BAB 2
LANDASAN TEORI

2.1 Game

Game adalah suatu aktifitas yang tidak dilakukan dengan sungguh sungguh [4]. Pengertian *Game* menurut beberapa ahli [5] diantaranya adalah sebagai berikut:

a. Menurut Ivan C Sibero, *Game* merupakan salah satu aplikasi yang paling banyak dipakai dan dinikmati oleh pengguna media elektronik saat ini.
b. Menurut Samuel Henry *Game* adalah sesuatu yang tidak bisa dipisahkan dari keseharian anak anak.
c. Menurut Fauzia A *Game* adalah salah satu bentuk hiburan yang dapat dijadikan sebagai penyegar pikiran dari kepenatan akibat dari padatnya aktivitas sehari hari.
d. Menurut John Naisbitt *Game* merupakan sistem partisipatoris dinamis karena *Game* memiliki tingkat penceritaan yang tidak dimiliki film.
e. Menurut Wiaya Ariyana dan Arifianto *Game* adalah salah satu kebutuhan sekaligus menjadi masalah besar bagi para pengguna komputer, hal tersebut dikarenakan untuk dapat menjalankan suatu *Game* dengan nyaman, maka dibutuhkan spesifikasi komputer yang relatif tinggi dibanding komputer untuk penggunaan biasa.

Dari beberapa pendapat yang dikemukakan oleh para ahli , dapat disimpulkan bahwa *Game* adalah sebuah kebutuhan untuk menghibur dari kepenatan dan tidak dapat dipisahkan dari keseharian anak anak.

2.1.1 Action-RPG

Action-Role Playing Game atau ARPG adalah bentuk sebuah subgenre yang didefinisikan dari *role-playing video game* yang menggabungkan unsur-unsur dari tindakan atau action-adventure game, menekankan tindakan real-time di mana pemain memiliki kontrol langsung pada karakter, bukan turn-based atau berbasis menu tempur. Permainan ini sering menggunakan sistem tempur yang mirip dengan *hack and slash* atau *game shooter*.

2.2 Taktik Infanteri Romawi

Taktik *infanteri* Romawi mengacu pada teori dan sejarah penyebaran, pembentukan dan manuver dari infanteri Romawi dari awal Republik Romawi jatuhnya Kekaisaran Romawi Barat. Artikel pertama menyajikan gambaran singkat dari pelatihan Romawi. Kinerja Roma terhadap berbagai jenis musuh kemudian dianalisa. Akhirnya penjumlahan dari apa yang membuat taktik Romawi dan strategi militer yang efektif melalui sejarah panjang mereka diberikan di bawah ini, seperti diskusi tentang bagaimana dan mengapa efektifitas ini akhirnya menghilang. Pada tingkat infanteri, Tentara Romawi disesuaikan senjata baru: The *Pilum* (lembing khusus), *gladius* (pedang menyodorkan pendek), dan perisai cembung baru (untuk perlindungan yang lebih baik terhadap tombak) yang membuka phalanx sementara masih memberikan perlindungan. Umumnya, pertempuran dibuka dengan tendangan voli dari tombak cahaya dari sampai dengan 18 m (20 yd) (dan sering jauh lebih sedikit), diikuti oleh tembakan dari tombak berat (pila). Berikut tembakan tersebut, tentara Romawi akan menutup musuh mereka untuk terlibat dengan gladi.

Ada beberapa perintah pada saat dalam peperangan, formasi yang berbeda diasumsikan sesuai dengan situasi taktis yang berbeda berikut adalah beberapa perintah yang di berikan :

a. *Equites Repellere* ("mengusir pasukan berkuda / ksatria") adalah pembentukan digunakan untuk melawan kavaleri. Legionaries akan menganggap formasi persegi, memegang pila mereka sebagai tombak di ruang antara perisai dan dirangkai bahu.
b. Pada perintah *pila iacere,* legionaries melemparkan pila mereka di musuh.
c. Pada perintah format *cuneum,* infanteri membentuk baji untuk mengisi dan istirahat garis musuh. Formasi ini digunakan sebagai taktik kejutan.
d. Pada perintah *contendite vestra sponte, legionaries* diasumsikan sikap agresif dan menyerang setiap lawan yang mereka hadapi.
e. Pada perintah format *orbem,* legionaries diasumsikan formasi lingkaran-seperti dengan pemanah ditempatkan di tengah-tengah dan belakang

legiun menyediakan dukungan api rudal. Taktik ini digunakan terutama ketika sejumlah kecil legiun harus menahan posisi dan dikelilingi oleh musuh.

f. Pada perintah *frontem ciringite*, legionaries memegang posisi mereka.
g. Pada perintah *allargate frontem*, formasi tersebar diadopsi.
h. Pada perintah format *testudinem, legionaries* diasumsikan formasi *Testudo* (kura-kura). Formasi tersebut adalah bergerak lambat tapi hampir tak tertembus tembakan musuh, dan dengan demikian sangat efektif selama pengepungan dan / atau ketika menghadapi pemanah musuh. Namun pembentukan *Testudo* tidak memungkinkan untuk pertempuran jarak dekat yang efektif dan oleh karena itu digunakan ketika musuh yang cukup jauh sehingga sebagai legiun bisa masuk ke formasi lain sebelum diserang.
i. Pada perintah *tecombre, legiun* akan mematahkan formasi *Testudo* dan kembali ke formasi mereka sebelumnya.
j. Pada perintah *Agmen* format, *legionaries* diasumsikan formasi persegi, yang juga merupakan bentuk khas dari satu abad dalam pertempuran.

