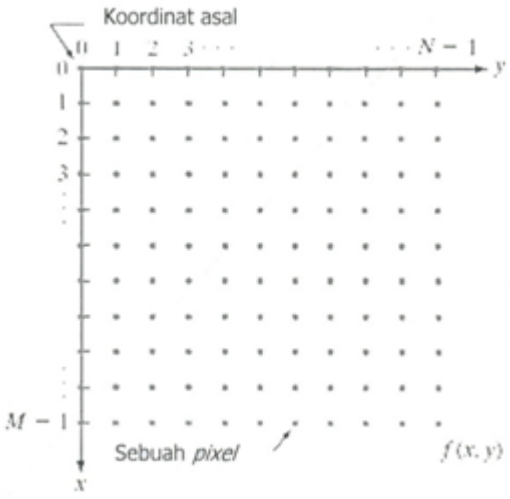
BAB II
LANDASAN TEORI

II.1 Pengolahan Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (array) yang berisi nilai real maupun kompleks yang direpresentasikan dengna deretan bit tertentu.

Suatu citra dapat didefinisikan sebagai fungsi tertentu f(*x*,*y*) berukuran M baris dan N kolom, dengan x dan y adalah kordinat spasial, dan amplituda f dititik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Gambar II.1 menunjukan posisi koordinat citra digital.



Gambar II.1 Koordinat Citra Digital

Citra digital dapat dituliskan dalam bentuk matrik sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \quad \text{..... (1)}$$

II.1.1 Jenis Citra

Nilai suatu pixel memiliki dalam suatu rentang tertentu, dari nilai minimum sampai nilai maksimum. Jangkauan yang digunakan berbeda-beda tergantung dari jenis warnanya. Namun secara umum jangkauanya adalah 0-255. Citra dengan penggambaran seperti ini digolongkan ke dalam citra integer. Berikut adalah jenis-jenis citra berdasarkan nilai pixelnya.

II.1.1.1 Citra Biner

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai pixel yaitu hitam dan putih. Citra biner juga sering disebut dengan citra *black and white* atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap pixel dari citra biner.

II.1.1.2 Citra Grayscale

Citra grayscale merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian RED = GREEN = BLUE. Nilai tersebut digunakan untuk menunjukan tingkat intensitas. Warna yang dimiliki adlah warna dari hitam, keabuan, dan putih. Tingkat keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Citra grayscale memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan).

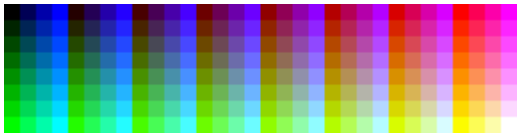
II.1.1.3 Citra Warna (8 bit)

Setiap pixel dari citra wana (8 bit) hanya diwakili oleh 8 bit dengan jumlah maksimum yang dapat digunakan adalah 256 warna. Ada dua jenis citra warna 8 bit. Pertama, citra warna 8 bit dengan menggunakan palet warna 256 degnan setiap paletnya memiliki pemetaan nilai (colormap) RGB tertentu. Model ini lebih sering digunakan. Kedua, setiap pixel memiliki format 8 bit sebagai berikut Tabel II-1 citra warna 8 bit truecolor.

Tabel II-1 citra warna 8 bit truecolor

Bit - 7	Bit - 6	Bit - 5	Bit -4	Bit - 3	Bit - 2	Bit - 1	Bit – 0
R	R	R	G	G	G	B	B

Bentuk kedua ini dinamakan 8 bit truecolor. Berikut ini adalah warna-warnanya pada Gambar II.2.



Gambar II.2 8 *bit truecolor*

II.1.1.4 Citra Warna (16 bit)

Citra warna 16 bit (biasanya disebut citra *highcolor*) dengan setiap pixelnya diwakili dengan 2 *byte memory* (16 bit).

Warna 166 bit meiliki 65.536 warna. Dalam formasi bitnya, nilai merah dan biru mengambil tempat di 5 bit di kanan dan kiri. Komponen hijau memiliki 5 bit ditambah 1 bit ekstra . pemilihan komponen hijau degnan deret bit dikarenakan penglihatan manusia lebih sensitif terhadap warna hiaju.

Tabel II-2 citra warna16 bit

Bit - 15	Bit - 14	Bit - 13	Bit - 12	Bit - 11	Bit - 10	Bit - 9	Bit - 8	Bit - 7	Bit - 6	Bit - 5	Bit - 4	Bit - 3	Bit - 2	Bit - 1	Bit - 0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

II.1.1.5 Citra Warna (24 bit)

Setiap pixel dari citra warna 24 bit diwakili degnan 24 bit sehingga total 16.777.216 variasi warna. Variasi ini sudah lebih dari cukup untuk memvisualisasikan seluruh warna yang dapat dilihat penglihatan manusia. Penglihatan manusia dipercaya hanya dapat membedakan hingga 10 juta warna saja.

Setiap poin informasi pixel (RGB) disimpan ke dalam 1 byte data. 8 bit pertama menyimpan nilai biru, kemudian diikuti dengan nilai hijau pada 8 bit kedua dan pada 8 bit terakhir merupakan warna merah.

II.2 Kecerdasan Buatan

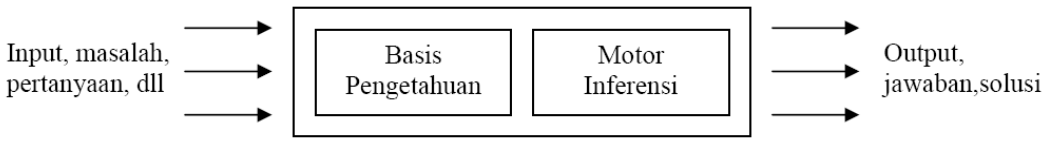
Artificial Intelligence merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik manusia [3]. Kelebihan penerapan *artificial intelligence* pada suatu sistem adalah sistem dapat menarik kesimpulan dari hasil pembelajaran berdasarkan basis pengetahuan yang dimasukkan. Contoh sistem berbasis artificial intelligence adalah sistem pengenalan gambar menggunakan jaringan syaraf tiruan, yaitu suatu sistem dimana sistem tersebut dapat mengenali gambar dengan menggunakan data yang sudah diperoleh dari proses pembelajaran gambar yang telah dilakukan sebelumnya.

