BAB II

LANDASAN TEORI

2.1 Jenis Burung dalam Perlombaan

Tidak semua jenis burung dilombakan atau dapat mengikuti sebuah kontes, biasanya kontes burung dinilai dari kicauan atau suaranya yang merdu selain juga penampilannya yang bagus juga gesit. Ada beberapa jenis burung di Indonesia yang sering di adakan lombanya, berikut ini beberapa jenis burung berkicau yang ada lomba atau kontesnya:

1. Burung Cucak Ijo
2. Burung Kacer
3. Burung Murai Batu
4. Burung Jalak Suren
5. Burung Perkutut
6. Burung Cucak Jenggot
7. Burung Kenari

Sebenarnya masih ada beberapa burung berkicau lainnya seperti tekukur atau Tledekan yang biasa dibuat lombanya, namun yang daftar diatas tersebut merupakan jenis burung paling populer di Indonesia dan paling banyak dilombakan.

Biasanya burung yang menang lomba selain mendapat hadiah berupa uang maupun barang juga harga burungnya tersebut otomatis berbanderol mahal, maka dari itu tidak heran jika ada seekor burung yang menang sebuah kontes berharga

sampai puluhan juta bahkan ratusan juta rupiah. Kategori burung yang bagus tentu saja burung yang sehat, memiliki suara kicauan yang khas dan melengking, gesit dan tidak loyo, selain tentunya warna dan kebersihan bulunya juga harus terlihat bagus.((infonesiana.com), 2015).

2.1.1 Burung Kenari

Burung kenari untuk pertama kalinya di temukan oleh seorang Pelaut asal Perancis, yaitu Jean de Berthan Cout di area Kepulauan Canary sekitar abad ke-15 silam. Ia dibuat terkesima akan keindahan bulu dan merdunya suara burung ini. Burung kenari memiliki nama latin *Serinus Canaria*. Keanekaragaman dari burung kenari pada saat ini merupakan hasil perkembangan dari kenari liar. Selain itu kondisi alam dan kawin silang yang terjadi menghasilkan beragam jenis kenari yang mana sudah terjadi sejak 5 abad silam. Gambar 2.1. menunjukkan foto burung kenari roller.

Gambar 2.1 Burung Kenari Roller ((omkicau.com),2016)

Burung kenari roller termasuk salah satu jenis kenari yang paling banyak digemari oleh kenari mania. Burung ini termasuk kenari yang berketurunan dari jerman, dan telah banyak di budidayakan di daerah pegunungan di jerman yang bernama Hartz.

2.1.2 Suara Burung Kacer

Pada gambar 2.2 menunjukan foto Burung kacer atau *Magpie Robin* yang populer di Indonesia saat ini ada dua jenis, yakni kacer hitam yang sering disebut kacer jawa dan kacer poci atau kacer sekoci yang sering disebut kacer sumatra. Burung ini memang masih berkerabat yakni sama-sama dalam *genus Copsychus.* Burung kacer jawa nama ilmiahnya adalah *Copsychus sechellarum* sedangkan kacer poci adalah *Copsychus saularis.*

Gambar 2.2 Burung Kacer ((omkicau.com),2016)

2.2 Karakteristik Penilaian Lomba Burung Berkicau

Ada 3 karakteristik penilaian dalam lomba burung berkicau yang harus di perhatikan, adapun 3 karakteristik penilaian tersebut adalah sebagai berikut: (sumber)

1. Irama
    a. Selalu aktif dalam berkicau atau berkicau lama dan tidak putus-putus.
    b. Memiliki banyak variasi lagu
    c. Tempo serasi
    d. Kombinasi yang tepat antara panjang dan pendeknya lagu.
    e. Suaranya merdu
    f. Konsisten dalam berkicau
2. Kualitas suara
    a. Suara bening atau bersih.
    b. Suara empuk dan nyaman didengarkan.
    c. Suaranya jauh dari cempreng
3. Fisik
    a. Kondisi tubuh harus sehat.
    b. Tidak ada cacat atau bekas luka.
    c. Bulunya lengkap, bersih dan rapi (tidak acak-acakan).
    d. Postur tubuh ideal (tidak terlalu gemuk atau tidak terlalu kurus).
    e. Berani dan tenang pada saat di lomba kan.