2.3 Kecerdasan Buatan

Kecerdasan buatan (*Artificial Intelligence*) atau sering disebut *AI* adalah suatu ilmu yang mempelajari cara membuat komputer melakukan sesuatu seperti yang dilakukan oleh manusia [6]. Kecerdasan buatan memiliki penafsiran yang berbeda beda oleh para ahli namun memiliki makna yang sama, berikut adalah pengertian kecerdasan buatan menurut para ahli [6]:

a. Menurut H. A. Simon menjelaskan bahwa kecerdasan buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas.
b. Menurut Rick and Knight menjelaskan bahwa Kecerdasan buatan (*Artificial Intelligence*) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan

lebih baik oleh manusia, dengan tujuan utama dari kecerdasan buatan itu sendiri adalah membuat mesin menjadi lebih pintar.

2.3.1 Jaringan Syaraf Tiruan

Jaringan syaraf adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

Jaringan Syaraf Tiruan (JST) merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik menyerupai jaringan syaraf biologi (JSB). JST tercipta sebagai suatu generalisasi model matematis dari pemahaman manusia (human cognition) yang didasarkan atas asumsi sebagai berikut [7]:

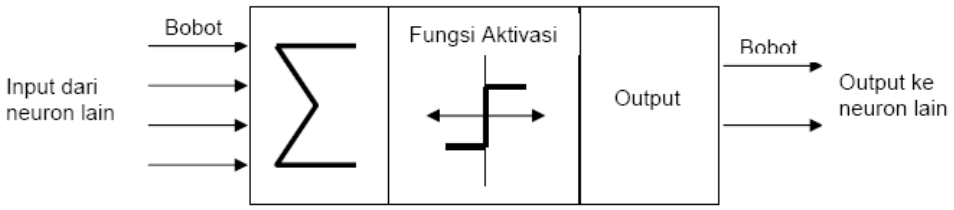
1. Pemrosesan informasi terjadi pada elemen sederhana yang disebut neuron
2. Sinyal mengalir diantara sel saraf/neuron melalui suatu sambungan penghubung
3. Setiap sambungan penghubung memiliki bobot yang bersesuaian. Bobot ini akan digunakan untuk menggandakan / mengalikan sinyal yang dikirim melaluinya.
4. Setiap sel syaraf akan menerapkan fungsi aktivasi terhadap sinyal hasil penjumlahan berbobot yang masuk kepadanya untuk menentukan sinyal keluarannya.

Analogi jaringan syaraf tiruan dengan jaringan syaraf buatan dijelaskan pada Tabel II.1 berikut ini:

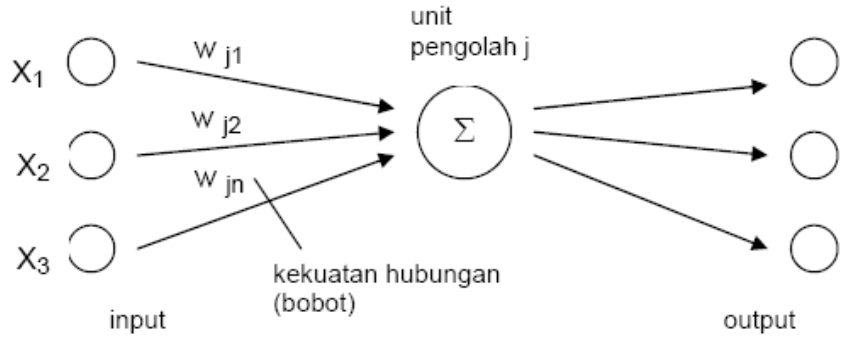
Tabel 2.1 Analogi JST dan JSB

JST	JSB
Node/masukan	Badan sel(soma)
Masukan	Dendrit
Keluaran	Akson
Bobot	Sinapsis

Model struktur neuron jaringan syaraf tiruan dijelaskan pada Gambar 2.1 dan Gambar 2.2.



Gambar 2.1 Model Struktur JST



Gambar 2.2 Model Struktur JST

Jaringan syaraf tiruan dapat belajar dari pengalaman, melakukan generalisasi atas contoh-contoh yang diperolehnya dan mengabstraksi karakteristik esensial masukan bahkan untuk data yang tidak relevan. Algoritma untuk JST beroperasi secara langsung dengan angka sehingga data yang tidak numerik harus diubah menjadi data numerik. JST tidak diprogram untuk menghasilkan keluaran tertentu. Semua keluaran atau kesimpulan yang ditarik oleh jaringan didasarkan pada pengalamannya selama mengikuti proses pembelajaran. Pada proses pembelajaran, ke dalam JST dimasukkan pola-pola masukan (dan keluaran) lalu jaringan akan diajari untuk memberikan jawaban yang bisa diterima. Pada dasarnya karakteristik JST ditentukan oleh :

1. Pola hubungan antar neuron (disebut arsitektur jaringan)
2. Metode penentuan bobot-bobot sambungan (disebut dengan pelatihan atau proses belajar jaringan)
3. Fungsi aktivasi

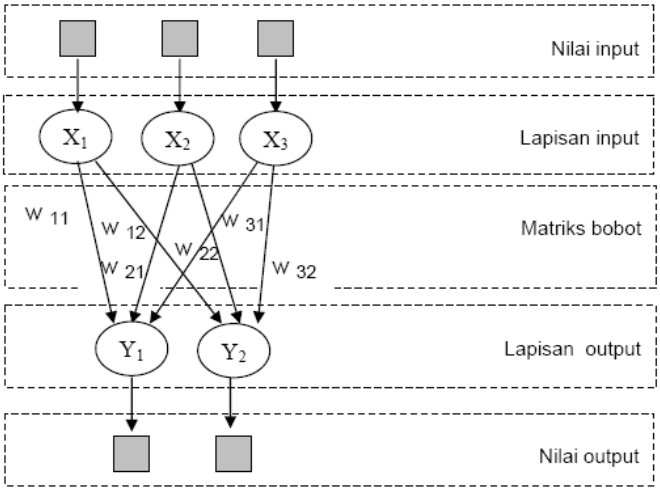
2.3.2 Arsitektur Jaringan Syaraf Tiruan

Pada JST, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (layer) yang disebut dengan lapisan neuron (neuron layers). Neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya. Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan masukan sampai ke lapisan keluaran melalui lapisan tersembunyi (hidden layer). Gambar berikut ini jaringan syaraf dengan 3 lapisan dan bukanlah struktur umum jaringan syaraf karena beberapa jaringan syaraf ada yang tidak memiliki lapisan tersembunyi.

Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Umumnya neuron-neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama sehingga pada setiap lapisan yang sama neuron-neuron memiliki fungsi aktivasi yang sama. Bila neuron-neuron pada suatu lapisan (misal lapisan tersembunyi) akan dihubungkan dengan neuron-neuron pada lapisan lain (misal lapisan keluaran) maka setiap neuron pada lapisan tersebut (lapisan tersembunyi) juga harus dihubungkan dengan setiap neuron pada lapisan lainnya (lapisan keluaran). Terdapat 3 macam arsitektur JST, yaitu [7]:

1. Jaringan dengan lapisan tunggal (single layer net)

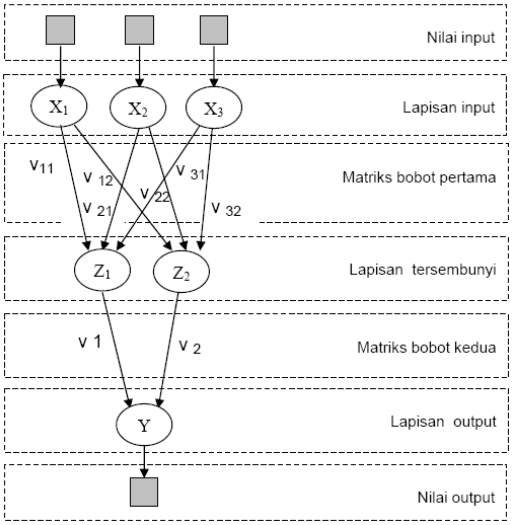
   Jaringan ini hanya memiliki 1 lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima masukan kemudian secara langsung akan mengolahnya menjadi keluaran tanpa harus melalui lapisan tersembunyi. Pada gambar berikut neuron-neuron pada kedua lapisan saling berhubungan Seberapa besar hubungan antara 2 neuron ditentukan oleh bobot yang bersesuaian. Semua unit masukan akan dihubungkan dengan setiap unit keluaran seperti terlihat pada Gambar 2.3.



Gambar 2.3 Jaringan Dengan Lapisan Tunggal

2. Jaringan dengan banyak lapisan (multilayer net)

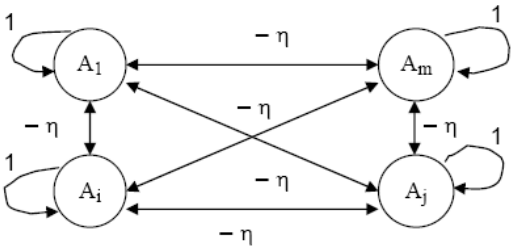
   Jaringan ini memiliki 1 atau lebih lapisan yang terletak diantara lapisan masukan dan lapisan keluaran. Umumnya ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan seperti terlihat pada Gambar 2.4. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit. Pada banyak kasus, pembelajaran pada jaringan dengan banyak lapisan ini lebih sukses dalam menyelesaikan masalah.



Gambar 2.4 Jaringan Dengan Banyak Lapisan

3. Jaringan dengan lapisan kompetitif (competitive layer net)

   Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Umumnya hubungan antar neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar 2.5 menunjukkan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot -n.



Gambar 2.5 Jaringan Dengan Lapisan Kompetitif

2.3.3 Fungsi Aktivasi

fungsi aktivasi adalah fungsi yang digunakan untuk mengaktifkan setiap neuron yang dipakai pada jaringan tersebut. Banyak fungsi yang dipakai sebagai pengaktif, diantaranya :

1. Fungsi Undak Biner (Hard Limit)

Jaringan dengan lapisan tunggal sering menggunakan fungsi undak untuk mengkonversi input dari suatu variable yang bernilai kontinu ke suatu output biner. Fungsi hard limit dirumuskan

$$Y = \begin{cases} 0, jika\ x \leq 0 \\ 1, jika\ x > 0 \end{cases} \quad (2.1)$$

2. Fungsi Undak Biner (Threshold)

   Fungsi ini sering disebut fungsi niali ambang atau fungsi Heaviside, dirumuskan :

$$Y = \begin{cases} 0, jika\ x < \theta \\ 1, jika\ x \geq \theta \end{cases} \quad (2.2)$$

3. Fungsi Bipolar Hampir sama dengan undak biner, hanya output yang dihasilkan berupa 1, 0 atau -1. Fungsi ini dirumuskan :

$$Y = \begin{cases} 1, jika\ x > 0 \\ 0, jika\ x = 0 \\ -1, jika\ x < 0 \end{cases} \quad (2.3)$$

4. Fungsi Bipolar (dengan Threshold)

   Fungsi yang menghasilkan output berupa 1, 0 atau -1

$$Y = \begin{cases} 1, jika\ x\ \geq \theta \\ -1, jika\ x < \theta \end{cases} \quad (2.4)$$

5. Fungsi Linear (identitas)

   Fungsi linear memiliki nilai output yang sama dengan nilai input, dirumuskan : y = x

