) \quad (19)$$

Dengan,

- $X_i^L = X_{i1}^L, X_{i2}^L, \ldots, X_{iN}^L$ adalah *local best* partikel ke-*i*,
- $X^G = X_1^G, X_2^G, \ldots, X_N^G$ adalah *global best* dari seluruh populasi,
- c_1 dan c_2 adalah konstanta positif (*learning factor*),
- r_1 dan r_2 adalah bilangan random positif antara 0 dan 1.

E. *Mean Absolute Deviation*

Mean Absolute Deviation (MAD) merupakan sebuah pengukuran *error data actual* dengan hasil estimasi. Persamaan (20) merupakan persamaan dasar menghitung MAD, di mana *N* merupakan jumlah data, γ merupakan data actual, dan $\hat{y}$ adalah nilai estimasi.

$$\text{MAD} = \frac{1}{N} \sum_{i=1}^{N} \text{abs}(\gamma_i - \hat{\gamma}_i) \quad (20)$$

F. *Dataset Iris Flower*

Dataset Iris Flowers digunakan sebagai *dataset* dalam penelitian ini. Data diperoleh dari situs resmi UCI Machine Learning [18]. Data ini mempunyai tiga kelas: *setosa*, *versicolor*, dan *virginica*. Visualisasi data ini dapat dilihat pada Gambar 2. Pada data ini terdapat 4 fitur: *Sepal Length* (f_1), *Sepal Width* (f_2), *Petal Length* (f_3), dan *Petal Width* (f_4). Karena fokus penelitian ini adalah pada klasifikasi biner, maka yang digunakan adalah data dengan kelas *versicolor* dengan warna hijau dan *virginica* dengan warna biru. Kelas *sentosa* tidak dimasukkan dalam penelitian ini karena secara visualisasi data warna merah sudah dapat terpisah secara sempurna dengan data yang lain. *Dataset* ini pernah dilakukan uji klasifikasi dengan mendapatkan hasil terbaik sebesar 99,85% [19]. *Dataset* ini mempunyai jumlah data sebanyak 150 data. Pada penelitian ini pembagian data uji dan data latih menggunakan *K-Fold Crossvalidation,* dengan nilai *K* adalah 5, sehingga pembagian data latih dan data uji adalah sebanyak 120 data dan 30 data untuk setiap uji coba.

G. *Skenario Uji Coba Penelitian*

Secara umum, penelitian ini terbagi dalam 3 tahap yaitu proses pelatihan, proses uji coba, dan pengukuran kinerja seperti yang ditunjukkan Gambar 3.

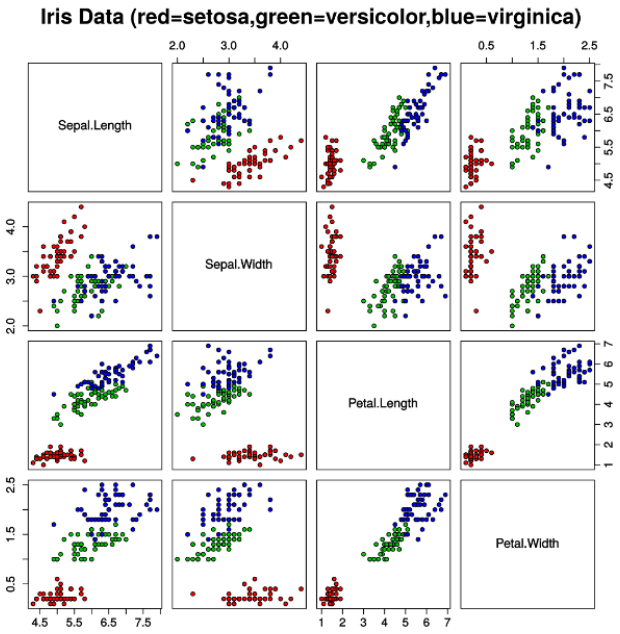
Proses pelatihan dilakukan dengan skema seperti Gambar 4. Tujuan dari proses ini adalah untuk pembentukan model. Pada Gambar tersebu terlihat bahwa setiap teknik klasifikasi (*Perceptron* dan SVM) digabungkan dengan masing-masing metode *optimization* (GD, PSO, dan GA). Penggunaan metode *optimization* dilakukan untuk pembentukan model dengan mendapatkan nilai parameter *w* dan *b* sebagai masukan dari *hyperplane* pada persamaan (1).