2.3 *Short Time Fourier Transform* (STFT)

Menurut Tulus Hayadi (2013), STFT (*Short Time Fourier Transform*) merupakan metode transformasi yang mengembangkan metode *Fourier Transform* dengan kelebihan pada kemampuan untuk mentransformasi *non-stationary signal*. Adapun ide dibalik metode ini adalah membuat *non-stationary signal* menjadi suatu representasi *stationary signal* dengan memasukkan suatu *window function*. Dalam hal ini, sinyal yang ada dibagi menjadi beberapa segmen dimana segmen yang didapatkan, diasumsikan terdiri dari *stationary signal*.

Adapun rumus yang digunakan dapat dilihat pada persamaan :

$$\boldsymbol{STFT}\{\boldsymbol{x}[\boldsymbol{n}]\}(\boldsymbol{m},\boldsymbol{\omega}) \equiv \boldsymbol{X}(\boldsymbol{m},\boldsymbol{\omega}) = \sum_{\boldsymbol{n}=-\infty}^{\infty} \boldsymbol{x}[\boldsymbol{n}]\boldsymbol{w}[\boldsymbol{n}-\boldsymbol{m}]\boldsymbol{e}^{-\boldsymbol{j}\boldsymbol{\omega}\boldsymbol{n}}$$

Keterangan:

$x[n]$ = sinyal masukan

n = waktu (sekon).

$w\,[n]$ = fungsi *windows*

ω = kecepatan sudut ($2\pi f$)

m = panjang *windows*

Perlu diperhatikan di sini bahwa x[n] adalah sinyal dengan domain waktu dan STFT{x[n]} adalah sinyal dengan domain frekuensi dan waktu. Karena itu, berbeda dengan *Fourier Transform*, STFT merupakan metode transformasi menghasilkan *Time-Frequency Representation* (TFR) dari sinyal. Di sini, w[n] adalah *window function* yang dapat mengambil bentuk distribusi normal dengan rumus berikut ini:

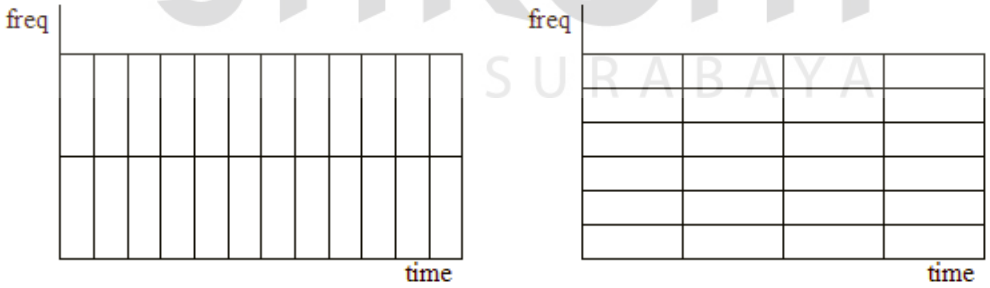
$$w[n] = e^{-a(n^2)/2}$$

Untuk menggambar spectrogram nya digunakan rumus

$$spectogram\{x(t)\}(\tau, \omega) = |X(\tau, \omega)|^2$$

Permasalahan yang muncul di sini adalah bahwa STFT menggunakan kernel window pada suatu interval waktu tertentu. Berbeda dengan *Fourier Transform* yang menggunakan kernel $e^{-2j\pi ft}$ sepanjang waktu, sehingga tidak ada permasalahan dalam hal resolusi frekuensi. Kalau STFT memilih *window* dengan lebar *infinity*, maka metode ini tidak akan ada bedanya dengan *Fourier Transform*. Gambar 2.3. menunjukan perbedaan dari kedua jenis window, dari ulasan yang singkat ini dapat diambil kesimpulan:

- *Window* sempit : mempunyai resolusi waktu yang bagus, tetapi resolusi frekuensi yang tidak bagus
- *Window* lebar : mempunyai resolusi frekuensi yang bagus, tetapi resolusi waktu yang tidak bagus



Sumber: (Tulus Hayadi, 2013)

Gambar 2.3 *Window* Sempit (Kiri) Dan *Window* Lebar (Kanan)

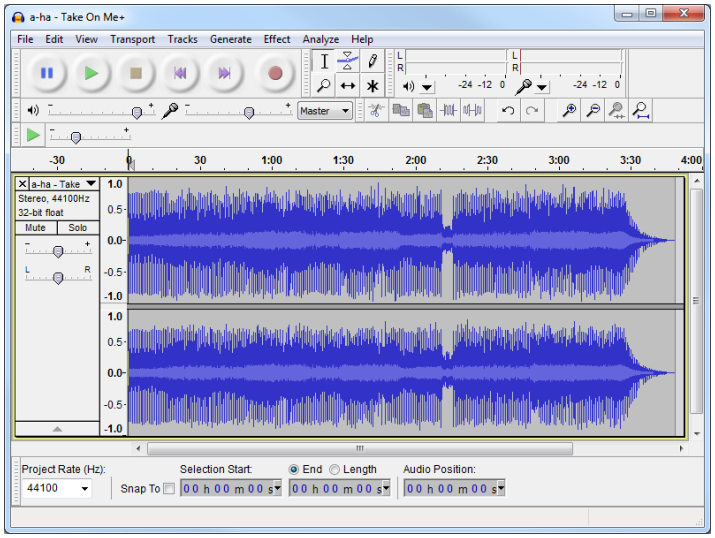
2.4 Audacity

Audacity adalah program yang memanipulasi bentuk gelombang audio digital. Selain rekaman suara langsung dari dalam program, aplikasi ini dapat mengimport banyak format file suara, termasuk WAV, AIFF, MP3, dan Ogg Vorbis. Format PCM dari 8,16,24 dan 32-bit juga dapat di-impor dan di-ekspor, Gambar 2.4. menunjukan contoh dari tampilan aplikasi Audacity. Hal yang dapat dilakukan aplikasi Audacity:

1. Potong, Salin, Tempel, Hapus, Diam, Duplikat, Split.
2. Berlaku plug-in efek untuk bagian suara.
3. Built-in editor amplop volume.
4. Spektrogram disesuaikan trek modus tampilan.
5. Analisis frekuensi window untuk aplikasi audio analisis.
6. Sederhana untuk operasi menyelaraskan kompleks untuk trek dan kelompok trek.

Meskipun Audacity adalah audio editor yang kuat untuk bekerja dengan jumlah trek yang tidak terbatas dan jumlah ukuran data yang hampir tidak terbatas, tetapi audacity tidak dapat melakukan semuanya. Hal yang tidak bisa di lakukan oleh audacity:

1. Audacity tidak dapat merekam lebih dari dua saluran sekaligus pada banyak sistem. Beberapa dukungan untuk rekaman multitrack termasuk dalam versi 1.2.3, tetapi tidak mendukung sistem sangat banyak belum.
2. Audacity membuka file MIDI, tetapi bukan editor MIDI, MIDI serta fitur yang sangat terbatas.



Gambar 2.4 Audacity

2.5 MATLAB (*Matrix Laboratory*)

Bahasa pemrograman sebagai media untuk berinteraksi antara manusia dan komputer saat dibuat semakin mudah dan cepat. Sebagai contoh, dapat dilihat dari perkembangan bahasa pemrograman Pascal yang terus memunculkan varian baru sehingga akhirnya menjadi Delphi, demikian pula dengan Basic dengan VisualBasicnya serta C dengan C + + Buildernya. Pada akhirnya semua bahasa pemrograman akan semakin memberikan kemudahan pemakainya (programmer) dengan penambahan fungsi fungsi baru yang sangat mudah digunakan bahkan oleh pemakai tingkat pemula.