6. Fungsi Sturating Linear

   Fungsi ini dirumuskan :

$$Y = \begin{cases} 1, jika\ x \geq 0 \\ x + 0{,}5\ , jika - o, 5 \leq x \leq 0{,}5 \\ 0, jika\ x \leq 0 \end{cases} \quad (2.5)$$

7. Fungsi Symetric Saturating Linear

   Fungsi ini dirumuskan :

$$Y = \begin{cases} 1, jika\ x \geq 0 \\ x\ , jika - 1 \leq x \leq 1 \\ -1, jika\ x \leq -1 \end{cases} \quad (2.6)$$

8. Fungsi Sigmoid Biner

   Digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode backpropagation. Fungsi ini dirumuskan :

$$y = f(x) = \frac{1}{(1+e^{\acute{a}x})} \quad (2.7)$$

9. Fungsi Sigmoid Bipolar

   Output dari fungsi ini memiliki range antara 1 sampai -1. Fungsi ini dirumuskan :

$$y = f(x) = \frac{1-e^{-x}}{(1+e^{-x})} \quad (2.8)$$

2.3.4 Learning Rate

Learning rate merupakan salah satu parameter training untuk menghitung nilai koreksi bobot pada waktu proses training. Nilai α ini berada pada range nol (0) sampai (1). Semakin besar nilai learning rate, maka proses training akan berjalan semakin cepat. Namun apabila nilai learning rate relatif terlalu besar, pada umumnya proses training dapat melampaui keadaan optimal yaitu pada saat dicapai nilai error yang paling minimal. Dengan kata lain, learning rate mempengaruhi ketelitian jaringan suatu sistem. Semakin besar learning rate, maka ketelitian jaringan akan semakin berkurang, tetapi berlaku sebaliknya, apabila learning rate-nya semakin kecil, maka ketelitian jaringan akan semakin besar atau bertambah dengan konsekuensi proses training akan memakan waktu yang semakin lama [8].

2.3.5 Min-Max Normalization

Normalisasi digunakan untuk menyamakan skala atribut data kedalam sebuah range yang spesifik, misal –1 s/d 1 atau 0 s/d 1. *Min-Max Normalization*

melakukan transformasi linier pada attribut data asli guna menghasilkan range nilai yang sama [9].

$$V' = \frac{V - min_a}{maxa - min_a} \quad (2.3)$$

dimana, *V'* : Nilai baru data hasil *Min-Max Normalization*

V : Nilai data yang akan dinormalisasi

min_A : Nilai minimum dari *field* data yang sama

max_A : Nilai maksimum dari *field* data yang sama

2.3.6 Proses Pembelajaran Jaringan

Cara belajar JST adalah sebagai berikut:

1. JST dimasukkan informasi yang sebelumnya telah diketahui hasil keluarannya.
2. Proses memasukkan informasi ini dilakukan lewat node-node atau unit-unit masukan. Bobot-bobot antarkoneksi dalam suatu arsitektur diberi nilai awal dan kemudian JST dijalankan. Bobot-bobot ini bagi jaringan digunakan untuk belajar dan mengingat suatu informasi. Pengaturan bobot dilakukan secara terus-menerus dan dengan menggunakan kriteria tertentu sampai diperoleh keluaran yang diharapkan.

Hal yang ingin dicapai dengan melatih/mengajari JST adalah untuk mencapai keseimbangan antara kemampuan memorisasi dan generalisasi. Kemampuan memorisasi adalah kemampuan JST untuk memanggil kembali secara sempurna sebuah pola yang telah dipelajari. Kemampuan generalisasi adalah kemampuan JST untuk menghasilkan respon yang bisa diterima terhadap pola-pola masukan yang serupa (namun tidak identik) dengan pola-pola yang sebelumnya telah dipelajari.

Hal ini sangat bermanfaat bila pada suatu saat ke dalam JST dimasukkan informasi baru yang belum pernah dipelajari, maka JST masih akan tetap dapat memberikan tanggapan yang baik, memberikan keluaran yang paling mendekati. Paradigma/metode pembelajaran/pelatihan JST adalah sebagai berikut [7]:

1. Pembelajaran terawasi (supervised learning)

   Pada pembelajaran ini kumpulan data masukan yang digunakan, data keluarannya telah diketahui. Perbedaan antara keluaran-keluaran aktual

dengan data keluaran yang diinginkan digunakan untuk mengoreksi bobot JST agar JST dapat menghasilkan jawaban sedekat (semirip) mungkin dengan jawaban yang benar yang telah diketahui oleh JST.

2. Pembelajaran tak terawasi (unsupervised learning)

   Pada pembelajaran ini, JST mengorganisasi dirinya sendiri untuk membentuk vektor-vektor masukan yang serupa, tanpa menggunakan data atau contoh-contoh pelatihan. Struktur menggunakan dasar data atau korelasi antara pola-pola data yang dieksplorasi. Paradigma pembelajaran ini mengorganisasi pola-pola ke dalam kategori-kategori berdasarkan korelasi yang ada.

3. Gabungan pembelajaran terawasi dan tak terawasi (hybrid)

   Merupakan kombinasi dari kedua pembelajaran tersebut. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi

2.3.6.1 Pembelajaran Tak Terawasi (Unsupervised Learning)

Metode pembelajaran tak terawasi merupakan metode pembelajaran yang tidak memerlukan target keluaran. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai masukan yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk pengelompokan pola [7].

2.3.6.2 Jaringan Kohonen

Jaringan kohonen pertama kali diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982. Pada jaringan ini, suatu lapisan yang berisi neuron-neuron akan menyusun dirinya sendiri berdasarkan masukan nilai tertentu dalam suatu kelompok yang dikenal dengan istilah pengelompokkan (*cluster*). Selama proses penyusunan diri, *cluster* yang memiliki vektor bobot paling cocok dengan pola masukan (memiliki jarak yang paling dekat) akan terpilih sebagai pemenang.