Setelah didapatkan pembentukan model, langkah selanjutnya adalah proses uji coba. Proses uji coba dilakukan dengan *dataset* Iris Flower. Selanjutnya, hasil uji coba setiap pengujian diukur kinerjanya berdasarkan *Mean Absolute Deviation* (MAD).

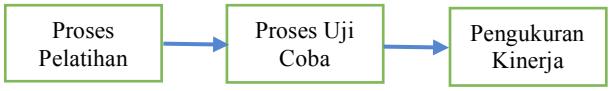
III. HASIL DAN PEMBAHASAN

Pada penelitian ini seluruh uji coba dilakukan dengan membagi dua proses yaitu proses pelatihan dan proses uji coba. Proses pelatihan bertujuan untuk mendapatkan model *hyperplane*. Model *hyperplane* terbaik diukur berdasarkan rumus kesalahan yang diberikan Persamaan (20). Model *hyperplane* terbaik kemudian diuji coba untuk mengetahui kinerja dari metode *Perceptron* dan metode SVM yang dioptimalkan menggunakan algoritme GD, GA, dan PSO.

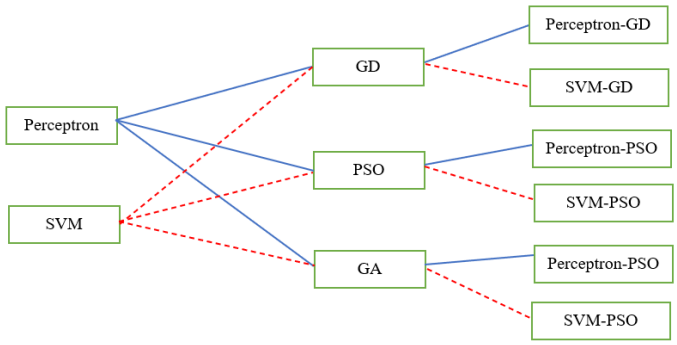
Parameter maksimum iterasi (*epoch*) dibagi menjadi dua, yaitu 50 iterasi digunakan pada algoritme PSO dan GA dan 500 iterasi untuk algoritme GD. Nilai parameter maksimum iterasi GD lebih banyak karena GD merupakan *local search*



Gambar 2 Visualisasi *dataset* Iris Flower [18]



Gambar 3 Alur penelitian



Gambar 4 Proses pelatihan

heuristic optimization yang hanya bekerja dengan satu individu. Selain parameter maksimum iterasi, GD juga dipengaruhi oleh parameter *learning rate*.

Tabel I merupakan hasil proses *training* metode *Perceptron*-GD. Percobaan dilakukan dengan 5 kali perulangan dengan 3 nilai *Lr* yang berbeda. Hasil optimal diambil berdasarkan nilai MAD terkecil. Berdasarkan nilai MAD, hasil klasifikasi terbaik metode *Perceptron*-GD diperoleh dengan nilai *learning rate* terbaiknya adalah sebesar 0,0001 dengan 7 data *misclass*.

Tahap pelatihan metode *Perceptron*-GA dan *Perceptron*-PSO diberikan pada Tabel II. Jumlah individu yang digunakan sebagai pengujian adalah 10 individu dan 25 individu. Tabel II melaporkan bahwa pada metode *Perceptron*-GA jumlah individu yang memberikan hasil optimal adalah 25 individu. Sementara itu, metode *Perceptron*-PSO mencapai optimal dengan jumlah individu sebanyak 10. Selain nilai MAD, perbandingan metode *Perceptron*-GA dan *Perceptron*-PSO bisa dilihat pada rata-rata data *misclass*. Pada uji coba *Perceptron*-PSO sendiri jumlah data yang *misclass* terkecilnya mencapai 6 data sedangkan perceptron GA mendapatkan hasil klasifikasi terbaiknya dengan 7 data yang *misclass*. Proses iterasi hasil terbaik dari *Perceptron* dengan masing-masing metode *optimization* diberikan pada Gambar 5.