MATLAB muncul di dunia bahasa pemrograman yang cenderung dikuasai oleh bahasa yang telah mapan. Tentu saja sebagai bahasa pemrograman yang baru MATLAB akan sukar mendapat hati dari pemakai. Namun MATLAB hadir tidak

dengan fungsi dan karakteristik yang umumnya ditawarkan bahasa pemrograman lain yang biasanya hampir seragam. MATLAB dikembangkan sebagai bahasa pemrograman sekaligus alat visualisasi, yang menawarkan banyak kemampuan untuk menyelesaikan berbagai kasus yang berhubungan langsung dengan disiplin keilmuan matematika. MATLAB memiliki kemampuan mengintegrasikan komputasi, visualisasi, dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. MATLAB menyediakan beberapa pilihan untuk dipelajari, mempelajari metode visualisasi saja, pemrograman saja, atau kedua-duanya. MATLAB adalah bahasa pemrograman level tinggi yang dikhususkan untuk komputasi teknis. Bahasa ini mengintegrasikan kemampuan komputasi, visualisasi, dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. MATLAB memberikan sistem interaktif yang menggunakan konsep array sebagai standar variabel elemennya tanpa membutuhkan pendeklarasian *array* seperti pada bahasa pemrograman lain.

Kehadiran MATLAB sebagai bahasa pemrograman memberikan jawaban sekaligus tantangan. MATLAB menyediakan beberapa pilihan untuk dipelajari, mempelajari metoda visualisasi saja, pemrograman saja, atau kedua-duanya. MATLAB memang dihadirkan bagi orang-orang yang tidak ingin disibukkan dengan rumitnya sintaks dan alur logika pemrograman, sementara pada saat yang sama membutuhkan hasil komputasi dan visualisasi yang maksimal untuk mendukung pekerjaannya. Selain itu, MATLAB juga memberikan kemudahan bagi programmer/developer program yaitu untuk menjadi pembanding yang

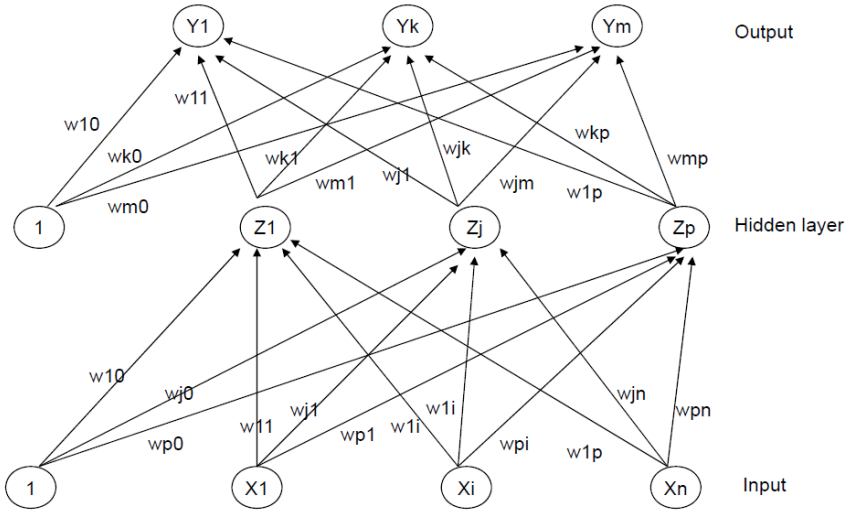
sangat handal, hal tersebut dapat dilakukan karena kekayaannya akan fungsi matematika, fisika, statistika, dan visualisasi.

2.6 Jaringan Saraf Tiruan *Backpropagation*

Perambatan galat mundur (*Backpropagation*) menurut Kiki (2004) adalah sebuah metode sistematik untuk pelatihan *multilayer* jaringan saraf tiruan. Metode ini memiliki dasar matematis yang kuat, obyektif dan algoritma ini mendapatkan bentuk persamaan dan nilai koefisien dalam formula dengan meminimalkan jumlah kuadrat galat *error* melalui model yang dikembangkan (*training set*).

