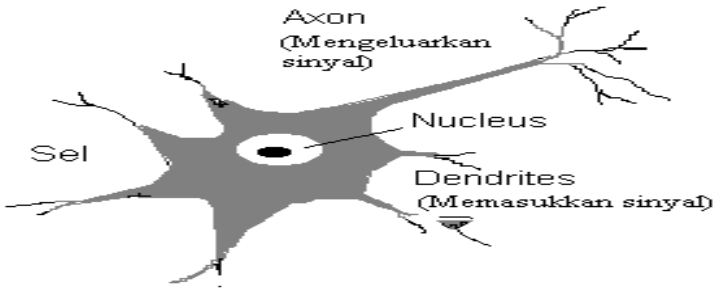
Bab 4
Model Neural Network

Model Jaringan Neural Buatan (JNB) merupakan suatu sistem yang dibangun berdasarkan cara kerja neural pada makhluk hidup (neural biologis). Neural biologis terdiri atas sel neural yang disebut ***neuron*** yang terdiri atas **dendrit** yang menerima sinyal dari neuron lain, **soma** yang meneruskan operasi matematis terhadap sinyal yang masuk, dan **axon** yang meneruskan sinyal ke neuron yang lain[1][4][6]. Otak merupakan sistem pengolah informasi atau komputer super kompleks, nonlinier dan paralel, mempunyai kemampuan untuk mengorganisasikan neuron sedemikian rupa sehingga dapat melakukan operasi-operasi komputasi (pengolahan pola, persepsi, dan sistem kendali). Contoh, sistem penglihatan manusia (human vision), fungsi dari sistem tersebut menyediakan suatu representasi dari lingkungan disekitar kita, dan yang lebih penting adalah menyediakan informasi yang kita butuhkan untuk berinteraksi dengan lingkungan. Biasanya otak kita secara rutin memenuhi tugas-tugas pengenalan citra (mengenali wajah seorang teman kita diantara sekian banyak wajah tak dikenal) dalam orde berkisar 100-200 milidetik, sedangkan komputer konvensional yang sangat canggih memerlukan waktu bisa sampai berhari-hari. Contoh lainnya, sistem sonar dari seekor kelelawar. Sonar adalah sebuah sistem temu-lokasi yang bekerja berdasarkan prinsip pemantulan (echo-location system). Selain menyediakan informasi tentang seberapa jauh suatu sasaran berada (seekor serangga yang sedang terbang), sistem sonar dari kelelawar juga menyampaikan informasi tentang kecepatan relatif (antara kelelawar tersebut dengan sasaran), ukuran dan beberapa ciri penting sasaran, serta sudut azimut (sudut pandang horisontal) dan elevasi (sudut pandang vertikal). Dibutuhkan suatu komputasi neural yang sangat rumit untuk mendapatkan semua informasi tersebut

dari gelombang yang dipantulkan tetapi semua proses komputasi tersebut dapat dilakukan dengan sebuah otak yang berukuran hanya sebesar kelereng.

Bagaimana otak manusia (mahluk hidup) dapat melakukannya? Otak kita mempunyai suatu struktur yang mampu untuk mengembangkan aturan-aturannya sendiri yang biasa disebut sebagai 'pengalaman'. Dibutuhkan waktu bertahun-tahun untuk pembentukan pengalaman tersebut, dan perkembangan pengalaman secara dramatis terjadi pada 2 tahun pertama sejak lahir. Selama proses perkembangan awal ini, milyaran synapse atau koneksi terbentuk.