Gambar 5 memaparkan konvergensi metode GD, GA, dan PSO untuk *Perceptron*. Kecepatan konvergensi ketiga metode *optimization* tersebut berkisar pada iterasi ke 25. Namun, jika dibandingkan hasilnya, PSO memberikan hasil yang terbaik dengan nilai MAD terkecil.

Garis pemisah pada Gambar 6 merupakan *hyperplane* terbaik dari ketiga metode *optimization*. *Hyperplane* merupakan proyeksi bobot *w* dan bias *b* yang didapatkan dari proses pelatihan. Arah *hyperplane* yang berbeda didapatkan dari GD, sedangkan PSO dan GA mempunyai arah yang sama.

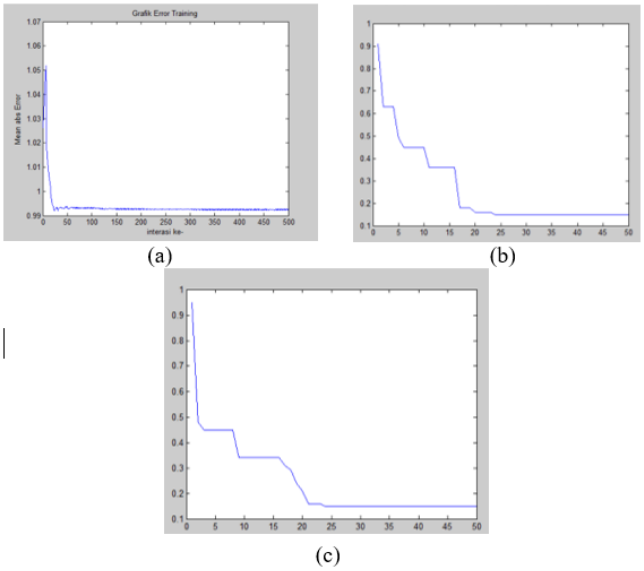
Pengujian yang sama dengan *Perceptron* akan juga dilakukan pada uji coba dengan metode SVM. Perbedaan utama yang telah dipaparkan pada bagian Pendahuluan adalah pada pengukuran proses pelatihan. Pengukuran *Perceptron* mengunakan selisih target aktual dengan nilai estimasi, maka SVM menggunakan persamaan (10).

TABEL I
HASIL UJI METODE *PERCEPTRON*-GD

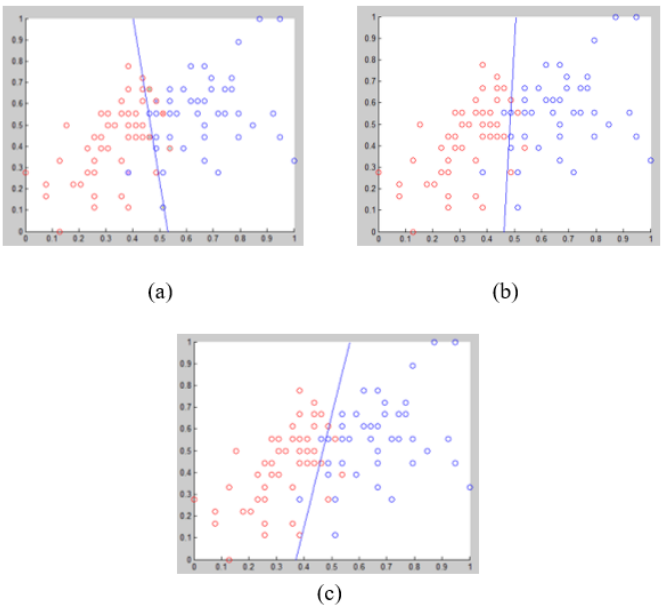
Percobaan ke-	*Lr* = 0,01		*Lr* = 0,001		*Lr* = 0,0001	
	MAD	***Mis class***	**MAD**	***Mis class***	**MAD**	***Mis class***
1	0,9274	7	0,936	17	0,8981	7
2	0,9298	7	0,923	17	0,9077	8
3	0,9262	7	0,954	16	0,9957	7
4	0,9077	7	0,939	14	0,8632	19
5	0,9322	7	0,951	18	0,9027	15
Rerata	**0,9246**		**0,941**		**0,9134**	