Menurut John McCarthy, 1956, AI :

"Untuk mengetahui dan memodelkan proses – proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia."

Cerdas adalah memiliki pengetahuan dan pengalaman, penalaran yaitu bagaimana membuat keputusan dan mengambil tindakan, serta moral yang baik. Agar mesin bisa cerdas atau bertindak seperti dan sebaik manusia, maka harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Penerapan kecerdasan buatan digambarkan pada Gambar II.3. Dua bagian utama yg dibutuhkan untuk aplikasi kecerdasan buatan adalah [3]:

1. Basis pengetahuan (*knowledge base*): berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
2. Motor inferensi (*inference engine*): kemampuan menarik kesimpulan berdasarkan pengetahuan.



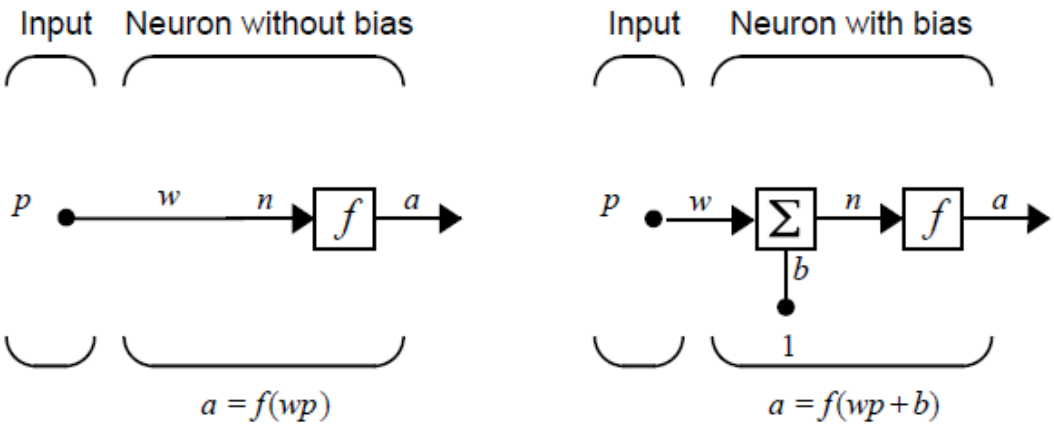
Gambar II.3Penerapan Kecerdasan Buatan

II.3 *Neural Network*

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 [4].

II.3.1 Simple Neuron

Sebuah neuron dengan masukan skalar tunggal dan ada bias muncul di sebelah kiri bawah seperti pada Gambar II.4.



Gambar II.4 *Simple Neuron*

Skalar masukan p ditularkan melalui koneksi yang mengalikannya kekuatan dengan berat skalar w, untuk membentuk wp produk, lagi skalar. di sini wp masukan bobot adalah satu-satunya argumen dari fungsi transfer f, yang menghasilkan skalar output. Neuron di sebelah kanan memiliki bias skalar, b.

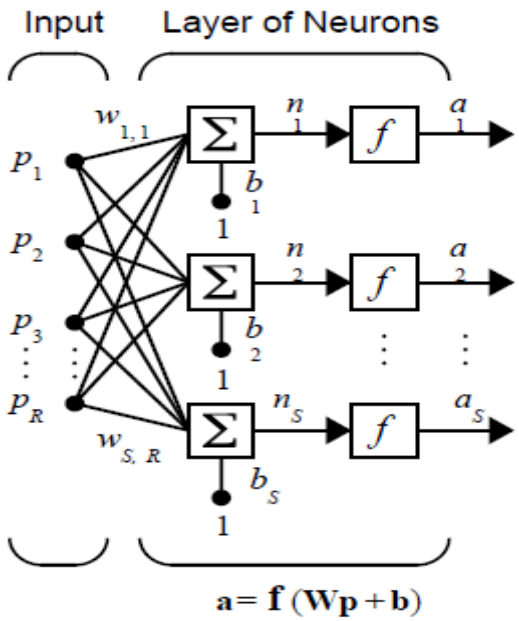
Anda dapat melihat bias karena hanya akan ditambahkan pada wp produk seperti yang ditunjukkan oleh menjumlahkan persimpangan atau pergeseran fungsi f ke kiri dengan jumlah yang b. Itu Bias jauh seperti berat, kecuali bahwa ia memiliki masukan konstan 1 [5].

II.3.2 *Network Architectures*

Dua atau lebih neuron ditampilkan sebelumnya dapat dikombinasikan dalam sebuah layer, dan jaringan tertentu bisa mengandung satu atau lebih lapisan tersebut. Pertama mempertimbangkan single layer neuron.

II.3.2.1 *A Layer of Neurons*

Sebuah jaringan satu layer dengan elemen masukan R dan S neuron, berikut pada Gambar II.5.



