BAB 2
LANDASAN TEORI

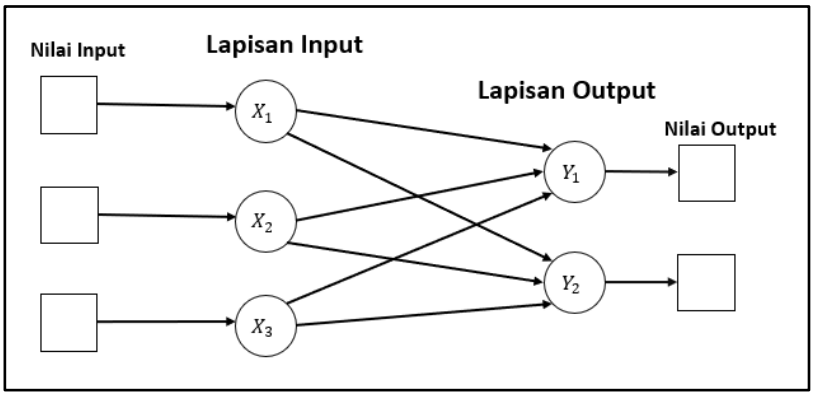
Pada bagian ini berisi teori-teori penunjang dalam pembuatan tugas akhir ini. Sistem ini menggunakan berbagai teori-teori sebagai pustaka untuk memperkaya proses penelitian.

2.1. *Neural Network*

Neural network didefinisikan sebagai sistem komputasi yang arsitektur dan operasinya terinspirasi dari pengetahuan tentang sel saraf biologis di dalam otak manusia (Astuti, E. D. 2009). *Neural network* atau jaringan syaraf tiruan dikembangkan sebagai model matematika yang menyamai pola pikir manusia atau jaringan syaraf makhluk hidup.

Neuron-neuron dalam *neural network* disusun dalam lapisan-lapisan (*layer*) dan mempunyai pola keterhubungan yang terdapat pada antar lapisannya. Susunan dari neuron-neuron dan pola keterhubungan dalam serta antar lapisannya disebut dengan arsitektur jaringan. Arsitektur yang sering digunakan dalam jaringan syaraf tiruan antara lain (Ardhiyanta, 2016):

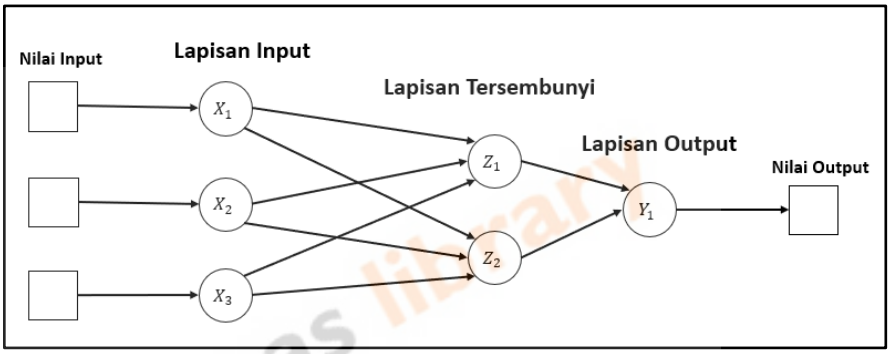
1. Jaringan layar tunggal (*single layer network*) adalah sekumpulan *input neuron* atau unit yang dihubungkan langsung dengan sekumpulan *output*. Dalam jaringan ini, semua unit *input* dihubungkan dengan semua unit *output*. Tidak ada unit input yang dihubungkan dengan unit input lainnya, demikian pula dengan unit output.



Gambar 2.1 Arsitektur Layer Tunggal

Pada gambar 2.1 merupakan gambar dari arsitektur layer tunggal dengan data masukkan yaitu X_1, X_2 dan X_3. Sedangkan untuk data keluaran yaitu Y_1, Y_2.