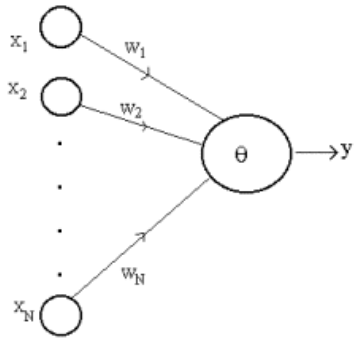
Neuron yang menjadi pemenang beserta neuron-neuron tetangganya akan memperbaiki bobotnya.

2.3.7 Metode Pembelajaran Terawasi (supervised learning)

Metode pembelajaran pada jaringan syaraf disebut terawasi jika keluaran yang diharapkan telah diketahui sebelumnya, misalkan seperti pada proses AND. Pada proses pembelajaran, satu pola masukan akan diberikan ke satu neuron pada lapisan masukan . Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan keluaran. Lapisan keluaran ini akan membangkitkan pola keluaran yang nantinya akan dicocokkan dengan pola keluaran targetnya. Apabila terjadi perbedaan antara pola keluaran hasil pembelajaran dengan pola target, maka akan muncul error. Apabila nilai error masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Metode-metode yang termasuk dalam pembelajaran terawasi antara lain:

2.3.7.1 Hebb rule

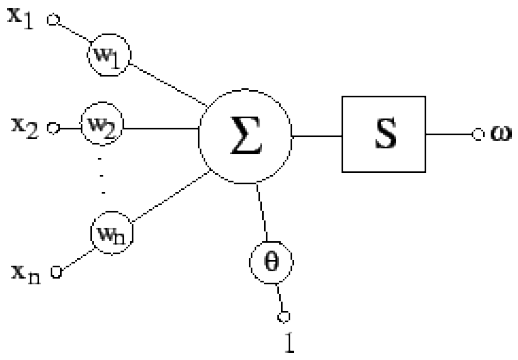
Hebb rule adalah metode pembelajaran yang paling sederhana. Pada metode ini pembelajaran dilakukan dengan cara memperbaiki nilai bobot sedemikian rupa sehingga jika ada 2 neuron yang terhubung, dan keduanya pada kondisi hidup‟ (*on*) pada saat yang sama, maka bobot antara keduanya dinaikkan. Gambar 2.6 merupakan gambar jaringan *hebb rule*.



Gambar 2.6 Bentuk Jaringan *Hebb Rule*

2.3.7.2 Perceptron

Jaringan jenis ini hanya terdiri dari *layer* masukan dan *layer* keluaran saja. Data masukan yang masuk melalui *vector* masukan akan langsung diproses dan kemudian ditentukan target keluaran dari hasil pengolahan. Jaringan *perceptron* dapat dilihat pada Gambar 2.7.

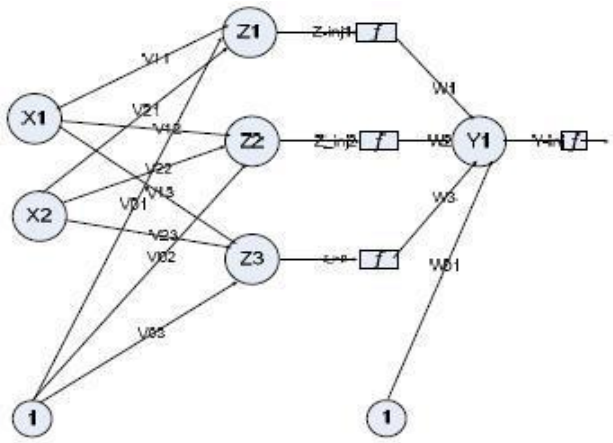


Gambar 2.7 Bentuk jaringan *Perceptron*

2.3.7.3 Backpropagation

Salah satu arsitektur jaringan saraf tiruan yang sangat populer adalah *multilayer feedforward networks*. Secara umum, jaringan seperti ini terdiri dari sejumlah unit neuron sebagai lapisan masukan, satu atau lebih lapisan simpul-simpul neuron komputasi lapisan tersembunyi, dan sebuah lapisan simpul-simpul neuron komputasi keluaran. Sinyal masukan dipropagasikan ke arah depan (arah lapisan keluaran), lapisan demi lapisan. Jenis jaringan ini adalah hasil generalisasi dari arsitektur *perceptron* satu lapisan, jadi biasa disebut sebagai *multilayer perceptron* (MLPs). *Error back propagation* adalah algoritma MLPs yang menggunakan prinsip pembelajaran terawasi. Propagasi balik (ke arah lapisan masukan) terjadi setelah jaringan menghasilkan keluaran yang mengandung *error*. Pada fase ini seluruh bobot *synaptic* (yang tidak memiliki aktivasi nol) dalam jaringan akan disesuaikan untuk mengkoreksi/memperkecil *error* yang terjadi (*error correction rule*). Untuk pelatihan jaringan, pasangan fase propagasi ke depan dan balik dilakukan secara berulang untuk satu set data latihan, kemudian diulangi untuk sejumlah *epoch* (satu sesi lewatan untuk seluruh data latihan dalam sebuah proses pelatihan jaringan) sampai *error* yang

terjadi mencapai batas kecil toleransi tertentu atau nol. Jaringan *backpropagation* dapat dilihat pada Gambar 2.8 [10].



Gambar 2.8 Bentuk Jaringan *Backpropagation*

Dari Gambar II.12 dapat dilihat bahwa yang bertindak sebagai dendrit adalah X1 dan X2, yaitu data masukan pada jaringan tersebut. Terdapat 2 sinapsis atau bobot yaitu V dan W, sedangkan Z dan Y merupakan bagian dari soma atau badal sel dari jaringan tersebut. Dan yang bertindak sebagai akson atau data keluaran adalah Y. Berikut ini algoritma pembelajaran metode backpropagation.