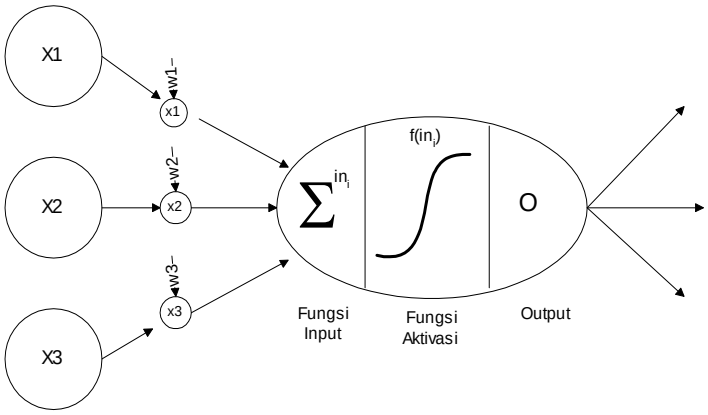
Gambar 4.1 Sel Neuron

4.1 Model Neuron

Model Jaringan Neural Buatan (JNB) adalah sebuah mesin yang dirancang untuk meniru cara kerja otak mehluk hidup dalam melakukan tugas tertentu. model jaringan tersebut diimplementasikan dalam perangkat lunak komputer digital. Untuk mencapai kinerja yang tinggi, JNB memberdayakan sistem interkoneksi yang cukup banyak dan rumit dari sebuah sel pemroses yang sederhana. Dapat pula kita memandang bahwa JNB sebagai sebuah mesin komputasi adaptif. JNB terdiri atas pemroses paralel terdistribusi yang mempunyai kecendrungan alami untuk menyimpan pengetahuan dan memanfaatkannya. Neural tersebut mirip dengan otak manusia dalam dua hal:

1. Neural mendapat pengetahuan melalui proses pembelajaran,

2. Pengetahuan hasil belajar tersimpan dengan baik.



Gambar 4.2 Model Neuron

Prosedur yang digunakan dalam proses pembelajaran disebut sebagai **algoritma pembelajaran**, yang fungsinya untuk memodifikasi bobot-bobot koneksi (synaptik) secara bertahap untuk mencapai tujuan perancangan yang diinginkan. Konsep JNB kemudian digeneralisasi menjadi suatu model matematis berdasarkan beberapa asumsi [1] :

1. Pemrosesan informasi dilakukan pada sejumlah elemen sederhana yang disebut neuron.
2. Neuron mengirimkan sinyal melalui suatu koneksitas.
3. Setiap koneksi memiliki bobotnya sendiri, yang biasanya menjadi faktor pengali dari sinyal yang ditransmisikan.
4. Setiap neuron menjalankan fungsi aktivasi pada masukkannya untuk menentukan sinyal keluarannya.

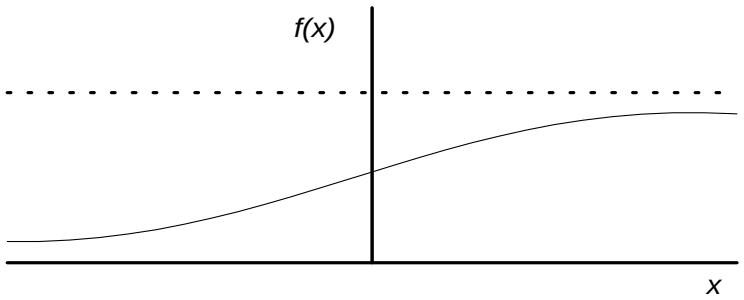
Representasi pengetahuan atau informasi pada JNB disimpan pada bobot – bobot koneksinya secara keseluruhan. Secara matematis, JNB berusaha mencari fungsi / pemetaan dari sejumlah sampel yang terdiri dari masukkan dan keluaran[3]. Gambar-2 merepresentasikan sebuah neuron yang menerima sinyal keluaran dari neuron - neuron X_1, X_2, dan X_3, yaitu masing – masing sinyal x_1, x_2, dan x_3. Bobot koneksi antara neuron tersebut masing – masing adalah w_1, w_2, dan w_3. Total masukkan untuk neuron tersebut, in_i, adalah jumlah dari perkalian sinyal dengan bobot koneksi dari neuron X_1, X_2, dan X_3, sehingga

$in_i = w_1x_1 + w_2x_2 + w_3x_3$

Keluaran neuron tersebut O, adalah fungsi dari total masukkan in_i, yaitu fungsi aktivasi $O = f(in_i)$. Dainatara fungsi aktivasi yang sering digunakan adalah fungsi sigmoid biner :

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

Sebuah fungsi aktivasi harus memiliki karakteristik kontinu, dapat



Gambar 4.3 Grafik Fungsi Sigmoid Biner

diturunkan, dan monoton naik [1].