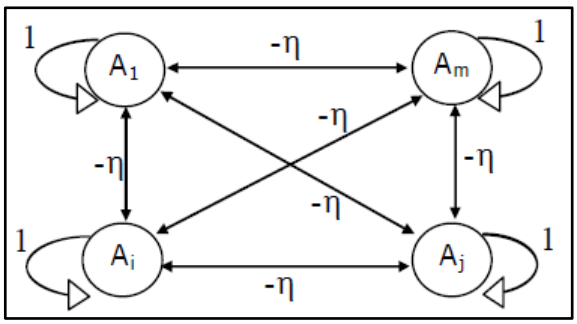
2. Jaringan layar jamak (*multi layer network*) merupakan perluasan dari layar tunggal. Dalam jaringan ini, selain unit *input* dan *output* terdapat juga unit layar tersembunyi atau *hidden layer*. Jaringan layar jamak dapat menyelesaikan masalah yang lebih kompleks dibandingkan dengan jaringan layar tunggal, walaupun untuk proses pelatihannya memakan waktu yang cukup lama (Najwa et al., 2017).



Gambar 2.2 Arsitektur Layar Jamak

Gambar 2.2 merupakan arsitektur layar jamak, dengan masukkan yaitu X_1, X_2 dan X_3 pada lapisan input, lapisan tersembunyi terdapat Z_1 dan Z_2, serta lapisan output dengan Y_1.

3. Model jaringan recurrent mirip dengan jaringan layar tunggal maupun jamak. Hanya saja, pada jaringan ini terdapat simpul keluaran yang memberikan sinyal pada unit masukan, sering juga disebut dengan feedback loop.



Gambar 2.3 Arsitektur Layar Kompetitif

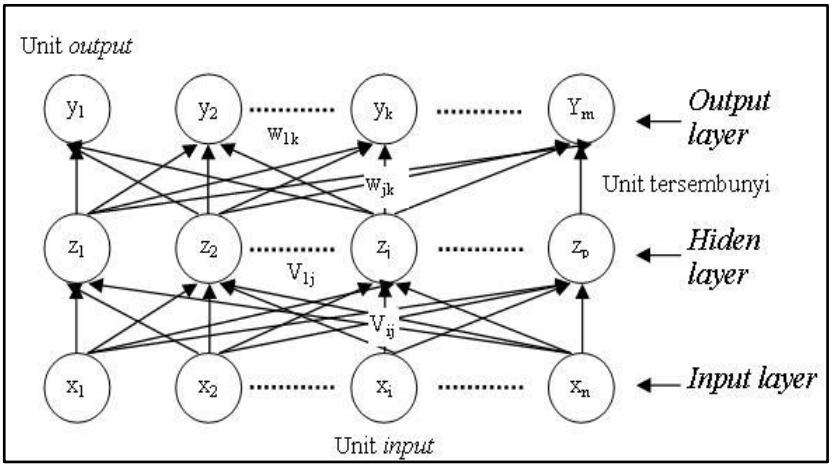
Pada gambar 2.3 merupakan arsitektur layar kompetitif yang mempunyai satu lapisan *neuron* tunggal dengan masing-masing *neuron* memberikan kembali keluarannya sebagai input (X) pada semua *neuron* yang lain.

2.2. *Algoritma Backpropagation*

Algoritma yang paling populer pada jaringan syaraf tiruan adalah algoritma *backpropagation*. Algoritma pelatihan *backpropagation* atau yang diterjemahkan menjadi propagasi balik pertama kali dirumuskan oleh Paul Werbos pada tahun 1974 dan dipopulerkan oleh Rumelhart bersama McClelland untuk dipakai pada *neural network* (Astuti, E. D. 2009).

Backpropagation merupakan suatu teknik pembelajaran atau pelatihan jenis *supervised learning* yang sering digunakan oleh *perceptron* dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan unit-unit yang ada pada lapisan tersembunyi. Setiap perubahan bobot yang terjadi dapat mengurangi *error*. Siklus perubahan bobot (*epoch*) dilakukan pada setiap set pelatihan sehingga kondisi berhenti dicapai, yaitu bila mencapai jumlah *epoch* yang diinginkan atau hingga sebuah nilai ambang yang ditetapkan terlampaui.

Metode ini merupakan salah satu metode yang sangat baik dalam menangani masalah pengenalan pola yang kompleks. Arsitektur backpropagation termasuk dalam jaringan layar jamak (Ardhiyanta, 2016).



Gambar 2.4 Arsitektur Backpropagation

Sumber: Almas, Setiawan et al., 2018

Pada gambar 2.4 terdapat satu lapisan input (*input* layer) yang terdiri dari X_1 sampai dengan X_n. Lapisan tersembunyi atau *hidden* layer juga terdapat pada arsitektur *backpropagation* yang terdiri dari z_1 sampai dengan z_p. Sedangkan pada lapisan keluaran atau *output* layer terletak pada lapisan terakhir di arsitektur *backpropagation* dengan nilai y_1 sampai dengan y_m.