1. Dimulai dengan lapisan masukan, hitung keluaran dari setiap elemen pemroses melalui lapisan luar.
2. Hitung kesalahan pada lapisan luar yang merupakan selisih antara data aktual dan target.
3. Transformasikan kesalahan tersebut pada kesalahan yang sesuai di sisi masukan elemen pemproses.
4. Propagasi balik kesalahan-kesalahan ini pada keluaran setiap elemen pemroses ke kesalahan yang terdapat pada masukan. Ulangi proses ini sampai masukan tercapai.
5. Ubah seluruh bobot dengan menggunakan kesalahan pada sisi masukan elemen dan luaran elemen pemroses yang terhubung.

Arsitektur Model *Backpropagation*



Sumber: (Jong, J.S., 2005)

Gambar 2.5 Arsitektur Model *Backpropagation* Menurut Jong, J.S.

Gambar 2.5 menunjukan Arsitektur Model *Backpropagation* menurut Jong, J.S.

Fungsi Aktivasi menurut Jong J.S:

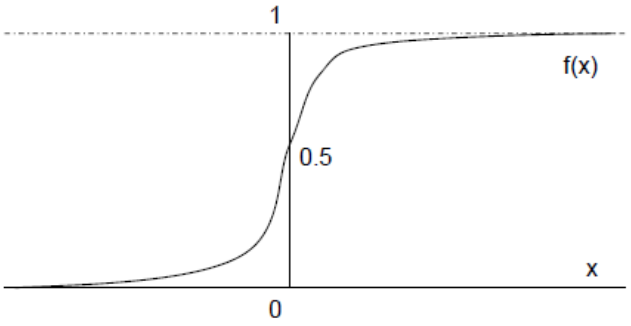
Syarat fungsi aktivasi yang dapat dipakai adalah *continue*, terdeferensial dengan mudah dan merupakan fungsi yang tidak turun.

Fungsi yang sering dipakai adalah :

- *sigmoid* biner yang memiliki range(0,1)

  Grafik fungsinya bisa di lihat pada gambar 2.6. Grafik fungsi aktivasi *sigmoid* biner.

  $f(x) = 1/(1 + e^{-x})$ dengan turunan $f'(x) = f(x)(1 – f(x))$



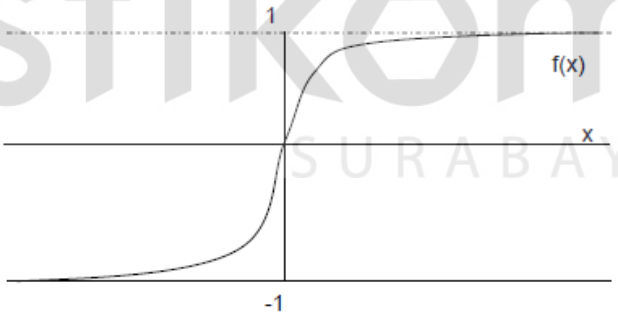
Sumber: (Jong, J.S., 2005)

Gambar 2.6 Grafik Fungsi Aktivasi *Sigmoid* Biner

- Fungsi *sigmoid* bipolar dengan *range* (1,-1)

  Grafik fungsinya bisa di lihat pada gambar 2.7. Grafik fungsi aktivasi *sigmoid* bipolar.