4.2 Karakteristik Jaringan Neural Buatan (JNB)

Karakteristik suatu JNB ditentukan oleh 3 (tiga) hal penting [1]:

1. Arsitektur, antara lain :
   - Neural lapis tunggal, yang hanya memiliki satu lapis bobot koneksi yaitu antara unit masukkan dengan keluaran.
   - Neural lapis banyak, yang memiliki satu atau lebih lapis neuron di antara unit masukkan dan keluarannya.
2. Metode Pembelajaran, yaitu :
   - Pembelajaran dengan pengarahan, JNB diberi target keluaran yang harus dicapai.
   - Pembelajaran tanpa pengarahan, tidak melibatkan target keluaran.
   - Neural dengan bobot tetap.
3. Fungsi aktivasinya, berupa :
   - Fungsi identitas
   - Fungsi tangga biner
   - Fungsi sigmoid biner
   - Fungsi sigmoid bipolar

4.3 Keuntungan Model Jaringan Neural Buatan (JNB)

Rahasia kekuatan komputasi dari JNB antara lain: struktur paralel terdistribusi dengan padat, kemampuannya untuk belajar dan membuat ‘generalisasi’. Generalisasi adalah kemampuan JNB untuk menghasilkan keluaran yang diinginkan (sesuai dengan tujuan perancangan neural) untuk masukan yang tidak dilatihkan/dibelajarkan selama proses pembelajaran. Dua kemampuan pemrosesan informasi inilah yang membuat JNB dapat menyelesaikan permasalahan-permasalahan berskala besar.

Penggunaan JNB menawarkan beberapa sifat dan kapabilitas yang bermanfaat:

1. Non-linieritas : Sebuah neuron pada dasarnya berprilaku secara nonlinier. Konsekuensinya, sebuah JNB yang tersusun dari neuron-neuron yang saling terkoneksi, juga berprilaku nonlinier. Sifat nonlinieritas merupakan sifat yang sangat penting, karena pada umumnya mekanisme fisis yang mendasari terbentuknya sinyal masukan (sinyal suara) bagi JNB secara inheren bersifat nonlinier.

2. Pemetaan Masukan-Keluaran : Salah satu paradigma pembelajaran yang cukup populer adalah pembelajaran dengan pengarahan yang melibatkan modifikasi dari bobot-bobot synaptik dari JNB dengan mengaplikasikan satu set sampel pelatihan. Setiap sampel terdiri atas sebuah masukan dan sinyal keluaran yang dikehendaki. Sinyal masukan diambil secara acak untuk dimasukkan kedalam JNB, dan bobot-bobot synaptiknya diperbaharui sedemikian rupa sehingga kesalahan (selisih keluaran aktual dengan target) menjadi sekecil mungkin. Pelatihan JNB dilakukan berulang-kali untuk semua sampel hingga JNB mencapai keadaan stabil (tidak ada lagi perubahan yang berarti pada bobot-bobot synaptik). JNB telah belajar dari contoh (learning by examples) dengan membangun pemetaan masukan-keluaran dari sampel pelatihan. Pendekatan ini mirip dengan metode statistik non-parametrik. Dimana dalam metode tersebut, kita dapat membuat ‘estimasi’ dari pemetaan sejumlah data tanpa harus mengetahui/menghitung fungsi distribusinya.

3. Adaptifitas : JNB mempunyai kapabilitas untuk menyesuaikan bobot synaptiknya dengan perubahan yang terjadi pada lingkungan kerjanya. JNB yang telah dilatih untuk beroperasi pada lingkungan tertentu dapat dengan mudah di latih-ulang untuk menyesuaikan diri lagi pada perubahan kondisi lingkungannya.

Jika keadaan lingkungan berubah terus secara stokastik/acak, sebuah JNB dapat pula dirancang untuk merubah keadaan bobot-bobot synaptiknya secara real time (seketika).

4. Hasil yang dapat dibuktikan : Dalam konteks pengenalan pola, sebuah JNB dapat dirancang untuk menyediakan informasi tidak hanya tentang pola mana yang harus dipilih, tapi juga tentang tingkat confidence dalam keputusan yang diambil. Oleh karena itu, dapat menolak pola-pola yang kita ambiguity, sehingga akan meningkatkan kinerja pengenalan dari JNB.

5. Informasi kontekstual : Pengetahuan direpresentasikan secara langsung oleh struktur dan keadaan JNB. Setiap neuron di dalam JNB potensial untuk dipengaruhi oleh aktivitas global dari semua neuron-neuron di dalam sistem neural. Sehingga informasi kontekstual ditangani oleh JNB secara spontan/kontekstual.