Algoritma pelatihan *backpropagation* terdiri dari dua tahapan yaitu *feedforward* dan *backward*. Algoritma ini 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. Saat perambatan maju, unit-unit diaktifkan dengan menggunakan fungsi aktivasi yang dapat dideferensiasikan seperti sigmoid:

$$\boxed{y = \mathrm{f(x)} = \frac{1}{1 + e^{-\sigma x}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

$$\boxed{\mathbf{f'(x) = \boldsymbol{f}(\boldsymbol{x})[1 - f(x)]}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Keterangan:

y : Fungsi aktivasi

f(x) : Fungsi sigmoid biner

f'(x) : Turunan fungsi sigmoid biner

Atau seperti sigmoid bipolar:

$$\boxed{y = \mathrm{f(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

$$\boxed{\mathrm{f'^{(x)}} = [1 + \mathrm{f(x)}][1 - \mathrm{f(x)}]} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

Keterangan:

y : Fungsi aktivasi

f(x) : Fungsi sigmoid bipolar

f'(x) : Turunan fungsi sigmoid bipolar

Pelatihan *Backpropagation* dilakukan melalui langkah-langkah berikut ini:

Langkah ke-0 : Inisialisasi bobot;

Langkah ke-1 : Selama kondisi berhenti bernilai salah, kerjakan Langkah 2-9;

Langkah ke-2 : Untuk setiap data training, lakukan langkah 3-8.

Umpan Maju (*Feedforward*)

Langkah ke-3 : Setiap unit input (X_i, i= 1,…n) menerima sinyal input dan menyebarkan sinyal tersebut ke seluruh hidden unit

Langkah ke-4 : Pada setiap hidden unit ($Z_j, j = 1, \dots, p$) menjumlahkan sinyal-sinyal input yang sudah berbobot (termasuk biasnya)

$$\boxed{z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}} \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots(5)$$

Keterangan:

z_{in_j} : *Hidden* unit input ke-j

v_{0j} : Bias untuk hidden unit ke-j

x_i : Data training input x

v_{ij} : Bobot antara unit input ke-i dengan hidden unit ke-j

Lalu menghitung sinyal output dari hidden unit dengan menggunakan fungsi aktivasi yang telah ditentukan:

$$\boxed{z_j = f(z_{in_j})} \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots(6)$$

Keterangan:

z_j : Hidden unit ke-j

f : Fungsi aktivasi

z_{in_j} : *Hidden* unit ke-j

Sinyal output ini selanjutnya dikirim ke seluruh unit pada unit atas (unit output).

Langkah ke-5 : Tiap-tiap unit output $(Y_k, k = 1, \dots, m)$ menjumlahkan bobot sinyal input:

$$y_{-in_k} = w_{0k} + \sum_{i=1}^{n} z_i w_{jk} \quad \text{.....(7)}$$

Keterangan:

y_{-in_k} : Unit *output*

w_{0k} : Bias untuk unit output ke-k

z_i : Hidden unit ke-i

w_{jk} : Bobot antara hidden unit ke-j dengan unit output ke-k

Lalu menghitung sinyal output dari unit output bersangkutan dengan menggunakan fungsi aktivasi yang telah ditentukan

$$y_k = f(y_{in_k}) \quad \text{.....(8)}$$

Keterangan:

y_k : Unit output ke-k

f : Fungsi aktivasi

y_{in_k} : Unit output ke-k

Sinyal output ini selanjutnya dikirim ke seluruh unit pada output.