TABEL II
HASIL UJI METODE *PERCEPTRON* - GA DAN PSO

Per-cobaan ke-	GA				PSO			
	10 individu		25 individu		10 individu		25 individu	
	MAD	***Mis class***	**MAD**	***Mis class***	**MAD**	***Mis class***	**MAD**	***Mis class***
1	0,16	7	0,17	8	0,15	6	0,15	7
2	0,16	7	0,15	7	0,17	8	0,15	7
3	0,15	7	0,16	7	0,15	7	0,16	8
4	0,23	12	0,15	7	0,14	7	0,16	6
5	0,17	8	0,16	7	0,14	7	0,16	7
Rerata	**0,174**	**8,2**	**0,158**	**7,2**	**0,15**	**7**	**0,15**	**7**



Gambar 5 Grafik pelatihan *Perceptron*: (a) GD; (b) GA; (c) PSO



Gambar 6 *Hyperplane Perceptron*: (a) GD; (b) GA; (c) PSO

Parameter jumlah iterasi maksimum optimasi SVM menggunakan GD diberikan jumlah yang sama dengan *Perceptron*. Hasil percobaan mendapatkan *hyperplane* terbaik dengan metode SVM-GD diberikan pada Tabel III. Hasil *training* terbaik didapatkan dari parameter *Lr* = 0,003 dengan nilai rerata 2,99244, tetapi hasil data *misclass* yang diperoleh paling tinggi dibandingkan nilai *Lr* yang lain. Nilai *misclass* terbaik adalah 6 data *misclass*.

Tahap pelatihan metode SVM-GA dan SVM-PSO ditampilkan pada Tabel IV. Parameter jumlah individu pada proses *optimization* SVM oleh GA dan PSO juga ditentukan sebanyak 10 dan 25 indivindu. SVM-GA mencapai kondisi terbaik dengan 25 jumlah individu, sedangkan SVM-PSO cukup 10 individu untuk memperoleh hasil terbaik Hal ini dikarenakan ciri algoritme PSO adalah melakukan *update* data secara *gradient* dan berpusat ke satu individu terbaik *Gbest,* sedangkan GA melakukan *update* nilai dengan *crossover* dan mutase. Ketika *Gbest* ditentukan, maka semua individu akan dipusatkan di *Gbest*. Sesuai dengan data percobaan, ketika PSO menemukan solusi, maka tidak ada perubahan *update* nilai Gbest lagi. Hal ini berbanding terbalik dengan GA yang mempunyai proses mutasi (menjauhkan individu dengan solusi terbaik).