R = jumlah elemen dalam vektor masukan

S = jumlah neuron pada lapisan

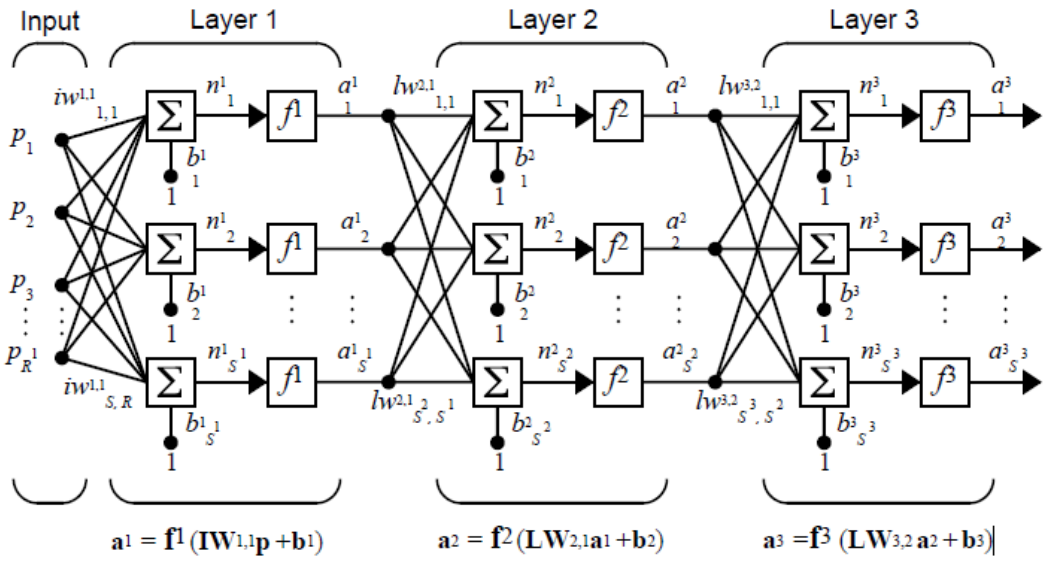
Gambar II.5 *A Layer of Neurons*

Dalam jaringan ini, setiap elemen dari vektor masukan p terhubung ke setiap neuron masukan melalui matriks bobot W. engan neuron memiliki musim panas yang mengumpulkan input bobot dan bias membentuk sendiri keluaran skalar n (i). Berbagai n (i) diambil bersama-sama membentuk sebuah S-elemen

input net vektor n. Akhirnya, lapisan neuron utput membentuk kolom vektor. Menunjukkan ekspresi untuk di bagian bawah angka [5].

II.3.2.2 *Multiple Layers of Neurons*

Sebuah jaringan dapat memiliki beberapa lapisan. Setiap lapisan memiliki W matriks bobot, bias vektor b, dan output vektor. Untuk membedakan antara matriks berat, vektor output, dll, untuk masing-masing lapisan tersebut dalam angka, menambahkan jumlah lapisan sebagai superscript ke variabel bunga. Anda dapat melihat penggunaan notasi lapisan ini di jaringan tiga lapis ditunjukkan di bawah, dan di persamaan di bagian bawah Gambar II.6.



$$\mathbf{a}_3 = \mathbf{f}_3(\mathbf{LW}_{3,2}\mathbf{f}_2(\mathbf{LW}_{2,1}\mathrm{f}_1(\mathbf{IW}_{1,1}\mathbf{p} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3)$$

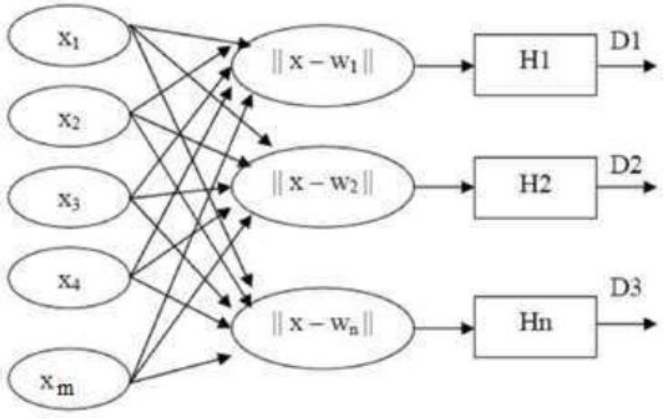
Gambar II.6 *Multiple Layers of Neurons*

Jaringan yang ditunjukkan di atas memiliki input R1, S1 neuron di lapisan pertama, S2 neuron pada lapisan kedua, dll Adalah umum bagi lapisan yang berbeda untuk memiliki perbedaan jumlah neuron. Sebuah masukan konstan 1 diumpankan ke bias untuk masing-masing neuron [5].

II.3.3 *Learning Vector Quantization*

II.3.3.1 *Architecture*

Learning Vector Quantization (LVQ) merupakan salah satu bentuk Algoritma Jaringan Syaraf Tiruan (*Neural Network*) dengan metode pembelajaran pada lapisan kompetitif yang terawasi [6]. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor input. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor input. Jika 2 vektor input mendekati sama, maka lapisan kompetitif akan melakukan kedua vektor input tersebut ke dalam kelas yang sama. Untuk melihat arsitektur *learning vector quantization* bisa dilihat pada Gambar II.7.



Gambar II.7 Arsitektur LVQ

Keterangan :

- x1 sampai dengan xm = nilai input
- || x - w1 || sampai dengan || x - wn || = jarak bobot
- H1 sampai dengan Hn = lapisan output
- D1 sampai dengan Dn = nilai output
- m : jumlah elemen input

- n = jumlah kelas

II.3.3.2 Algoritma *Learning Vector Quantization*

Langkah 0 : Inisialisasikan vektor referensi dan *learning rate(α).*

Langkah 1 : Selama kondisi berhenti bernilai salah, kerjakan langkah 2 dan 3. (Kondisi berhenti jika perulangan mencapai maksimal perulangan dan perubahan rasio lebih kecil dari minimal eror)

Langkah 2 : Untuk masing-masing pelatihan vektor masukan , kerjakan:

a. Temukan J sehingga $\|x - wj\|$ bernilai minimum.

$$Cj = \sqrt{\sum_{i=1}^{n}(x_i - w_{ij})^2} \quad \text{..... (2)}$$

b. Perbaiki Wj dengan :

- Jika T=Cj maka

  Wj(baru)=Wj(lama)+α[x-Wj(lama)]..... (3)

- Jika T≠Cj maka

  Wj(baru) = Wj(lama) - α [x - Wj(lama)]..... (4)

Langkah 3 : Kurangi learning rate.