  $f(x) = 2/(1 + e^{-x})-1$, $f'(x) = (1+f(x))(1-f(x))/2$



Sumber: (Jong, J.S., 2005)

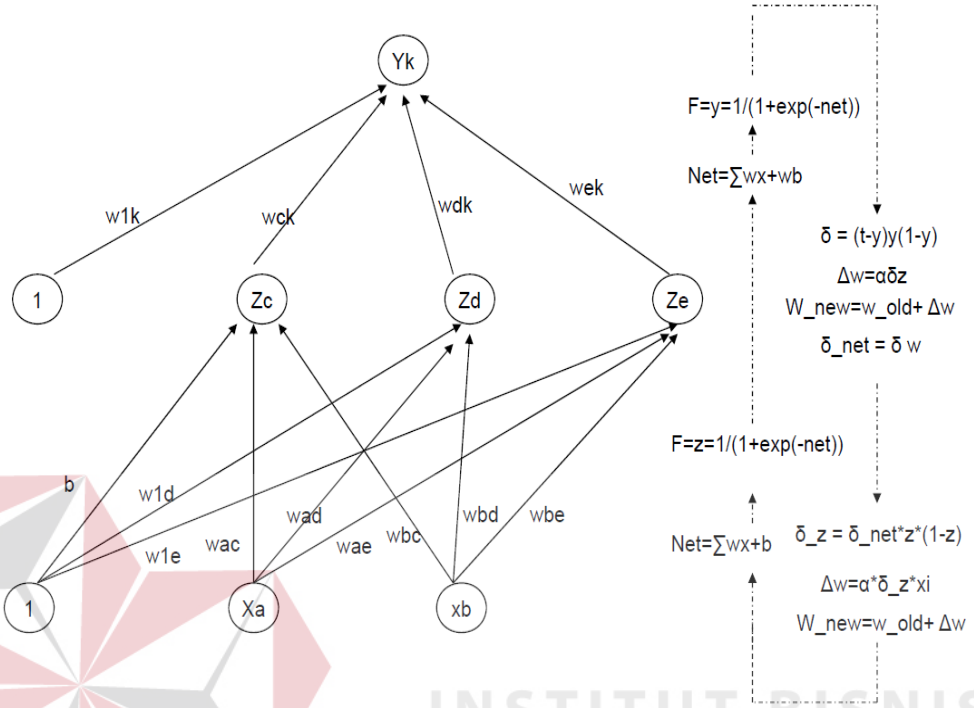
Gambar 2.7 Grafik Fungsi Aktivasi *Sigmoid* Bipolar

Adapun pelatihan standar *backpropagation* menurut Jong, J.S. adalah:

- Meliputi 3 fase, maju, mundur, dan modifikasi bobot

- Fase I Propagasi maju, sinyal masukan(x_i) dipropagasikan ke *hidden layer* menggunakan fungsi aktivasi yang ditentukan. Keluaran dari setiap unit *hidden*(z_j) selanjutnya dipropagasikan maju lagi ke layar *hidden* diatasnya menggunakan fungsi aktivasi yang ditentukan, demikian seterusnya hingga menghasilkan keluaran jaringan (y_k). Berikutnya, keluaran jaringan dibandingkan dengan target yang harus dicapai (t_k). Selisih t_k - y_k adalah kesalahan yang terjadi. Jika kesalahan ini lebih kecil dari batas toleransi maka iterasi dihentikan, tetapi bila kesalahan masih lebih besar maka bobot setiap garis dalam jaringan akan dimodifikasi untuk mengurangi kesalahan yang terjadi
- Fase II Propagasi mundur, Berdasarkan kesalahan t_k - y_k, dihitung faktor δ_k(k=1,2,3,..,m) yang dipakai untuk mendistribusikan kesalahan di unit y_k ke semua unit *hidden* yang terhubung langsung dengan y_k. δ_k juga dipakai untuk mengubah bobot garis yang berhubungan langsung dengan unit keluaran. Dengan cara yang sama, dihitung faktor δ_j di setiap unit di *hidden layer* sebagai dasar perubahan bobot semua garis yang berasal dari unit tersembunyi di layar di bawahnya. Demikian seterusnya hingga semua faktor δ di unit *hidden* yang berhubungan langsung dengan unit masukan dihitung
- Fase III Perubahan bobot, bobot semua garis dimodifikasi bersamaan. Perubahan bobot suatu garis didasarkan atas faktor δ neuron di layar atasnya. Sebagai contoh, perubahan bobot garis yang menuju ke layar keluaran didasarkan atas δ_k yang ada di unit keluaran. Fase tersebut diulang hingga penghentian terpenuhi. Umumnya kondisi penghentian yang dipakai adalah