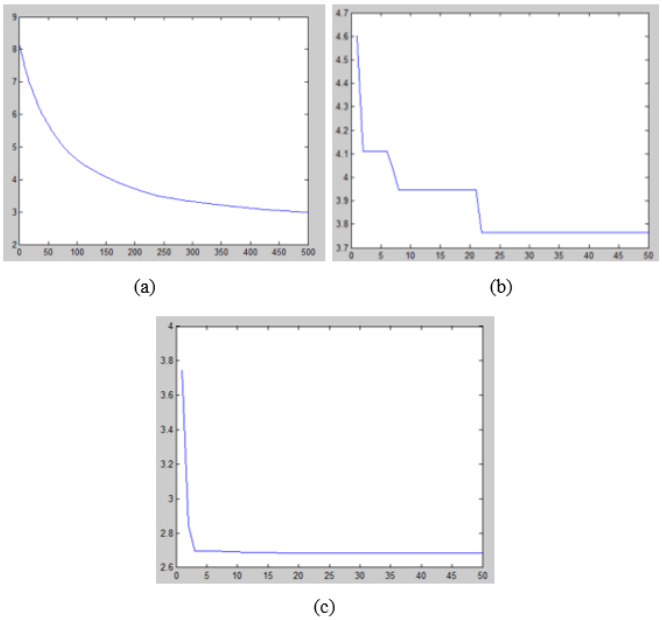
Pada setiap percobaan, model *hyperplane* SVM-PSO mendapatkan nilai yang sama sehingga standar deviasinya 0 dan nilai presisinya 100%. Proses pelatihan metode SVM dengan metode *optimization* diberikan pada Gambar 7. Konvergensi ketiga metode *optimization* jelas terlihat pada Gambar 7. Sampai pada iterasi ke-500, metode GD masih belum mampu mencapai akurasi metode *metaheuristic* GA dan PSO. Dengan batasan iterasi yang sama, PSO terlihat konvergen pada iterasi ke-16, sedangkan GA mulai konvergen pada iterasi ke-23.

TABEL III
HASIL UJI METODE SVM - GD

Percobaan ke-	*Lr* = 0,001		*Lr* = 0,003	
	Object function	*Mis class*	*Object function*	*Mis class*
1	4,0582	12	2,9881	6
2	3,9316	8	3,0018	8
3	4,086	11	2,9752	6
4	4,0453	11	3,002	19
5	4,0507	12	2,9951	15
Rerata	4,03436	10.8	2,99244	10,8

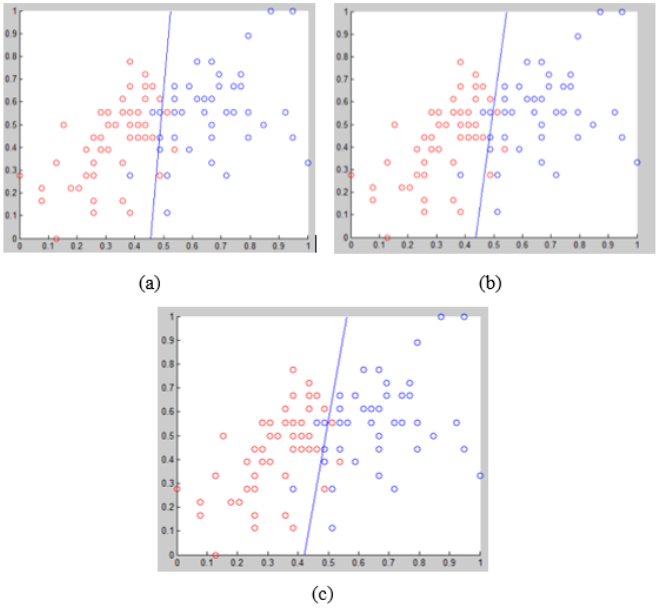
TABEL IV
HASIL UJI METODE SVM-GA DAN SVM-PSO

Per-cobaan ke-	GA				PSO			
	10 individu		25 individu		10 individu		25 individu	
	MAD	*Mis class*	MAD	*Mis class*	MAD	*Mis class*	MAD	*Mis class*
1	3,76	7	2,889	7	2,68	7	2,683	7
2	3,06	8	2,829	7	2,68	7	2,684	7
3	3,01	8	2,833	7	2,68	7	2,68	7
4	3,34	11	2,884	7	2,68	7	2,684	7
5	3,18	8	2,880	8	2,68	7	2,683	7
Rerata	3,27	8,4	2,863	7,2	2,68	7	2,684	7



Gambar 7 Grafik pelatihan SVM: (a) GD; (b) GA; (c) PSO

Visualisasi hasil klasifikasi SVM dengan optimasi GD, GA, dan PSO diberikan pada Gambar 8. Grafik *hyperplane* yang diperoleh dari ketiga *optimization* ini dengan metode SVM mendapatkan arah yang sama, seperti Gambar 8. Hal ini



Gambar 8 *Hyperplane* SVM: (a) GD; (b) GA; (c) PSO

berbeda dengan metode *Perceptron*. Pengukuran kesalahan jarak *margin* (Persamaan (20)) dan banyaknya *misclass* metode *Perceptron* dan SVM yang telah dioptimalisasi dengan GD, GA dan PSO dirangkum pada Tabel V.