Keterangan rumus:

T : Target

J : Jumlah selisih data dan bobot

C : kelas selisih bobot terkecil

W : Bobot

α : Rasio pembelajaran

x : Data

Sebuah jaringan LVQ memiliki lapisan kompetitif pertama dan lapisan kedua linear. itu Lapisan kompetitif belajar untuk mengklasifikasikan vektor masukan dalam banyak cara yang sama seperti lapisan kompetitif . Lapisan linear

mengubah kompetitif Kelas layer ke dalam klasifikasi target yang telah ditetapkan oleh pengguna. mengacu pada Kelas dipelajari oleh lapisan kompetitif sebagai subclass dan kelas dari Lapisan linear sebagai kelas sasaran.

Kedua lapisan kompetitif dan linear memiliki satu neuron per (sub atau target) kelas. Dengan demikian, lapisan kompetitif dapat belajar sampai S^1 subclass. Ini, pada gilirannya, digabungkan oleh lapisan linier untuk membentuk S^2 kelas sasaran. (S^1 selalu lebih besar dari S^2.)

Misalnya, neuron 1, 2, dan 3 di lapisan kompetitif semua belajar subclass dari ruang input yang dimiliki lapisan kelas target linear No 2. Kemudian neuron kompetitif 1, 2, dan 3, akan memiliki $LW^{2,1}$ bobot 1,0 sampai neuron n^2 di lapisan linear, dan bobot 0 sampai semua neuron linear lainnya. Dengan demikian, linear neuron menghasilkan 1 jika salah satu dari tiga neuron kompetitif (1,2, dan 3) memenangkan persaingan dan output 1. Ini adalah bagaimana subclass dari Lapisan kompetitif digabungkan ke dalam kelas target dalam lapisan linear.

Singkatnya, a 1 di baris i^{th} dari a^1 (sisanya untuk elemen a^1 akan menjadi nol) efektif mengambil kolom ke-i dari $LW^{2,1}$ sebagai output jaringan. setiap seperti kolom berisi satu 1, sesuai dengan kelas tertentu. Dengan demikian, 1s subclass dari layer 1 bisa dimasukkan ke dalam berbagai kelas, oleh $LW^{2,1}a^1$ perkalian dalam lapisan 2.

Sebelumnya apa fraksi dari lapisan 1 neuron harus diklasifikasikan ke dalam berbagai output kelas layer 2, sehingga dapat menentukan unsur-unsur $LW^{2,1}$ di awal. Namun, harus melalui prosedur pelatihan untuk mendapatkan lapisan pertama untuk menghasilkan output yang benar untuk setiap subclass vektor training set. Pertama mempertimbangkan cara menciptakan jaringan asli.

II.3.3.3 *Creating an LVQ Network (newlvq)*

Sebuah jaringan *LVQ* dapat dibuat dengan fungsi *newlvq*.

net = newlvq(PR,S1,PC,LR,LF)

di mana:

a. PR adalah R-by-2 matriks nilai-nilai minimum dan maksimum untuk R masukan elemen.
b. S1 adalah jumlah lapisan pertama neuron tersembunyi.
c. PC adalah elemen vektor S^2 persentase kelas yang khas.
d. LR adalah tingkat pembelajaran (default 0,01).
e. LF adalah fungsi pembelajaran (default adalah learnlv1).

Misalkan memiliki 10 vektor input. Maka membuat jaringan yang memberikan masing-masingmasukan ini vektor ke salah satu dari empat subclass. Dengan demikian, memiliki empat neuron di Lapisan pertama kompetitif. Subclass ini kemudian ditugaskan untuk salah satu dari dua keluaran kelas oleh dua neuron pada lapisan 2. Vektor input dan target yang ditentukan oleh

P = [-3 -2 -2 0 0 0 0 +2 + 2 +3;

0 +1 -1 +2 +1 -1 -2 +1 -1 0]

Dan

Tc = [1 1 1 2 2 2 2 1 1 1];

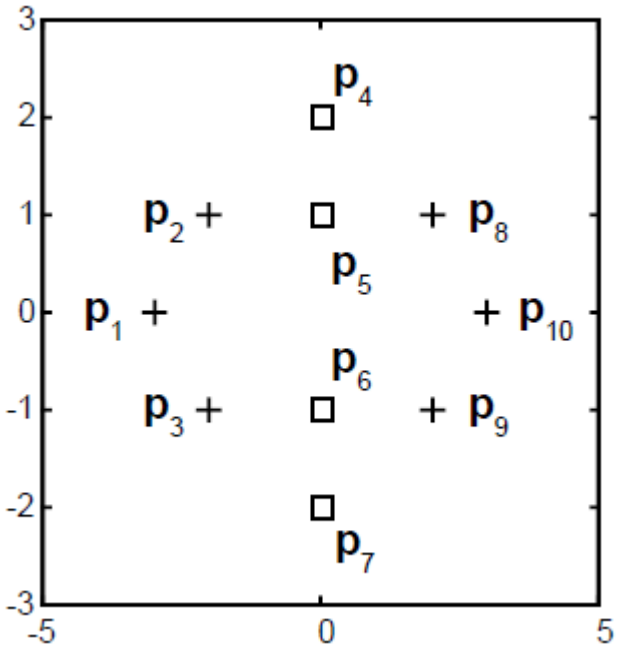
Ini dapat membantu untuk menunjukkan rincian dari apa yang didapatkan dari dua baris kode.