Langkah 0 :

Pemberian inisialisasi penimbang (diberi nilai kecil secara acak)

Langkah 1 :

Jika kondisi berhenti tidak tercapai, Ulangi langkah 2 hingga 9 sampai kondisi akhir iterasi dipenuhi(kondisi berhenti jika perulangan mencapai maksimal perulangan atau *mean square error (MSE)* sudah lebih kecil dari rasio pembelajaran)

Langkah 2 :

Untuk masing-masing pasangan data pelatihan (training data) lakukan langkah 3 hingga 8 Propagasi maju (Feedforward)

Langkah 3 :

Masing-masing unit masukan (X_i, I = 1,…,n) menerima sinyal masukan X_i dan sinyal tersebut disebarkan ke unit-unit bagian berikutnya (unit-unit lapisan tersembunyi)

Langkah 4 :

Masing-masing unit dilapisan tersembunyi dikalikan dengan faktor penimbang dan dijumlahkan serta ditambahkan dengan biasnya :

$$Z_{in_j} = V_{oj} + \sum_{I=1}^{n} X_i V_{ij} \quad (2.9)$$

Kemudian menghitung sesuai dengan fungsi aktifasi yang digunakan :

$$Z_j = f(Z_{in_j}) \quad (2.10)$$

Bila yang digunakan adalah fungsi sigmoid maka bentuk fungsi tersebut adalah :

$$Z_j = \frac{1}{1+exp^{-z_in_j}} \quad (2.11)$$

Masing-masing unit keluaran (Y_k, k=1,2,3,…m) dikalikan dengan faktor penimbang dan dijumlahkan :

$$Y_{in_k} = W_{ok} + \sum_{j=1}^{p} Z_j W_{jk} \quad (2.12)$$

Menghitung kembali sesuai dengan fungsi aktifasi Backpropagation dan galatnya

$$y_k = f(y_{in_k}) \quad (2.13)$$

Langkah 6 :

Masing-masing unit keluaran (Y_k, k=1,…,m) menerima pola target sesuai dengan pola masukan saat pelatihan dan dihitung galatnya :

$$\delta_k = y_k(1 - y_k) \quad (2.14)$$

Menghitung perbaikan faktor penimbang (kemudian untuk memperbaiki w_{jk}

$$\Delta W_{kj} = \alpha . \partial_k . Z_j \quad (2.15)$$

Menghitung perbaikan koreksi :

$$\Delta W_{0j} = \alpha . \partial_k \quad (2.16)$$

Dan menggunakan nilai δ_k pada semua unit lapisan sebelumnya.

Langkah 7 :

Masing-masing penimbang yang menghubungkan unit-unit l;apisan keluaran dengan unit-unit pada lapisan tersembunyi $(Z_j, j = 1, \dots, p)$ dikalikan delta dan dijumlahkan sebagai masukan ke unit-unit lapisan berikutnya.

$$\delta_in_j = \sum_{k=1}^{n} \delta_k W_{jk} \quad (2.17)$$

Selanjutnya dikalikan dengan turunan dari fungsi aktifasinya untuk menghitung galat.

$$\delta_j = \delta_in_j f'(y_in_j) \quad (2.18)$$

Kemudian menghitung perbaikan penimbang (digunakan untuk memperbaiki V_{ij})

$$\Delta V_{ij} = \alpha \delta_j X_i \quad (2.19)$$

Kemudian menghitung perbaikan bias (untuk memperbaiki V_{oj})

$$\Delta V_{oj} = \alpha \delta_j \quad (2.20)$$

Memperbaiki penimbang dan bias

Langkah 8 :

Masing-masing keluaran unit (y_k, k=1,…,m) diperbaiki bias dan penimbangnya (j=0,…,p)

$$W_{jk}(baru) = W_{jk}(lama) \Delta W_{jk} \quad (2.21)$$

Masing-masing unit tersembunyi (Z_j, j:1,…,p) diperbaiki bias dan penimbangnya (j=0,…,n)

$$V_{jk}(baru) = V_{jk}(lama)\Delta V_{jk} \quad (2.22)$$

Langkah 9 :

Uji kondisi pemberhentian (akhir iterasi) pengujian pemberhentian dilakukan dengan menghitung nilai *error*. Jika mencari nilai *error* yang tidak memiliki nilai target maka rumus mencari nilai *error* adalah sebagai berikut :

$$1 - (y)^2 \quad (2.23)$$

Jika sudah memiliki nilai target maka rumus mencari nilai *error* adalah sebagai berikut :

$$\frac{1}{2}\sum k\,(Tk - Yk)^2 \quad (2.24)$$

Tabel 2.2 Daftar Keterangan Notasi Backpropagation

Notasi	Keterangan
X^p	Pola masukan pelatihan ke-p, p=1,2,...,p<=1. (X_1,X_2,X_3, ...,X_n)
t^p	Pola keluaran target dari pelatihan. (t_1, t_2, t_3, ..., t_n)
xi	Unit ke-I pada lapisan masukan
Xi	Nilai aktifasi dari unit Xi
Z_j	Unit ke-j pada lapisan tersembunyi
Z_in_j	Keluaran untuk unit Z_j
z_j	Nilai aktifasi dari unit Z_j
Y_k	Unit ke-k pada lapisan keluaran
Y_in_k	Net masukan untuk unit Y_k
W_{k0}	Nilai penimbang sambungan pada bias untuk unit Y_k
W_{kj}	Nilai penimbang sambungan dari Z_{ij} ke unit Y_k
∇W_{kj}	Selisih $W_{kj}(t)$ dengan $W_{kj}(t+1)$
V_{io}	Nilai penimbang sambungan pada bias untuk unit Z_i
∇W_{ij}	Selisih antara $V_{ij}(t)$ dengan $V_{ij}(t+1)$