Umpan Mundur/Propagasi Error (*Backpropagation of Error*)

Langkah ke-6 : Setiap unit output $(Y_k, k = 1, \dots, m)$ menerima suatu pola target yang sesuai dengan pola input pelatihan,

untuk menghitung kesalahan (error) antara target dengan output yang dihasilkan jaringan

$$\delta = (t_k - y_k) f'(y_{in_k}) \quad \text{.....(9)}$$

Keterangan:

δ : Faktor koreksi error

t : Data training untuk target output t-k

f' : fungsi aktivasi

y_{ink} : input unit output ke-k

Faktor δ_k digunakan untuk menghitung koreksi error (Δw_{jk}) yang nantinya akan dipakai untuk memperbaiki w_{jk}, dimana

$$\Delta w_{jk} = \alpha \delta_k Z_j \quad \text{.....(10)}$$

Keterangan:

Δw_{jk} : Koreksi bobot antara hidden unit ke-j dengan unit output ke-k

α : Learning rate

δ_k : Faktor koreksi error untuk bobot w_{jk}

Z_j : Hidden unit ke-j

Selain itu juga dihitung koreksi bias (Δw_{0k}) yang nantinya akan dipakai untuk memperbaiki w_{0k}, dimana

$$\Delta w_{0k} = \alpha \delta_k \quad \text{.....(11)}$$

Keterangan:

Δw_{0k} : koreksi bias dengan unit output ke-k

α : Learning rate

δ_k : Faktor koreksi error untuk bobot w_{jk}

Faktor δ_k kemudian dikirimkan ke lapisan yang berada pada langkah ke-7.

Langkah ke-7 : Setiap *hidden* unit $(Z_j, j = 1, \dots, p)$ menerima input delta (dari langkah ke-6) yang sudah berbobot

$$\boxed{\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk}} \quad \dots\dots\dots\dots\dots\dots\dots(12)$$

Keterangan:

δ_{in_j} : input faktor koreksi error untuk bobot v_{ij}

δ_k : Faktor koreksi error untuk bobot w_{jk}

w_{jk} : Bobot antara hidden unit ke-j dengan unit output ke-k

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghitung informasi kesalahan error δ_j, dengan

$$\boxed{\delta_j = \delta\_in_j f'(z_{in_j})} \quad \dots\dots\dots\dots\dots\dots\dots(13)$$

Keterangan:

δ_j : Faktor koreksi error untuk bobot v_{ij}

δ_in_j : Input faktor koreksi error untuk bobot v_{ij}

f' : Turunan fungsi aktivasi

z_{in_j} : Hidden unit ke-j

Kemudian hitunglah koreksi bobot (untuk memperbaiki v_{ij})

$$\boxed{\Delta v_{ij} = \alpha \delta_j x_i} \quad \dots\dots\dots\dots\dots\dots\dots(14)$$

Keterangan:

Δv_{ij} : Koreksi bobot antara hidden unit ke-j dengan unit output ke-j

α : Learning rate

δ_j : Faktor koreksi error untuk bobot v_{ij}

x : Data training input x

Setelah itu hitung koreksi bias (digunakan untuk memperbaiki v_{0j}

$$\boxed{\Delta v_{0j} = \alpha \delta_j} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(15)$$

Keterangan:

Δv_{0j} : Koreksi Bias untuk hidden unit ke-j

α : Learning rate

δ_j : Faktor koreksi error untuk bobot v_{ij}

Memperbarui Bobot dan Bias (Adjustment)

Langkah ke-8 : Setiap unit output ($Y_k, k = 1, \ldots, m$) memperbaiki bobot dan biasnya

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

Keterangan:

w_{jk} : Bobot antara hidden unit ke-j dengan unit output ke-k

Δw_{jk} : Koreksi bobot antara hidden unit ke-j dengan unit output ke-k

Demikian pula untuk setiap hidden unit ($Z_j, j = 1, \ldots, p$) akan memperbaharui bobot dan biasnya

$$\boxed{v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(17)$$

Keterangan:

v_{ij} : Bobot antara hidden unit ke-j dengan unit output ke-k

Δv_{ij} : koreksi bobot antara hidden unit ke-j dengan unit output ke-k

Langkah ke-9 : Tes kondisi berhenti apabila error ditemukan, jika kondisi STOP telah terpenuhi, maka pelatihan jaringan dapat dihentikan. Untuk memeriksa kondisi STOP, biasanya digunakan kriteria MSE (Mean Square Error) berikut ini :

$$MSE = 0.5 \times \{(t_{k1} - y_{k1})^2 + (t_{k2} - y_{k2})^2 + (t_{km} - y_{km})^2\} \quad \text{....(18)}$$

Keterangan:

MSE : Mean Square Error

m : Momentum

T_{dk} : Data training

Y_{dk} : Unit output

2.3. *Particle Swarm Optimization* (PSO)