Hasil penelitian ini mendukung penelitian terdahulu [19] [20], bahwa algoritme PSO mampu menemukan solusi terbaik atau bersifat *global optimal* dibandingkan dengan algoritme genetika. Pada penelitan sebelumnya, PSO digunakan untuk optimasi fungsi nonlinear dan PSO digunakan pada metode *Adaptive Neural Fuzzy Inference System*.

IV. SIMPULAN

Setelah melakukan uji coba setiap metode terhadap *K-Fold Crossvalidation* dengan nilai K = 5, maka kesimpulan dari penelitian ini adalah SVM dan *Perceptron* merupakan mesin pembelajaran yang menghasilkan garis pemisah (*hyperplane*) dengan baik. Pada *Perceptron*, *optimization* GD dan PSO memperoleh akurasi paling bagus, dan GA juga mendapatkan nilai akurasi yang kompetitif. Kesimpulan pada metode SVM, *metahuiristik optimization* (PSO dan GA) memperoleh hasil yang lebih kompetitif dibandingkan dengan optimasi GD. Secara umum *metahuristik optimization* lebih dapat bersinergi dengan *Perceptron* dan SVM.

Saran untuk penelitian selanjutnya berdasarkan penelitian ini adalah perlu dilakukan penelitian lebih lanjut untuk mengoptimalkan metode SVM dikarenakan nilai akurasi yang diperoleh paling tinggi dari metode SVM adalah 93%, sedangkan pada penelitian sebelumnya dengan menggunakan *dataset* nilai akurasi mencapai 98%. Hal ini dapat disebabkan beberapa factor. Faktor pertama adalah dapat mengantikan kernel dengan kernel nonlinear. Faktor kedua adalah dengan mencoba metode *optimization* yang lebih bagus dari metode optimasi-optimasi yang telah dilakukan.

TABEL V

PERBANDINGAN METODE

	Akurasi (%)
Perceptron-GD	93,0
Perceptron-GA	92,8
Perceptron-PSO	93,0
SVM-GD	89,2
SVM-GA	92,8
SVM-PSO	93,0

DAFTAR REFERENSI

[1] C. Cortes dan V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, hlm. 273-297, 1995.

[2] Y. Qiao, J. Peng, L. Ge dan H. Wang, "Application of PSO LS-SVM forecasting model in oil and gas production forecast," *2017 IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing*, Oxford, 2017, hlm. 470-474.

[3] S. Wei Fei dan X. Bin Zhang, "Fault diagnosis of power transformer based on support vector machine with genetic algorithm," *Expert Syst. Appl.*, vol. 36, no. 8, hlm. 11352–11357, 2009.

[4] K. Nguyen, T. Le, V. Lai, dan D. Nguyen, "Least square support vector machine for large-scale dataset," *2015 International Joint Conference on Neural Networks (IJCNN)*, 12-17 July 2015.

[5] G. Cheng, R. Guo, dan W. Wu, "Petroleum lithology discrimination based on PSO-LSSVM classification model," dalam *Proceedings of the 2010 Second International Conference on Computer Modeling and Simulation*, vol. 4, January 2010, hlm. 365-368.

[6] P. Samui dan T. Lansivaara, "Least square support vector machine applied to slope reliability analysis," *Geotechnical and Geological Engineering*, vol. 31, no. 4, hlm. 1329–1334, 2013.

[7] J. Platt, 1998. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. [Daring]. Tersedia: <https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/>.