δ_k	Faktor pengaturan nilai penimbang sambungan pada lapisan keluaran
δ_j	Faktor pengaturan niloai penimbang sambungan pada lapisan tersembunyi
α	Konstanta laju pelatihan (learning rate) 0<α<1
E	Total galat

2.4 OOP (Object Oriented Programing)

Objek adalah kesatuan entitas yang memiliki sifat dan tingkah laku. Dalam kehidupan sehari-hari, objek adalah benda, baik benda berwujud nyata seperti manusia, hewan, mobil, komputer, handphone, pena, ataupun benda yang tidak nyata atau konsep, seperti halnya tabungan bank, sistem antrian, sistem internet banking, dan sebagainya. Jadi pengertian OOP adalah konsep yang membagi program menjadi objek-objek yang saling berinteraksi satu sama lain. Objek adalah benda, baik benda yang berwujud nyata maupun benda yang tidak nyata (konsep). Jika menggunakan OOP maka akan ada enam keuntungan yang dapat diperoleh [11], yaitu :

1. Alami (*Natural*).
2. Dapat diandalkan (*Reliable*).
3. Dapat digunakan kembali (*Reusable*).
4. Mudah untuk dalam perawatan (*Maintainable*).
5. Dapat diperluas (*Extendable*).
6. Efisiensi waktu.

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep OOP, adalah :

1. C++.
2. Visual C++.
3. Visual Basic.
4. Java.

2.4.1 Analisis Berorientasi Objek

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahap untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem berorientasi objek.

OOA biasanya menggunakan kartu CRC (*Component*, *Responsibility*, *Collaborator*) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (*Unifed Modeling Language*) pada bagian diagram use case, diagram kelas, dan diagram objek [11].

2.4.2 Desain Berorientasi Objek

Desain berorientasi objek atau Object Oriented Design (OOD) adalah tahapan perantara untuk spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasikan objek.

Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat lunak dengan menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (Unified Modeling Language). Kendala dan permasalahan pembangunan sistem berorientasi objek biasanya dapat dikenali dalam tahap ini [11].

2.4.3 Pemodelan

Pemodelan adalah gambaran dari realita yang sederhana dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. Pemodelan dapat menggunakan bentuk yang sama dengan realitas. Pemodelan juga banyak digunakan untuk merencanakan suatu hal agar kegagalan dan resiko yang mungkin terjadi dapat meminimalisir.

Pada dunia pembangunan perangkat lunak juga diperlunakan pemodelan. Pemodelan perangkat lunak digunakan untuk mempermudah langkah berikutnya dari pengembangan sebuah sistem sehingga lebih terencana. Pemodelan pada pembangunan perangkat lunak untuk menvisualkan perangkat lunak yang akan dibuat [11].

2.4.4 Unified Modelling Languange (UML)

Unified Modelling Languange (*UML*) adalah sebuah bahasa pemodelan standar yang memiliki sintaks dan *semantic*. Pemodelan ini sangat cocok digunakan untuk merancang dan memodelkan sistem berorientasi objek. Diagram pada uml dibagi menjadi dua bagian yaitu *structural Diagram* dan *behavior Diagram*. *Structural Diagram* digunakan untuk mendeskripsikan relasi antar kelas, *Tools* yang digunakan pada bagian ini yaitu *Class Diagram*. Sedangkan *behavior* Diagram digunakan untuk mendeskripsikan interaksi antara aktor dan sebuah *Use Case* (bagaimana seorang aktor menggunakan sistem). *Tools* yang digunakan pada bagian ini yaitu *Use Case Diagram*, *Sequence Diagram*, dan *Activity Diagram* [12].

2.4.5 Diagram Use Case

Diagram *Use Case* digunakan untuk mendeskripsikan kejadian-kejadian apa saja yang dapat dilakukan oleh *user*/aktor dan fungsionalitas-fungsionalitas apa saja yang diharapkan dari sistem yang akan dibangun. Tanpa mendeskripsikan bagaimana sistem menyelesaikannya. Sebuah *Use Case* menggambarkan suatu urutan interaksi antara satu atau lebih aktor dan sistemm dalam tahap *requirements*. Model *Use Case* mengambarkan sistem sebagai sebuah kotak hitam dan interaksi antara aktor dan sistem dalam suatu bentuk naratif. Setiap *Use Case* menggambarkan perilaku sejumlah aspek sistem, tanpa mengurangi struktur *internal*-nya. Selama pembuatan model *Use Case* secara pararel juga harus ditetapkan obyek-obyek yang terlibat dalam setiap *Use Case* [12].

2.4.6 Diagram Sequence

Diagram *Sequence* menggambarkan interaksi antar objek di dalam dan di sekitar sistem yang menekankan pada pengiriman pesan dalam suatu waktu tertentu. Diagram *Sequence* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan dari sebuah event untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan

perubahan apa saja yang terjadi secara *internal* dan *output* apa yang dihasilkan [12].

2.4.7 Diagram Activity

Diagram *Activity* menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* atau keputusan yang mungkin terjadi, dan bagaimana mereka berakhir. Diagram *Activity* merupakan *state Diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu Diagram *Activity* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum [12].