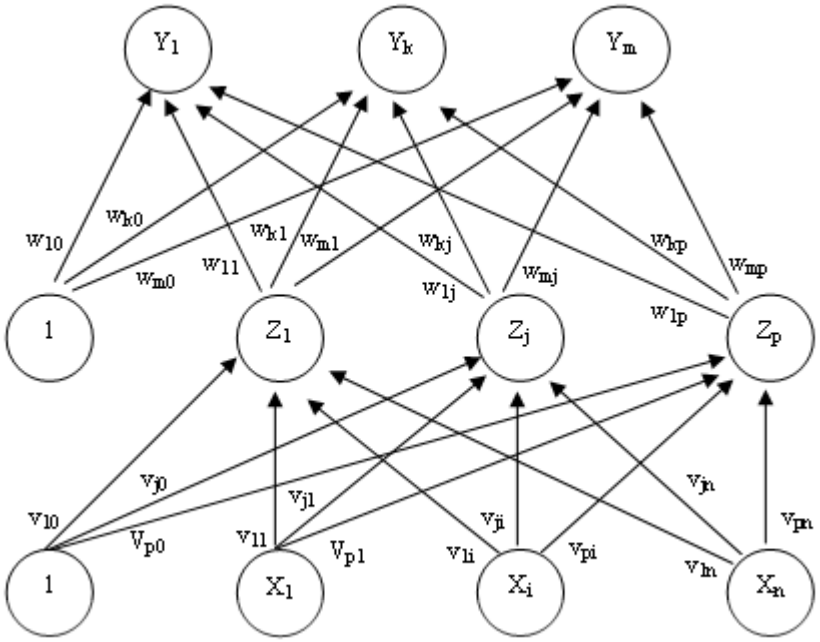
jumlah iterasi atau kesalahan. Gambar 2.8. menunjukan Algoritma *backpropagation* menurut Jong, J.S.



Sumber: (Jong, J.S., 2005)

Gambar 2.8 Algoritma *Backpropagation* Menurut Jong, J.S.

Backpropagation menurut Wirda Ayu Utari (2010), memiliki beberapa unit yang ada dalam satu atau lebih layar tersembunyi. Gambar 2.9. adalah arsitektur *backpropagation* menurut W.A. Utari dengan n buah masukan (ditambah sebuah bias), sebuah layar tersembunyi yang terdiri dari p unit (ditambah sebuah bias), serta m buah keluaran.



Sumber: (Utari, 2010)

Gambar 2.9 Arsitektur Model *Backpropagation* Menurut W.A. Utari

V_{ji} merupakan bobot garis dari unit masukan X_i ke unit layar tersembunyi Z_j (V_{j0} merupakan bobot garis yang menghubungkan bias di unit masukan ke unit layar tersembunyi zj). W_{kj} merupakan bobot dari unit layar tersembunyi Z_j ke unit keluaran V_k (w_{k0} merupakan bobot dari bias di layar tersembunyi ke unit keluaran Z_k).

Algoritma *backpropagation* menggunakan *error* keluaran untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan

menggunakan fungsi aktivasi yang dapat dideferensiasikan, seperti sigmoid, tansig atau purelin.

Adapun algoritma *backpropagation* menurut Wirda Ayu Utari (2010), adalah sebagai berikut:

a. Inisialisasi bobot (ambil bobot awal dengan nilai *random* yang cukup kecil).

b. Tetapkan maksimum *epoch*, target *error*, dan *learning rate* (α).

c. Inisialisasi *Epoch* = 0; MSE = 1.

d. Kerjakan langkah-langkah berikut selama epoch < maksimum *epoch* dan (MSE > target *error*):

   1. Epoch = Epoch + 12
   2. Untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan:

      Feedforward :

      a. Tiap-tiap unit masukan (x_i, i=1,2,3,...,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi).

      b. Tiap-tiap unit pada suatu lapisan tersembunyi (Z_j, j=1,2,3,...,p) menjumlahkan sinyal-sinyal masukan berbobot ($b1_j$, j=1,2,3,...n) :

$$z\_in_j = b1_j + \sum_{i=1}^{n} x_i v_{ij} \quad (1)$$

Gunakan fungsi aktivasi untuk menghitung sinyal keluarannya:

$$z_j = f(z\_in_j) \quad (2)$$

dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit keluaran).