P =-3 -2 -2 0 0 0 0 2 2 3 0 1 -1 2 1 -1 -2 1 -1 0

Tc =1 1 1 2 2 2 2 1 1 1

Sebuah plot vektor masukan berikut.

Seperti yang lihat terlihat pada Gambar II.8, ada empat subclass dari vektor input. ingin jaringan yang mengklasifikasikan p1, p2, p3, p8, p9, dan p10 untuk menghasilkan output 1, dan bahwa mengklasifikasikan vektor p4, p5, p6 p7 dan untuk menghasilkan output 2.



Gambar II.8 *Input Vectors*

Catatan bahwa ini masalah adalah nonlinearly dipisahkan, sehingga tidak dapat diselesaikan dengan perceptron, tapi jaringan LVQ memiliki kesulitan.

Selanjutnya mengubah matriks Tc untuk menargetkan vektor.

T = ind2vec(Tc)

Hal ini memberikan T matriks jarang yang dapat ditampilkan secara penuh dengan

targets = full(T)

yang memberikan

targets = 1 1 1 0 0 0 0 1 1 1

0 0 0 1 1 1 1 0 0 0

Ini terlihat benar. Ia mengatakan, misalnya, bahwa jika memiliki kolom pertama dari P sebagai input, harus mendapatkan kolom pertama dari target sebagai output; dan output yang kata input jatuh di kelas 1, yang benar. Sekarang siap untuk memanggil newlvq.

Menyebutnya newlvq dengan argumen yang tepat sehingga membuat jaringan dengan empat neuron di lapisan pertama dan dua neuron pada lapisan kedua. itu pertama-lapisan bobot diinisialisasi ke pusat input berkisar dengan fungsi titik tengah. Kedua-layer berat memiliki 60% (6 dari 10 di Tc di atas) kolom dengan 1 pada baris pertama, (sesuai dengan kelas 1), dan 40% dari kolom yang akan memiliki 1 di baris kedua (sesuai dengan kelas 2) [5].

```
net = newlvq(minmax(P),4,[.6 .4], 0,1);
```

Bisa memeriksa untuk melihat nilai awal matriks berat pertama-layer.

```
net.IW{1,1}
ans =     0 0
          0 0
          0 0
          0 0
```

Ini nol bobot memang nilai-nilai pada titik tengah dari kisaran (-3 hingga +3) dari input, seperti yang diharapkan ketika menggunakan titik tengah untuk inisialisasi. Bisa melihat bobot kedua layer dengan

```
net.LW{2,1}
ans =     1 1 0 0
          0 0 1 1
```

Ini masuk akal juga. Ia mengatakan bahwa jika lapisan kompetitif menghasilkan 1 sebagai pertama atau kedua elemen. Vektor input diklasifikasikan sebagai kelas 1; kalau tidak kelas 2.

Anda mungkin memperhatikan bahwa dua pertama neuron kompetitif yang terhubung ke pertama neuron linear (dengan bobot 1), sedangkan yang kedua dua neuron yang kompetitif adalah terhubung ke neuron linear kedua. Semua bobot lainnya antara neuron kompetitif dan neuron linear memiliki nilai 0. Dengan

demikian, masing-masing dua Kelas sasaran (neuron linear), pada kenyataannya, penyatuan dua subclasses (the neuron kompetitif).

Bisa mensimulasikan jaringan dengan sim. Menggunakan P matriks asli sebagai masukan hanya untuk melihat apa yang didapatkan.

Y = sim(net,P);

Y = vec2ind(Yb4t)

Y = 1 1 1 1 1 1 1 1 1 1

Jaringan mengklasifikasikan semua input ke dalam kelas 1. Sejak tis bukan apa yang diinginkan, maka harus melatih jaringan (menyesuaikan bobot dari lapisan 1 saja), sebelum bisa mengharapkan hasil yang baik. Pertama membahas dua aturan pembelajaran LVQ, dan kemudian melihat pada proses pelatihan.

II.3.3.4 *LVQ1 Learning Rule(learnlv1)*

Pembelajaran LVQ pada lapisan kompetitif didasarkan pada satu set pasangan input / target.

$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ ..... (5)

Setiap vektor target memiliki satu 1. Sisanya unsur-unsurnya adalah 0. 1 The menceritakan klasifikasi yang tepat atas masukan terkait. Sebagai contoh, pertimbangkan berikut pair pelatihan.

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right\}$$ ..... (6)

Memiliki vektor masukan dari tiga unsur, dan setiap vektor input menjadi ditugaskan untuk salah satu dari empat kelas. Jaringan ini harus dilatih sehingga mengklasifikasikan vektor input yang ditunjukkan di atas ke tingkat yang ketiga dari empat kelas.

Untuk melatih jaringan, vektor masukan p disajikan, dan jarak dari p untuk setiap baris dari matriks berat input, $IW^{1,1}$ dihitung dengan fungsi ndist. Neuron yang tersembunyi dari lapisan 1 bersaing. Misalkan elemen i dari n^1 adalah yang paling positif, dan neuron i * memenangkan kompetisi. Kemudian kompetitif fungsi transfer menghasilkan 1 sebagai i * th elemen a^1. Semua unsur-unsur lain dari a^1 adalah 0.

Ketika a^1 dikalikan dengan bobot 2 lapisan $LW^{2,1}$, single 1 di memilih a^1 kelas, k * terkait dengan input. Dengan demikian, jaringan telah menugaskan vektor masukan p ke kelas k * dan a^2_{k*} akan 1. Tentu saja, tugas ini mungkin satu yang baik atau buruk, untuk t_{k*} mungkin 1 atau 0, tergantung pada apakah input milik kelas k * atau tidak.