6. Toleran kesalahan : Jika sebuah neuron atau koneksi synaptik padanya rusak, usaha untuk memanggil kembali pola yang tersimpan akan menemui kesalahan. Akan tetapi, karena watak distribusinya (JNB adalah pemroses paralel terdistribusi) JNB terbukti dapat mempertahankan tingkat pengenalan yang masih dapat diterima, jadi tidak terjadi kesalahan fatal.

7. Implementasi pada VLSI : Watak paralel terdistribusi padat membuat JNB sangat potensial untuk diimplementasikan pada teknologi VLSI (Very Large Scale Integrated) sehingga penggunaan JNB untuk aplikasi-aplikasi real-time seperti pengenalan pola, pemrosesan sinyal, dan sistem kendali sangat dimungkinkan.

8. Keseragaman dalam analisa dan perancangan : Pada dasarnya, JNB merupakan pemroses informasi yang bersifat serba-guna dan universal. Hal ini disebabkan

pengunaan notasi yang sama di dalam semua bahasan tentang JNB. Keseragaman ini dapat dimanifestasikan sebagai berikut:

- Neuron-neuron direpresentasikan dengan cara yang mirip untuk semua jenis JNB.
- Kemiripan tersebut memungkinkan JNB untuk berbagi teori maupun algoritma pembelajaran di dalam aplikasi JNB yang berbeda.
- JNB dapat dibangun dengan mudah secara modular, yakni dengan mengintegrasikan modul-modul yang terpisah.

9. Analogi dengan Neurobiologi : Perancangan dari JNB dimotivasi oleh analogi dengan otak manusia yang merupakan bukti hidup bahwa pemroses paralel yang dapat mentoleransi kesalahan terbukti tidak saja secara fisis mungkin dibuat tetapi juga terbukti cepat dan tangguh. Di kalangan ahli dan peneliti JNB hal ini memberikan harapan dan kepercayaan bahwa pemahaman secara fisis dari struktur neurobiologi memang dapat mempengaruhi perkembangan elektronika dan tentunya teknologi VLSI.

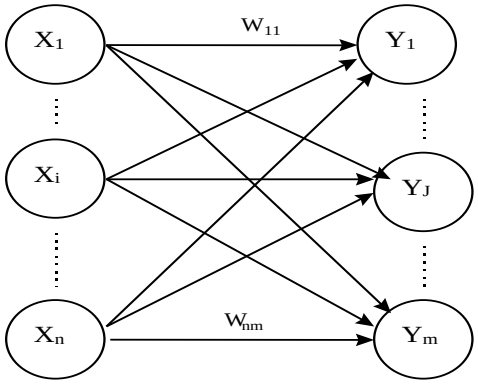
4.4 Model Jaringan Neural Buatan Lapis Tunggal

Salah satu jenis JNB lapis tunggal yang paling sering digunakan adalah Self-Organizing Map (SOM). SOM merupakan JNB yang banyak diterapkan pada persoalan klastering dengan kemampuan melakukan pembelajaran dalam tempo yang singkat. Dalam persoalan klastering, swa-organisasi standar tidak mengenal kemampuan beradaptasi dengan pola masukan. Setiap pola masukan akan dipaksa untuk masuk ke salah satu klaster yang telah ada walaupun sebenarnya pola masukan baru jauh dari klaster yang ada.

4.5 Arsitektur Jaringan Neural Swa-Organisasi

Neural swa-organisasi merupakan neural lapis tunggal yang mempunyai satu lapis hubungan bobot. Neuron dapat dibedakan atas neuron masukan yaitu neuron yang menerima sinyal dari luar dan neuron keluaran yang merupakan neuron respon dari neural . Arsitektur neural lapis tunggal dapat ditunjukkan seperti Gambar 4.4.

Pada neural swa-organisasi standar, jumlah neuron di lapis keluaran ditentukan lebih dulu, dan tergantung pada jumlah klaster yang diinginkan. Setiap vektor masukan dipaksa untuk menjadi salah satu anggota dari neuron keluaran yang ada.