Particle Swarm Optimization (PSO) adalah teknik optimasi berbasis populasi yang dikembangkan oleh Eberhart dan Kennedy pada tahun 1995, yang terinspirasi oleh perilaku sosial kawanan burung atau ikan (Park, Lee, & Choi, 2009). *Particle swarm optimization* dapat diasumsikan sebagai kelompok burung yang sedang mencari makanan disuatu daerah. Burung tersebut tidak tahu dimana makanan tersebut berada, tapi mereka tahu sebarapa jauh makanan itu berada, jadi strategi terbaik untuk menemukan makanan tersebut adalah dengan mengikuti burung yang terdekat dari makanan tersebut (Salappa, Doumpos, & Zopounidis, 2007). *Particle swarm optimization* digunakan untuk memecahkan masalah optimasi dengan mengoptimasi nilai minimum *error* pada jaringan sehingga didapat bobot jaringan jaringan syaraf tiruan yang ideal.

Untuk menemukan solusi yang optimal, masing-masing partikel bergerak ke arah posisi yang terbaik sebelumnya dan posisi terbaik secara global.

Sebagai contoh, partikel ke-i dinyatakan sebagai: $xi = (xi1, xi2, \ldots . xid)$ dalam ruang dimensi. Posisi terbaik sebelumnya dari partikel ke-i disimpan dan dinyatakan sebagai $Pbesti = (Pbesti, 1, Pbesti, 2, \ldots Pbesti, d)$. Indeks partikel terbaik diantara semua partikel dalam kawanan grup dinyatakan sebagai $Gbestd$. Kecepatan partikel dinyatakan sebagai: $vi = (vi, 1, vi, 2, \ldots . vi, d)$. Modifikasi kecepatan dan posisi partikel dapat dihitung menggunakan kecepatan saat ini dan jarak $Pbesti, Gbestd$.

Adapun langkah-langkah pada *Particle Swarm Optimization* (PSO) dapat dijelaskan sebagai berikut (Darmayanti et al., 2018):

1. Proses Inisialisasi
   a. Inisialisasi Kecepatan dan Posisi Awal Partikel
      Pada iterasi awal, nilai kecepatan awal semua partikel diinisialisasi dengan nilai 0, serta posisi awal partikel dibangkitkan secara acak.
   b. Menghitung Nilai Fitness
   c. Inisialisasi Nilai Pbest dan Gbest
2. Memperbarui kecepatan partikel
   Untuk memperbarui kecepatan pada tiap partikelnya digunakan rumus dengan sebagai berikut.

$$V_j(i) = V_j(i-1) + c_1 r_1 \left[P_{best,j} - x_j(i+1)\right] + c_2 r_2 [G_{best,j} - x_j(i-1)] \quad \ldots\ldots\ldots.(19)$$

Keterangan :

$V_j(i)$: Kecepatan sebelumnya

$c_1 c_2$: Konstanta akselerasi (*learning rate*)

$r_1 r_2$: nilai acak antara 0-1

$P_{best,j}$: nilai personal terbaik

$G_{best,j}$: nilai global terbaik

x_j : posisi sekarang

Dengan persamaan tersebut, menghitung kecepatan baru untuk tiap partikel (solusi potensial) berdasarkan pada kecepatan sekarang (V_j), lokasi partikel dimana nilai *fitness* terbaik telah dicapai ($Pbest$), dan lokasi populasi global ($Gbest$ untuk versi global) dimana nilai *fitness* terbaik telah dicapai.

Dengan memperbarui kecepatan pada tiap partikelnya ini dapat membuat partikel mampu bergerak mendekati solusi yang optimal.

3. Memperbarui posisi partikel

   Untuk memperbarui posisi partikel digunakan rumus sebagai berikut.

$$\boxed{x_j(i) = x_j(i-1) + V_j(i)} \quad \text{.....(20)}$$

   Keterangan :

   $V_j(i)$: Kecepatan posisi sekarang

   x_j : posisi sekarang

   Memperbarui posisi dari partikel menggunakan kecepatan yang telah diperbarui. Setelah memperbarui posisi, kemudian dihitung kembali nilai fitness tiap partikel yang baru sehingga dapat dilakukan pembaruan untuk *Pbest* dan *Gbest.*

4. Memperbarui Nilai *Pbest* dan *Gbest*

   Untuk mendapat nilai *Pbest* dapat dilakukan dengan cara membandingkan antara nilai *Pbest* pada iterasi sebelumnya dengan hasil dari perhitungan update posisi. Nilai fitness yang lebih rendah dari keduanya akan menjadi nilai *Pbest* yang baru.