Menyesuaikan $i*^{th}$ deretan $IW^{1,1}$ sedemikian rupa untuk memindahkan baris ini lebih dekat ke vektor masukan p jika tugas sudah benar, dan untuk memindahkan baris jauh dari p jika tugas tidak benar. Jadi jika p diklasifikasikan dengan benar,

$$a^2_{k*} = t_{k*} = 1$$

Menghitung nilai baru dari $i*^{th}$ deretan $IW^{1,1}$ sebagai:

$$_{i*}\mathbf{IW}^{1,1}(q) = {}_{i*}\mathbf{IW}^{1,1}(q-1) + \alpha(\mathbf{p}(q) - {}_{i*}\mathbf{IW}^{1,1}(q-1))$$

..... (7)

Di sisi lain, jika p diklasifikasikan salah,

$$a^2_{k*} = 1 \neq t_{k*} = 0$$ ..... (8)

Menghitung nilai baru dari $i*^{th}$ deretan $IW^{1,1}$ sebagai:

$$_{i*}\mathbf{IW}^{1,1}(q) = {}_{i*}\mathbf{IW}^{1,1}(q-1) - \alpha(\mathbf{p}(q) - {}_{i*}\mathbf{IW}^{1,1}(q-1))$$

..... (9)

Koreksi ini ke $i*^{th}$ deretan $IW^{1,1}$ dapat dibuat secara otomatis tanpa mempengaruhi baris lain $IW^{1,1}$ oleh backpropagating kesalahan keluaran kembali ke lapisan 1.

Koreksi tersebut memindahkan neuron tersembunyi terhadap vektor yang masuk dalam kelas yang membentuk subclass, dan jauh dari vektor yang jatuh ke lainnya kelas.

Fungsi pembelajaran yang mengimplementasikan perubahan-perubahan di lapisan 1 beban di Jaringan LVQ adalah learnlv1. Hal ini dapat diterapkan selama pelatihan.

II.3.3.5 *Supplemental LVQ2.1 Learning Rule (learnlv2)*

Aturan pembelajaran berikut ini adalah salah satu yang mungkin diterapkan setelah terlebih dahulu menerapkan

LVQ1. Ini dapat meningkatkan hasil belajar pertama. Versi ini dari LVQ2 (disebut sebagai LVQ2.1 dalam literatur) diwujudkan dalam fungsi learnlv2. Perhatikan lagi LVQ2.1 yang akan digunakan hanya setelah LVQ1 memiliki telah diterapkan.

Belajar di sini adalah mirip dengan yang di learnlv1 kecuali sekarang dua vektor lapisan 1yang paling dekat dengan vektor input dapat diperbarui menyediakan satu yang dimiliki ke kelas yang benar dan satu milik kelas yang salah dan selanjutnya memberikan bahwa input jatuh menjadi "jendela" dekat midplane dari dua vektor.

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s \text{ di mana } s \equiv \frac{1-w}{1+w} \quad \text{.....(10)}$$

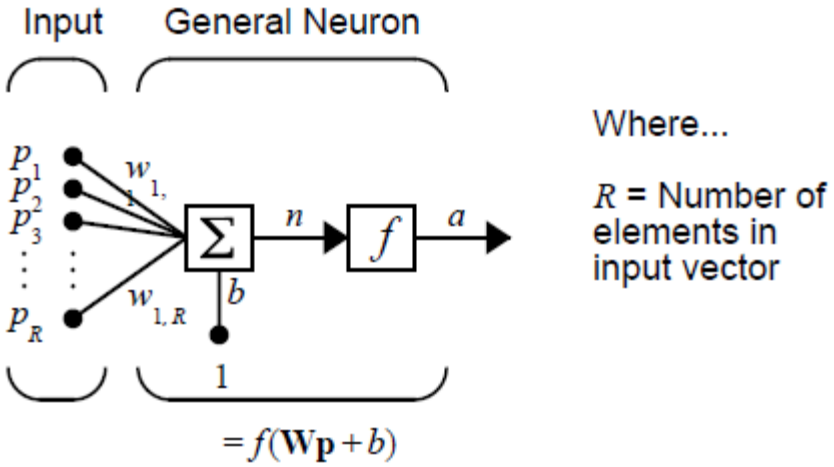
II.3.4 *Backpropagation*

Backpropagation diciptakan oleh generalisasi pembelajaran aturan Widrow-Hoff ke jaringan multi-layer dan fungsi nonlinier terdiferensialkan transfer. masukan vektor dan vektor target yang sesuai digunakan untuk melatih jaringan sampai dapat mendekati fungsi, masukan asosiasi vektor dengan output tertentu vektor, atau mengklasifikasikan vektor masukan dengan cara yang tepat seperti yang didefinisikan oleh Anda. Jaringan dengan bias, lapisan sigmoid, dan lapisan output linier mampu mendekati fungsi apapun dengan jumlah terbatas diskontinuitas.

Standar backpropagation adalah algoritma gradient descent, seperti Aturan belajar Widrow-Hoff, di mana bobot jaringan dipindahkan sepanjang negatif dari gradien dari fungsi kinerja. istilah backpropagation mengacu pada cara di mana gradien dihitung untuk jaringan multilayer nonlinear. Ada beberapa variasi pada dasar algoritma yang didasarkan pada teknik optimasi standar lainnya, seperti conjugate gradient dan metode Newton. The Neural Network Toolbox menerapkan sejumlah variasi ini. Bab ini menjelaskan cara menggunakan masing-masing rutinitas ini dan membahas keuntungan dan kekurangan masing-masing.