Gambar 4.4 Arsitektur Neural Swa-Organisasi

4.6 Pembelajaran Model Jaringan Neural Swa-Organisasi

Pembelajaran pada neural swa-organisasi merupakan pembelajaran kompetisi. Dalam pembelajaran ini, neuron-neuron keluaran berkompetisi diantara mereka untuk menjadi neuron pemenang. Berbeda dengan pembelajaran supervisi dimana semua neuron keluaran aktif bersama-sama, pada pembelajaran kompetisi hanya satu neuron yang aktif pada satu saat.

Setiap neuron khusus mempelajari suatu set pola tertentu dan menjadi pendeteksi karakteristik pola tersebut. Misalkan neuron yang menjadi pemenang adalah neuron ke-j, aktivitas neuron (v_j) lebih besar dibanding neuron lainnya untuk suatu pola masukan x. Jika sebuah neuron tidak merespon terhadap suatu pola

tertentu (neuron yang kalah), pembelajaran tidak dilakukan pada neuron tersebut. Misalkan w_{ij} menyatakan bobot dari neuron masukan i ke neuron keluaran j, maka koreksi bobot (Δw_{ij}) yang diterapkan pada bobot w_{ij} adalah :

$$\Delta w_{ji} = \begin{cases} \alpha(x_i - w_{ji}) & \text{jika neuron ke j menang} \\ 0 & \text{jika neuron ke j kalah} \end{cases}$$

Konvergensi neural dicapai dengan cara menurunkan kecepatan pembelajaran (α) neural setiap epohnya (round). Karena hanya ada satu neuron yang aktif pada satu saat, maka ditentukan vektor masukan x yang sesuai dengan vektor bobot w_j dengan cara mencari jarak yang paling dekat antara vektor bobot dan vektor masukan. Cara untuk menentukan jarak yang paling kecil bermacam-macam, diantaranya *inner product* dan *jarak euclidean*. Nilai inner product ($w^T_j x$) untuk j = 1, 2. , N yang paling maksimum menyatakan similaritas atau kemiripan yang paling tinggi antara vektor masukan dengan pusat klaster yang dinyatakan dengan vektor bobot. Hasil inner product menentukan pada neuron mana proses komputasi nilai aktivasi akan dilakukan. Proses mencari nilai inner product yang paling *maksimum* setara dengan proses mencari nilai euclidean yang paling *minimum*. Setiap vektor masukan dilokalisasi pada suatu wilayah tertentu. Misalkan vektor masukan adalah:

$$x_i = [x_1, x_2 ... x_p]^T \quad p = \textit{jumlah neuron masukan}$$

Vektor bobot dari neuron masukan ke neuron keluaran dinyatakan dengan

$$w_{ij} = [w_{i1}, w_{i2} ... w_{in}]^T, \quad n = \textit{jumlah neuron keluaran}$$

Pada neural swa-organisasi standar biasanya digunakan jarak euclidean yang paling minimum (minimum-euclidean distance norm). Misalkan i(x) menyatakan neuron yang paling sesuai untuk vektor masukan x, secara matematis dapat dinyatakan :

$$i(x) = \arg_j \min \left\| x(n) - w_j \right\|, \qquad j = 1,2,\ldots,N$$

dimana ||.|| menyatakan euclidean norm dari argumen vektor.

4.7 Algoritma Model jaringan Swa-Organisasi

Step 0: Inisialisasi bobot-bobot w_{ij}

Step 1: Jika kondisi henti gagal, lakukan Step 2-8.

Step 2: Untuk setiap vektor masukan x, lakukan Step 3-5

Step 3: Untuk setiap j, hitung:

$$D(j) = \sum_i \left(w_{ij} - x_i\right)^2$$

Step 4: Temukan indeks j sehingga D(j) minimum

Step 5: Untuk semua neuron j dengan tetangga J dan semua i:

$$w_{ij}(baru) = w_{ij}(lama) + \alpha\left[x_i - w_{ij}(lama)\right]$$

Step 6: Perbaiki kecepatan pembelajaran

Step 7: Kurangi radius ketetanggaan pada waktu yang ditentukan

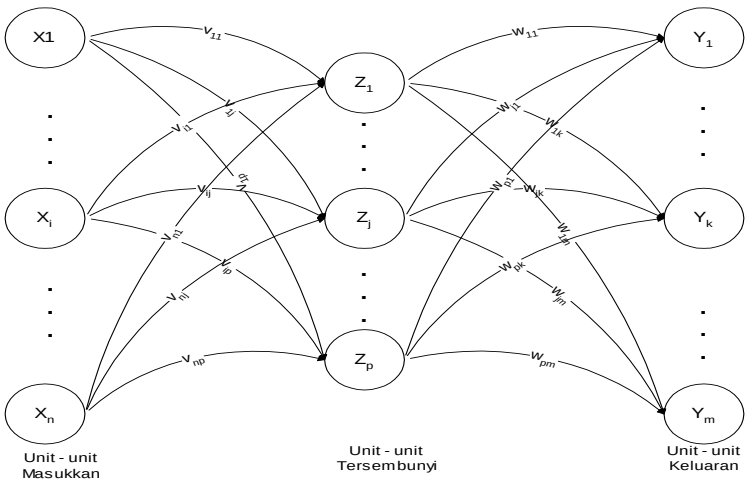
Step 8: Tes kondisi henti

4.8 JNB Lapis Banyak Feed-Forward

Salah satu jenis JNB yang paling sering digunakan adalah JNB lapis banyak *feed-forward*. JNB ini dapat memecahkan permasalahan yang lebih rumit daripada yang dapat dipecahkan lapis tunggal[1]. JNB jenis ini memiliki satu atau lebih lapis neuron di antara unit masukkan dan unit keluarannya. Lapis tambahan ini disebut dengan lapis tersembunyi (hidden). Adanya unit tersembunyi memungkinkan representasi yang lebih kompleks. Setiap unit tersembunyi menyimpan representasi internal dari neural [4].

Sinyal yang ditransmisikan pada JNB lapis banyak *feed-forward* merambat maju mulai dari lapis masukkan, lapis tersembunyi, terus menuju lapis keluaran. Arah perambatan sinyal pada pengoperasian JNB lapis banyak *feed-forward* selalu maju (satu arah) sesuai dengan koneksi yang ada. Pada JNB ini tidak terdapat koneksi yang arahnya berbeda dari yang lain atau membentuk siklik. Oleh karena itu JNB ini disebut neural *feed-forward*.

Dengan kemampuannya memecahkan persoalan yang rumit, JNB lapis banyak dapat disebut sebagai sebuah Universal Approximators karena JNB jenis ini dapat mempelajari dan merepresentasikan sembarang fungsi pemetaan yang kontinu[1]. Walaupun memiliki kemampuan yang lebih dibandingkan lapis tunggal, JNB lapis banyak feed-forward relatif lebih sulit dilatih. Beberapa ilmuwan telah berhasil menemukan sebuah metode umum yang efektif untuk melatih JNB lapis banyak yang adalah pelatihan propagasi balik atau aturan delta tergeneralisasi (generalized delta rule)[1].



Gambar 4.5 : JNB Lapis Banyak Feed-Forward

Metode propagasi balik mengubah himpunan bobot pada JNB waktu pelatihan step demi step menuju tingkat kesalahan yang minimum[6][3]. Proses ini dapat diinterpretasi sebagai gerakan turun berdasarkan derajat kemiringan (*gradien descent*). Kemiringan yang dimaksud adalah kemiringan pada kurva atau permukaan kesalahan yang merupakan fungsi dari seluruh bobot – bobot yang ada pada neural [6].

Pada dasarnya, propagasi balik menyediakan suatu cara untuk melakukan pembagian kalkulasi dari gradien di antara unit – unit pada neural, sehingga perubahan di tiap bobot dapat dihitung oleh masing – masing unit, dengan hanya memanfaatkan informasi lokal[6].

Perubahan bobot pada suatu koneksi didasari oleh kontribusi kesalahan yang diberikan oleh koneksi tersebut sesuai $-\frac{\partial E}{\partial w}$ dengan bobot yang dimilikinya. Dengan demikian perubahan bobot suatu koneksi proposional dengan , dimana *E* adalah kesalahan dan *w* adalah bobot[3][6].

Pelatihan JNB dengan propagasi balik meliputi tiga tahapan, yaitu:

1. Tahap *feedforward* dari pola pelatihan masukkan. Pada tahap ini sinyal masukkan ditransmisikan maju mulai dari lapis masukkan, lapis tersembunyi hingga ke lapis keluaran untuk menghasilkan keluaran dari neural.
2. Tahap propagasi balik dari kesalahan yang dihasilkan neural berdasarkan perhitungan antara target dengan keluaran aktual.
3. Tahap pengubahan bobot – bobot.