   Sedangkan untuk mendapatkan nilai *Gbest* terbaru, dapat dilakukan perhitungan dengan cara melihat nilai *Pbest* yang memiliki nilai fitness yang paling rendah.

5. Mengulangi langkah ke-2 sampai kondisi berhenti terpenuhi. Ada beberapa kondisi berhenti yang digunakan, diantaranya adalah:

a. Ketika iterasi sudah mencapai Maximum
b. Iterasi berhenti tetapi tidak ada perubahan yang signifikan
c. Ketika telah mencapai waktu Maximum

2.4. Prediksi

Prediksi adalah usaha menduga atau memperkirakan sesuatu yang akan terjadi di waktu mendatang dengan memanfaatkan berbagai informasi yang relevan pada waktu-waktu sebelumnya (historis) melalui suatu metode ilmiah (Wanto, 2018). Suatu prediksi dianggap baik apabila mendekati kebenaran.

Pengertian prediksi menurut Heizer dan Render (2015) adalah suatu seni dan ilmu pengetahuan memprediksi peristiwa pada masa yang akan datang. Prediksi akan melibatkan mengambil data historis dan memproyeksikan data tersebut ke masa yang akan datang dengan menggunakan model matematika. Prediksi pada umumnya digunakan untuk memprediksi apa yang akan terjadi berdasarkan sekelompok asumsi dan situasi tertentu. Perencanaan melibatkan penggunaan ramalan-ramalan untuk membantu menghasilkan keputusan yang baik tentang alternatif dan secara umum prediksi adalah masukan untuk proses perencanaan (Makridakis & Wheelwright, 1994).

2.5. Pengujian Kinerja Sistem

Dalam mengukur kinerja dari sebuah sistem prediksi jumlah penumpang kereta api digunakan pengujian dengan melakukan perhitungan kesalahan dengan menggunakan MAPE (*Mean Absolute Percentage Error*) dan MSE (*Mean Square Error*). Kedua perhitungan ini biasa digunakan untuk menghitung kesalahan yang dihasilkan. Rumus untuk perhitungan kesalahan dengan MAPE dan MSE adalah sebagai berikut.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|Y_t - \hat{Y}_t|}{Y_t}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(Y_t - \hat{Y}_t)^2$$

Keterangan:

n = jumlah data

y = nilai hasil aktual

$\hat{y}$ = nilai hasil prediksi

y_t = nilai hasil aktual

Keterangan: $\hat{y}_t$ = nilai hasil prediksi

n = jumlah data

2.6. Studi Kasus

Pada subbab ini dijelaskan mengenai studi kasus dari metode *backpropagation* dan *particle swarm optimization.*

2.6.1. Studi Kasus Algoritma *Particle Swarm Optimization*

Berikut ini adalah contoh studi kasus perhitungan dengan menggunakan algoritma *particle swarm optimization* untuk melakukan pencarian solusi yang paling optimal dari suatu persoalan.

Diberikan persoalan optimasi dengan satu variabel sebagai berikut min $f(x) = (100 - x)^2$, dimana $60 \leq x \leq 120$

1. Tentukan jumlah partikel yaitu $N = 4$, ukuran jumlah partikel ini sebaiknya N tidak terlalu besar, tetapi juga tidak terlalu kecil. Supaya terdapat beberapa kemungkinan posisi untuk menuju solusi yang terbaik atau optimal.

   Selanjutnya, tentukan populasi awal secara random, misalkan didapat populasi awal adalah dengan sebagai berikut.

   $x_1(0) = 80,$

   $x_2(0) = 90,$

   $x_3(0) = 110,$

   $x_4(0) = 75.$

2. Evaluasi nilai fungsi tujuan untuk setiap partikel $x_j(0)$ untuk j = 1,2,3,4. Menggunakan

$$\boxed{f(x) = (100 - x)^2}$$

dan nyatakan dengan

$f_1 = f(80) = 400,$

$f_2 = f(90) = 100,$

$f_3 = f(110) = 100,$

$f_4 = f(75) = 625,$

3. Tentukan kecepatan awal dengan $v_1(0) = v_2(0) = v_3(0) = v_4(0) = 0$. Awalnya semua kecepatan dari partikel diasumsikan sama dengan nol. Tetapkan iterasi $i = 1$; Lalu ke langkah selanjutnya pada nomor 4.
4. Nilai terbaik yang ditemukan pada sebuah partikel dari semua iterasi dinyatakan sebagai P_{best}. Sedangkan dari semua partikel dinyatakan dengan G_{best}.