Tiap-tiap unit keluaran (y_k, k=1,2,3,...,m) menjumlahkan sinyal-sinyal masukan berbobot:

$$y\_in_k = b2_k + \sum_{i=1}^{p} z_i w_{jk} \quad (3)$$

gunakan fungsi aktivasi untuk menghitung sinyal keluarannya:

$$y_k = f(y\_in_k) \quad (4)$$

dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit keluaran).

Catatan: Langkah (b) dilakukan sebanyak jumlah lapisan tersembunyi.

c. Tiap-tiap unit keluaran (y_k, k=1,2,3,...,m) menerima target pola yang berhubungan dengan pola masukan pembelajaran, hitung informasi errornya ($\delta 2_k$, k=1,2,3,...n):

$$\delta 2_k = (t_k - y_k) f'(y\_in_k) \quad (5)$$

$$\varphi 2_{jk} = \delta_k z_j \quad (6)$$

$$\beta 2_k = \delta_k \quad (7)$$

kemudian hitung koreksi bobot (Δw_{jk}) (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}):

$$\Delta w_{jk} = \alpha \varphi 2_{jk} \quad (8)$$

hitung juga koreksi bias ($\Delta b2_k$) (yang nantinya akan digunakan untuk memperbaiki nilai $b2_k$):

$$\Delta b2_k = \alpha \beta 2_k \quad (9)$$

langkah (d) ini juga dilakukan sebanyak jumlah lapisan tersembunyi, yaitu menghitung informasi *error* dari suatu lapisan tersembunyi ke lapisan tersembunyi sebelumnya.

d. Tiap-tiap unit tersembunyi (Z_j, j=1,2,3,...,p) menjumlahkan delta masukannya (dari unit-unit yang berada pada lapisan di atasnya):

$$\delta\_in_j = \sum_{k=1}^{m} \delta 2_k w_{jk} \quad (10)$$

kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi *error* ($\delta 1_j$, j=1,2,3,....n):

$$\delta 1_j = \delta\_in_j f'(z\_in_j) \quad (11)$$

$$\varphi 1_{ij} = \delta 1_j x_j \quad (12)$$

$$\beta 1_j = \delta 1_j \quad (13)$$

kemudian hitung koreksi bobot (Δv_{ij}) yang nantinya akan digunakan untuk memperbaiki nilai (v_{ij}):

$$\Delta v_{ij} = \alpha \varphi 1_{ij} \quad (14)$$

hitung juga koreksi bias yang nantinya akan digunakan untuk memperbaiki nilai ($\Delta b1_j$):

$$\Delta b1_j = \alpha \beta 1_j \quad (15)$$

e. Tiap-tiap unit keluaran (Y_k, k=1,2,3,...,m) memperbaiki bias dan bobotnya (j=0,1,2,...,p):

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (16)$$

$$b2_k(baru) = b2_k(lama) + \Delta b2_k \quad (17)$$

Tiap-tiap unit tersembunyi (Z_j, j=1,2,3,...,p) memperbaiki bias ($b1_j$) dan bobotnya (v_{ij}) (i=0,1,2,...,n) :

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (18)$$

$$b1_j(baru) = b1_k(lama) + \Delta b1_j \quad (19)$$

3. Hitung MSE (*Mean Square Error*)

   Menghitung nilai rata-rata kuadrat *error* (E = selisih target nilai dengan keluaran; n = banyak data).

$$MSE = \frac{\sum E^2}{n}$$