Pada tahap *feedforward*, setiap unit masukkan (X_i) menerima sinyal masukkan dan meneruskannya ke setiap unit di lapis tersembunyi Z_1 … $Z_{p.}$ Setiap unit tersembunyi kemudian menghitung aktivasinya dan mengirimkannya z_j ke setiap unit keluaran. Setiap unit keluaran (Y_k) lalu menghitung aktivasinya (y_k) sebagai keluaran dari neural .

Pada tahap pelatihan, setiap unit keluaran membandingkan aktivasi yang telah dihitungnya y_k dengan nilai target t_k untuk menentukan kesalahan masing – masing. Berdasarkan kesalahan ini, faktor δ_k (k = 1 … m) dihitung. δ_k ini digunakan untuk mendistribusikan kesalahan pada unit keluaran Y_k kembali ke semua unit di lapis sebelumnya (unit – unit tersembunyi). δ_k juga kemudian digunakan untuk mengubah bobot – bobot antara lapis keluaran dan lapis tersembunyi. Dengan cara yang sama, faktor δ_j (j = 1 … p) dihitung untuk setiap unit tersembunyi Z_j. Nilai kesalahan ini tidak perlu dipropagasikan ke lapis masukkan, tetapi digunakan untuk mengubah bobot – bobot antara lapis tersembunyi dan lapis masukkan.

Setelah semua faktor δ telah ditentukan, bobot – bobot di semua lapis disesuaikan secara simultan. Penyesuaian bobot w_{jk} (dari unit tersembunyi Z_j ke unit keluaran Y_k) dilakukan berdasarkan faktor δ_k dan nilai aktivasi unit tersembunyi Z_j, yaitu z_j. $-\frac{\partial E}{\partial w}$ Penyesuaian bobot v_{ij} (dari unit masukkan X_i menuju unit Z_j) dilakukan berdasarkan faktor δ_j dan nilai aktivasi dari unit masukkan, yaitu x_i. Faktor δ dan nilai aktivasi tersebut adalah komponen dari rumus .

Secara lebih formal dan jelas, algoritma pelatihan JNB dengan metode propagasi balik adalah sebagai berikut :

Step 0 Inisialisasi bobot – bobot (dengan nilai acak yang kecil).

Step 1 Selama kondisi berhenti belum terpenuhi, jalankan step 2 – 9.

Step 2 Untuk setiap pasangan data pelatihan, jalankan step 3 – 8

Feedforward :

Step 3 Setiap unit input (X_i, $i = 1, \ldots, n$) menerima sinyal masukkan x_i dan menyebarkan sinyal tersebut ke lapis tersembunyi.

Step 4 Setiap unit tersembunyi (Z_j, $j = 1, \ldots, p$) menjumlahkan sinyal

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

masukkan yang telah diberi bobot, lalu menjalankan fungsi aktivasinya untuk menghitung sinyal keluarannya, $z_j = f(z\_in_j)$, dan mengirimkan sinyal tersebut ke semua unit di lapis keluaran.

$$z\_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

Step 5 Setiap unit keluaran (Y_k, $k = 1, \ldots, m$) menjumlahkan sinyal masukkannya yang telah diberi bobot, lalu menjalankan fungsi aktivasinya untuk menghitung sinyal keluarannya, $y_k = f(y\_in_k)$.

$$y\_in_k = w_{ok} + \sum_{k=1}^{p} z_j w_{jk}$$

Propagasi kesalahan :

Step6 Setiap unit keluaran (Y_k, $k = 1, \ldots, m$) menerima sebuah pola target yang bersesuaian dengan pola pelatihan masukkan, lalu menghitung informasi kesalahan, $\delta_k = (t_k - y_k)\, f'(y\_in_k)$, menghitung koreksi bobotnya (untuk mengubah bobot w_{jk} nanti), $\Delta w_{jk} = \alpha\, \delta_k\, z_j$, dan menghitung koreksi biasnya (untuk mengubah w_{ok}), $\Delta w_{ok} = \alpha\, \delta_k$, lalu mengirimkan δ_k ke unit – unit di lapis sebelumnya.