Jaringan backpropagation terlatih cenderung memberikan jawaban yang masuk akal ketika disajikan dengan input yang mereka belum pernah melihat. Biasanya, sebuah masukan baru mengarah ke output yang mirip dengan output yang benar untuk vektor input yang digunakan dalam pelatihan yang mirip dengan masukan baru yang disajikan. generalisasi ini properti memungkinkan untuk melatih jaringan pada satu set mewakili input / menargetkan pasangan dan mendapatkan hasil yang baik tanpa pelatihan jaringan tersebut pada semua kemungkinan pasangan input / output. Ada dua fitur dari Neural Network Toolbox yang dirancang untuk meningkatkan jaringan generalisasi - regularisasi dan awal berhenti.

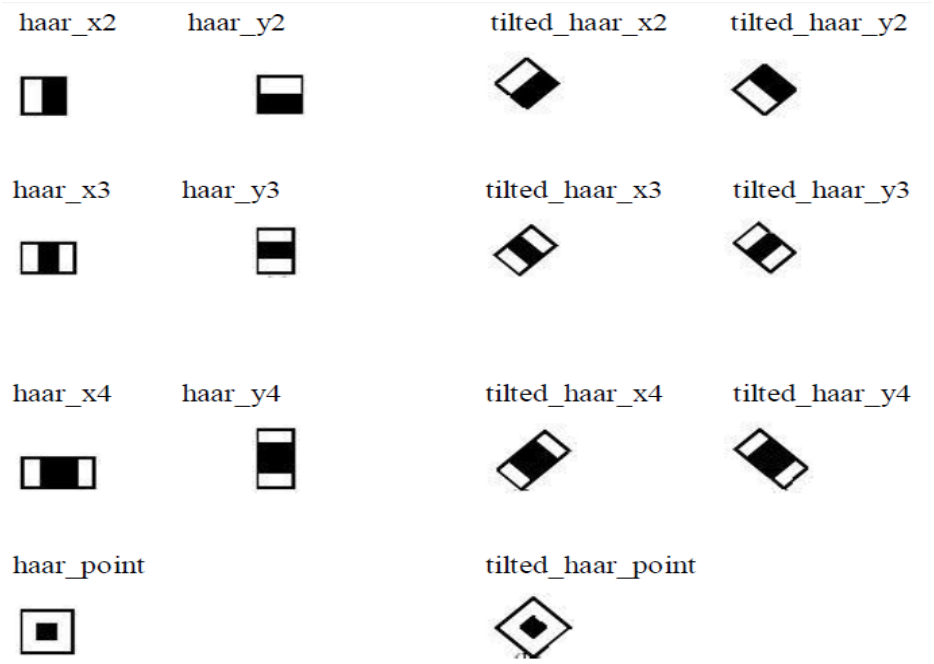
Sebuah neuron dasar dengan input R ditampilkan di bawah. Setiap input bobot dengan tepat w. Jumlah input bobot dan bias membentuk input ke fungsi transfer f. Neuron dapat menggunakan transfer terdiferensialkan berfungsi f untuk menghasilkan output.



Gambar II.9 arsitektur *backpropagation*

II.4 *Haar Like Features*

Algoritma Haar menggunakan metode statistikal dalam melakukan pendeteksian objek. Tiap *feature* dari *Haar-like feature* didefinisikan pada bentuk dari *feature*, diantaranya koordinat dari *feature* dan juga ukuran dari *feature* tersebut. *Feature* yang akan di *training* adalah *feature* yang didapat dari proses *cropping*. Adapun macam-macam variasi *feature haar* seperti pada Gambar II.10:

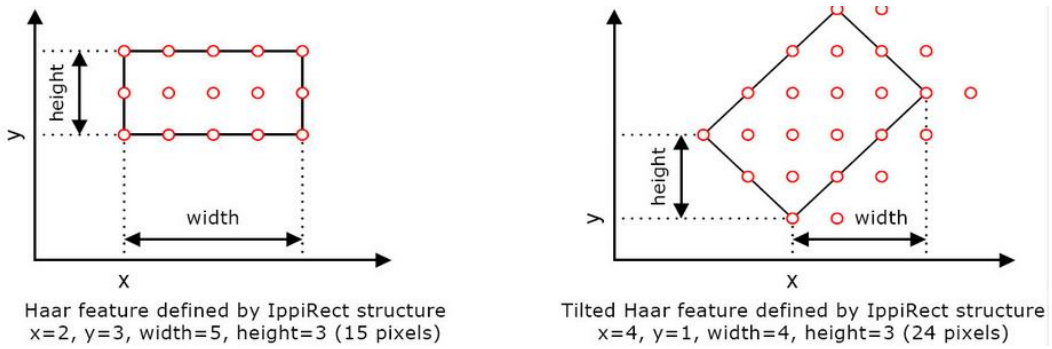


Gambar II.10 *Haar like feature*

Tiap *feature* terbagi atas dua bagian, yaitu bagian berwarna putih dan bagian berwarna hitam. Penggunaan fitur dilakukan karena pemrosesan fitur berlangsung lebih cepat dibandingkan pemrosesan citra perpiksel sehingga cocok untuk diterapkan pada proses pendeteksian objek secara *real time.* Pada penelitian ini untuk menghitung nilai fitur Haar pada sebuah citra dan pada skala yang berbeda secara cepat menggunakan satu teknik yang disebut *integral image.*

Secara umum integrating berarti menggabungkan unit-unit kecil menjadi satu. Dalam hal ini unit-unit kecil adalah nilai piksel. Nilai integral suatu piksel adalah jumlah dari semua piksel diatas dan di kiri piksel tersebut. Seluruh citra dapat digabungkan dengan operasi nilai yang lebih sedikit tiap pikselnya. *Integral image* sangat membantu dalam perhitungan fitur *Haar-like*. Dengan menggunakan *integral image*, perhitungan fitur *Haar-like* dapat dilakukan dengan sangat cepat. Untuk contohnya dapat dilihat pada Gambar II.11.