[8] S. S. Shwartz, Y. Singer, N. Srebo, dan A. Cotter. "Pegasos: primal estimated sub-gradient solver for SVM." *Mathematical Programming*, vol. 127, hlm. 3-30, 2011.

[9] E. Teran, Z. Wang, dan D. A. Jimenez, "Perceptron learning for reuse prediction," dalam *MICRO-49: The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, no. 2, OCt 2016, hlm. 1-12.

[10] M. E. Lasulika, "Komparasi Naïve Bayes, *Support Vector Machine*, dan *K-Nearest Neighbor* untuk mengetahui akurasi tertinggi pada prediksi kelancaran pembayaran tv kabel," *Ilkom Jurnal Ilmiah*, vol. 11, no. 1, hlm. 11–16, 2019.

[11] Y. Muliono dan F. Tanzil, "A comparison of text classification methods k-NN, Naïve Bayes, and Support Vector Machine for news classification," *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 3, no. 2, hlm. 157–160, 2018.

[12] Suyanto, *Artificial Intelligence*. Bandung: Informatika Bandung, 2007.

[13] Z. Maojin, M. Z. W. Liu, P. Gao, Y. Wang, dan W. Yu, "A unitary weights based one-iteration quantum perceptron algorithm for non-ideal training sets," *IEEE Access*, vol. 7, hlm. 36854–36865, 2019.

[14] W. J. Niu, Z. K. Feng, B. F. Feng, Y. W. Min, C. T. Cheng, dan J. Z. Zhou, "Comparison of multiple linear regression, artificial neural network, extreme learning machine, and support vector machine in deriving operation rule of hydropower reservoir," *Water*, vol. 11, no. 1, 2019.

[15] L. L. Li, X. Zhao, M. L. Tseng, dan R. R. Tan, "Short-term wind power forecasting based on support vector machine with improved dragonfly algorithm," *Journal of Cleaner Production*, vol. 242, hlm. 118447, Jan 2020.

[16] K. Aoyagi, H. Wang, H. Sudo, dan A. Chiba, "Simple method to construct process maps for additive manufacturing using a support vector machine," *Additive Manufacturing*, vol. 27, hlm. 353–362, 2019.

[17] S. Abe, *Support Vector Machines for Pattern Classification*, Second Ed., London: Springer-Verlag London, 2010.
[18] UCI Machine Learning Repository, "Iris Data Set". [Daring]. Tersedia: https://archive.ics.uci.edu/ml/*dataset*s/iris
[19] M. Kurniawan dan N. Suciati, "Premise parameter optimization on adaptive network based fuzzyinference system using modification hybrid particle swarm optimization ad genetic algorithm," *Jurnal Iptek-Media Komunikasi Teknologi*, vol. 22, no. 2, hlm. 27–34, 2018.
[20] M. Kurniawan dan N. Suciati, "Modifikasi kombinasi *particle swarm optimization* dan *genetic algorithm* untuk permasalahan fungsi non-linier," *Integer: Journal of Information Technology*, vol. 2, no. 2, hlm. 31–40, 2017.

Muchamad Kurniawan. Gelar Sarjana Komputer diperoleh di Institut Adhi Tama Surabaya (ITATS) dan gelar Magister Komputer didapatkan di Institut Teknologi Sepuluh November (ITS). Konsentrasi bidang minat yang ditekuni *optimization* dan mesin pembelajaran.
Maftahatul Hakimah. Gelar Sarjana Sains diperoleh di Institut Teknologi Sepuluh Nopember Surabaya (ITS) dan Magister Sains didapatkan di Institut Teknologi Sepuluh November (ITS). Konsentrasi bidang minat yang ditekuni *optimization* dan prediksi.
Siti Agustini. Gelar Sarjana Sains Terapan diperoleh di Politeknik Elektronika Negeri Surabaya (PENS) dan Magister Teknik didapatkan dari Institut Teknologi Sepuluh Nopember (ITS). Konsentrasi bidang minat yang ditekuni: jaringan komputer, telekomunikasi, dan sekuriti jaringan.