Step 7 Setiap unit tersembunyi (Z_j, $j = 1, \ldots, p$) menjumlahkan masukkan – masukkan deltanya (dari unit – unit di lapis sesudahnya), mengalikannya dengan turunan dari fungsi aktivasinya untuk menghitung nilai kesalahannya, $\delta_j = \delta\_in_j\, f'(z\_in_j)$ menghitung koreksi bobot (untuk mengubah v_{ij} kemudian), $\Delta v_{ij} = \alpha\, \delta_j\, x_i$, dan menghitung koreksi biasnya (untuk mengubah v_{oj}), $\Delta v_{oj} = \alpha \delta_j$.

Pengubahan bobot – bobot dan bias :

Step 8 Setiap unit keluaran (Y_k, $k = 1, \ldots, m$) mengubah bias dan bobot – bobotnya ($j = 0, \ldots, p$) : $w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$. Setiap unit tersembunyi (Z_j, $j = 1, \ldots, p$) mengubah bias dan bobot – bobotnya ($i = 0, \ldots, n$) : $v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$.

Step 9 Pengujian kondisi berhenti.

4.9 Parameter dan Faktor yang Mempengaruhi Pembelajaran

Parameter dan faktor yang mempengaruhi keberhasilan proses pembelajaran dan kinerja neural adalah :

1. Inisialisasi bobot

   Nilai awal bobot – bobot pada neural akan menentukan apakan neural akan mencapai suatu tingkat kesalahan minimum secara global atau hanya lokal saja. Selain itu, bobot awal juga menentukan kecepatan neural menuju konvergensi.

   Pada umumnya, nilai awal bobot dipilih secara acak pada suatu interval tertentu (-0,5 hingga 0,5 atau –1 hingga 1).

   Nguyen dan Widrow berhasil melakukan modifikasi pada inisialisasi acak sehingga proses pembelajaran berjalan lebih cepat. Modifikasi dilakukan pada inisialisasi bobot – bobot antara unit masukkan dengan unit tersembunyi[1]. Definisi yang dipakai :

   n : jumlah unit – unit masukkan,

   p : jumlah unit – unit tersembunyi,

   β faktor skala : $\beta = 0{,}7\ (p)^{1/n}$

   Prosedur :

   Untuk setiap unit tersembunyi ($j = 1, \ldots, p$) :

   Inisialisasi vektor bobotnya (dari unit – unit masukkan) :

   v_{ij}(lama) = bilangan acak antara –0,5 dan 0,5 (antara $-\gamma$ dan γ).

Hitung normal $\mathbf{v}_j$, $\left\|\mathbf{v}_j(lama)\right\| = \sum_i v_{ij}$

Update kembali $v_{ij} = \frac{\beta v_{ij}(lama)}{\left\|\mathbf{v}_j(lama)\right\|}$ bobot :

Inisialisasi bias :

v_{oj} = bilangan acak antara –β dan β.

2. Laju pembelajaran

   Laju pembelajaran menentukan besar step perubahan bobot pada setiap iterasi pembelajaran. Laju pembelajaran yang terlalu kecil akan menyebabkan proses pembelajaran berjalan lambat, sedang laju yang terlalu besar dapat menyebabkan pembelajaran menjadi tidak stabil dan nilai kesalahan minimum terlewati (*overshoot*)[1].[4]. Besar laju pembelajaran bergantung pada kasus yang akan dihadapi, tetapi biasanya berupa bilangan kecil antara 0,05 – 0,25[3]. Pada propagasi balik, laju pembelajaran ini diberi simbol α.

3. Momentum

   Momentum digunakan untuk mempercepat konvergensi neural . Pada perhitungan perubahan bobot, ditambahkan fraksi dari perubahan bobot sebelumnya, sehingga perubahan bobot memiliki arah yang merupakan kombinasi antara gradien yang sekarang dengan yang terdahulu[1][3].

   Dengan demikian rumus perubahan bobot menjadi :

$$\Delta w(t) = -\alpha \frac{\partial E}{\partial w}(t) + \beta \Delta w_{(t-1)},$$

β = momentum.

4. Ukuran neuron

   Ukuran neural harus ditentukan sedemikian rupa agar kinerja neural optimal. Neural yang terlalu besar dapat menyebabkan rendahnya kemampuan generalisasi, sedang neural yang terlalu kecil dapat menyebabkan sulitnya neural mencapai tingkat ketelitian yang diinginkan[3][4].