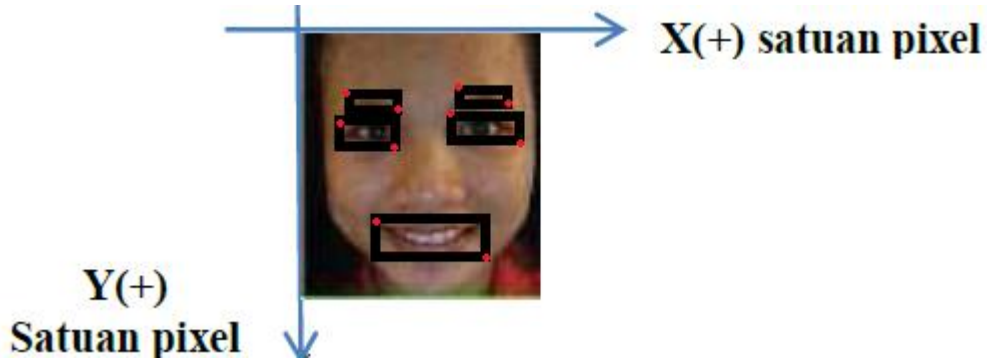
Rumus integral image: $P(x, y) = \sum_{x \leq x, y \leq y} i(x, y)$ .....(11)



Gambar II.11 *Representing Haar Features*

II.5 Fitur Wajah

Nilai Fitur wajah dapat diambil dari nilai koordinat titik tepinya dan dihitung nilai fiturnya berdasarkan titik koordinat tersebut [7]. Dimana posisi koordinat terhitung dimulai dari sisi pojok kiri atas (seperti pada Gambar II.12) .



Gambar II.12 Sistem Koordinat

Di mana nilai koordinat yang diperoleh dari masing-masing fitur adalah sebagai berikut :

1. Fitur mulut

Nilai lebar/tinggi mulut dengan perbedaan nilai kordinat pada sumbu y dengan rumus:

Lebar mulut = Y2 – Y1 ..... (12)

koordinat titik tengah mulut untuk menghitung nilai fitur lainnya.

titikTengah X = ((X2 - X1) / 2) + X1 ..... (13)

titikTengah Y = ((Y2 - Y1) / 2) + Y1 ..... (14)

2. Fitur Mata

Pada fitur mata dikenali 2 buah rectangle yang menyatakan posisi dari mata kiri dan mata kanan, sehingga nilai koordinat nya disimpan dalam array (mataX1[],mataY1[],mataX2[], mataY2[]).

$$\text{Lebar mata} = \frac{(\text{mataY1}[2]-\text{mataY1}[1])+(\text{mataY2}[2]-\text{mataY2}[1])}{2} \quad \text{..... (15)}$$

Mencari titik tengah mata1 dan mata 2 untuk menghitung jarak antara kedua mata

centermataX1 = ((mataX1[2]-mataX1[1])/2)+mataX1[1] ..... (16)

centermataY1 = ((mataY1[2]-mataY1[1])/2)+mataY1[1] ..... (17)

centermataX2 = ((mataX2[2]-mataX2[1])/2)+mataX2[1] ..... (18)

centermataY2 = ((mataY2[2]-mataY2[1])/2)+mataY2[1] ..... (19)

karena posisi titik tengah belum tentu dalam kesejajaran, maka perhitungan jarak mata menggunakan pytagoras.

$$\text{titik tengah kedua mata} = \sqrt{a^2 + b^2} \quad \text{..... (20)}$$

a = |centermataX2 – centermataX1| ..... (21)

b = |centerMataY2 – centerMataY1| ..... (22)

Mencari titik tengah kedua mata untuk menghitung jarak alismata dengan mata.

centermataX = ((centermataX2 – centermataX1)/2)+ centermataX1 ..... (23)

centermataY = ((centermataY2 – centermataY1)/2)+ centermataY1 ..... (24)

Fitur Alis mata berdasarkan koordinat pixel lokasi alis yang didapat akan dicari nilai dari ordinat x terendah dan x tertinggi (menyatakan titik ujung alis), yaitu alisX1[], alisY1[],alisX2[],alisY2[]. Menentukan nilai jarak alismata dengan mata, dengan menentukan nilai centeralisX dan centeralisY terlebih dahulu.

Centeralisx1 = ((alisX1[2]-alisX1[1])/2) + alisX1[1] ..... (25)

Centeralisy1 = ((alisY1[2]-alisY1[1])/2) + alisY1[1] ..... (26)

Centeralisx2 = ((alisX2[2]-alisX2[1])/2) + alisX2[1] ..... (27)

Centeralisy2 = ((alisY2[2]-alisY2[1])/2) + alisY2[1] ..... (28)

CenteralisX = ((centeralisx2 – centeralisx1)/2) + centeralisx1 ..... (29)

CenteralisY = ((centeralisy2 – centeralisy1)/2) + centeralisy1 ..... (30)

Menghitung nilai fitur lebar alis dengan mata

$$\sqrt{(\text{centeralisx} - \text{centermatax})^2 + (\text{centeralisy} - \text{centermatay})^2} \text{..... (31)}$$

Menghitung nilai fitur jarak antara kedua alismata

$$\sqrt{(\text{centeralisx2} - \text{centeralisx1})^2 + (\text{centeralisy2} - \text{centeralisy1})^2} \text{ ..... (32)}$$

Menghitung nilai fitur jarak antara alismata dengan mulut

$$\sqrt{(\text{titikTengahMulutX} - \text{centeralisX})^2 + (\text{titikTengahMulutY} - \text{centeralisY})^2}$$
..... (33)