2.4.8 Diagram Class

Diagram *Class* digunakan untuk menggambarkan keadaan suatu sistem dengan menjelaskan keterhubungan antara suatu *Class* dengan *Class* yang lain yang terdapat pada sistem. Sebuah *Class* terdiri dari nama, atribut dan *method*. Atribut dan *method* dari sebuah *Class* atau kelas mempunyai *visibility*. Ada tiga jenis *visibility* yang digunakan yaitu *private*, *public*, dan *protected*. Setiap kelas pada Diagram *Class* mempunyai hubungan dengan kelas lainnya. Ada beberapa jenis hubungan kelas, yaitu *dependency*, *asosiasi*, *agregasi* dan *generalisasi* [12].

a. *Dependency* merupakan hubungan terlemah antar kelas. *Dependency* bermakna satu kelas menggunakan atau memiliki pengetahuan terhadap kelas lain, namun hubungannya hanya sementara dan tidak ada batas waktu yang jelas.
b. *Asosiasi* mempunyai hubungan yang lebih kuat dari hubungan *dependency*. Dimana suatu *Class* tetap berhubungan dengan kelas lain seterusnya, hubungan *asosiasi* dibagi menjadi dua jenis yaitu *directional* dan *bidirectional*.

c. *Agregasi* merupakan bentuk hubungan yang mengimplikasikan kepemilikan suatu kelas. *Agregasi* juga merupakan bentuk yang lebih kuat dari *asosiasi*. Hubungan *agregasi* dinyatakan dengan simbol *diamond* pada pemilik kelas dan garis utuh berpanah kekelas yang dimiliki.
d. *Generalisasi* merupakan bentuk hubungan antar kelas, dari kelas yang umum dengan kelas yang lebih khusus. Contoh kelas hewan memiliki hubungan *generalisasi* dengan kelas kucing, karena kucing merupakan kelas khusus dari kelas hewan yang lebih umum.

2.5 C Sharp (C#)

C# (dibaca: C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan [13].

2.6 Unity

Unity 3D adalah sebuah game engine yang berbasis cross-platform. *Unity* dapat digunakan untuk membuat sebuah game yang bisa digunakan pada perangkat komputer, ponsel pintar *android*, *iPhone*, PS3, dan bahkan X-BOX. *Unity* merupakan sebuah tool yang terintegrasi untuk membuat game, arsitektur bangunan dan simulasi. Unity bisa untuk games PC dan games Online. Untuk games Online diperlukan sebuah plugin, yaitu *Unity Web Player*, sama halnya dengan *Flash Player* pada *browser*. Fitur kompiler yang disediakan, mendukung 3 bahasa pemrograman yaitu JavaScript, C#, dan Boo.

2.7 Pengujian Black Box

Pengujian *black box* merupakan pendekatan komplementer dari teknik *white box*, karena pengujian *black box* diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik *white box*. Pengujian *black* box berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan

serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian black box merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan. Pengujian *black box* berusaha menemukan kesalahan dalam kategori [14]:

1. fungsi-fungsi yang tidak benar atau hilang
2. kesalahan *interface*
3. kesalahan dalam struktur data atau akses *database eksternal*
4. kesalahan kinerja
5. inisialisasi dan kesalahan terminasi.

2.7.1 Weka

Weka machine learning workbench adalah sebuah flatform modern untuk penerapan machine learning. Weka adalah singkatan yang merupakan singkatan dari waikato environment untuk analisis pengetahuan [15]. Terdapat lima fitur dari aplikasi weka yaitu :

1. Open source
   aplikasi ini bersifat open source dibawah GNU GPL. Dan memiliki lisensi eksklusif untuk digunakan pada platform intelejen bisnis dalam produk mereka
2. Antar muka
   memiliki *Graphical User Interface* (GUI). Hal ini memungkinkan anda untuk menyelesaikan proyek-proyek machine learning tanpa melakukan pemrograman
3. Comand line interface

semua fitur dari perangkat lunak dapat digunakan dari comand line inteface. Fungsi ini sangat berguna jika ingin melulaku pengkodean pada pekerjaan yang besar.

4. API java

   hal ini tulis di pemroraman java dan menyediakan API yang didokumentasikan dengan baik dan mempromosikan integrasi ke dalam aplikasi yang dibuat pada pemrograman java.

5. Dokumentasi

   Terdapat buku *manual*, wiki dan kursus MMOC yang dapat melatih pengguna weka bagaimana menggunakan platform ini secara efektif.

2.7.2 Cross validation

Cross Validation merupakan salah satu teknik untuk menilai/memvalidasi keakuratan sebuah model yang dibangun berdasarkan dataset tertentu . Pembuatan model biasanya bertujuan untuk melakukan prediksi maupun klasifikasi terhadap suatu data baru yang boleh jadi belum pernah muncul di dalam dataset. Data yang digunakan dalam proses pembangunan model disebut data latih/training, sedangkan data yang akan digunakan untuk memvalidasi model disebut sebagai data test.

k-*fold cross validation* [16].

k-*fold cross validation* adalah sebuah teknik intensif komputer yang menggunakan keseluruhan data yang ada sebagai traning set dan test set. Seluruh data secara acak dibagi menjadi K buah subset dengan ukurang yang sama dimana B_k merupakan himpunan bagian dari (1..n) sedemikian sehingga $\cup_K^B$ dan $B_j \cap B_k = \emptyset$ setelah itu dilakukan iterasi sebanyak k kali. Pada iterasi ke k subset bk menjadi test test, sedangkan subset yang lain menjadi training set. Setelah itu dihitung nilai rata-rata error dengan menggunakan hasil dari k buah iterasi.

Kelebihan dari metode ini adalah tidak adanya masalah dalam pembagian data. Setiap data akan menajdi test set sebanyak satu kali dan akan menjadi training set sebanyak K-1 kali. Kekurangan dari metode ini adalah algoritma pembelajaran

harus dilakukan sebanyak K kali yang berarti menggunakan K kali waktu komputasi [16].