   $P_{best1} = 80,$

   $P_{best2} = 90,$

   $P_{best3} = 110,$

   $P_{best4} = 75,$

   $G_{best} = 90.$

   Selanjutnya hitung $v(j)$ dengan $C_1 = C_2 = 1$, dengan nilai C_1 dan C_2 adalah konstanta untuk kemampuan individu partikel. Misalkan nilai random yang didapat, $r_1 = 0.4, r_2 = 0.5$, dengan rumus

$$V_j(i) = V_j(i-1) + c_1 r_1 \left[P_{best,j} - x_j(i+1)\right] + c_2 r_2 [G_{best,j} - x_j(i-1)]$$

   Didapatkan nilai:

   $r_1 = 0.4, r_2 = 0.5,$

   $c_1 = c_2 = 1$, maka

   $v_1(1) = 0 + 0.4(80 - 80) + 0.5(90 - 80) = 5$

   $v_2(1) = 0 + 0.4(90 - 90) + 0.5(90 - 90) = 0$

   $v_3(1) = 0 + 0.4(110 - 110) + 0.5(90 - 110) = -10$

   $v_4(1) = 0 + 0.4(75 - 75) + 0.5(90 - 75) = 7.5$

Hitung posisi atau koordinat partikel j pada iterasi ke-i dengan rumus sebagai berikut

$$x_j(i) = x_j(i-1) + V_j(i)$$

Dengan nilai:

$v_1 = 5$

$v_2 = 0$

$v_3 = -10$

$v_4 = 7.5$, maka didapat perhitungan dengan sebagai berikut

$x_1(1)$ = 80+5 = 85

$x_2(1)$ = 90+0 = 90

$x_3(1)$ = 110−10 = 100

$x_4(1)$ = 75+7.5 = 82.5

5. Evaluasi nilai fungsi tujuan sekarang pada partikel $x_j(1)$. Ini dilakukan untuk mengukur solusi yang didapat merupakan hasil yang sudah optimal atau belum, dengan menggunakan

$$f(x) = (100 - x)^2$$

$f_1(1) = f(85) = 225,$

$f_2(1) = f(90) = 100,$

$f_3(1) = f(100) = 0,$

$f_4(1) = f(82.5) = 306.25.$

Sedangkan pada iterasi sebelumnya kita dapatkan

$f_1(0) = f(80) = 400,$

$f_2(0) = f(90) = 100,$

$f_3(0) = f(110) = 100,$

$f_4(0) = f(75) = 625.$

Nilai f dari iterasi sebelumnya tidak ada yang lebih baik sehingga P_{best} untuk masing-masing partikel sama dengan nilai x-nya. Sedangkan $G_{best=}100$.

Cek apakah solusi x sudah konvergen, yaitu nilai x saling dekat. Jika tidak, tingkatkan ke iterasi berikutnya $i = 2$.

Lanjutkan ke langkah 4.

1. $P_{best1} = 85$, $P_{best4} = 82.5$,
   $P_{best2} = 90$, $G_{best} = 100$
   $P_{best3} = 100$,
   Hitung kecepatan baru dengan r_1 = 0.3 dan r_2 = 0.6 (ini hanya sekedar contoh untuk menjelaskan penghitungan, dalam implementasi angka ini dibangkitkan secara random).

$$V_j(i) = V_j(i-1) + c_1 r_1 \left[P_{best,j} - x_j(i+1)\right] + c_2 r_2 [G_{best,j} - x_j(i-1)]$$

Dengan nilai yang digunakan:

$r_1 = 0.3$ $v_1 = 5$ $v_4 = 7.5$

$r_2 = 0.6$ $v_2 = 0$

$c_1 = c_1 = 1$ $v_3 = -10$

$v_1(2) = 5 + 0.3(85 - 85) + 0.6(100 - 85) = 14$

$v_2(2) = 0 + 0.3(90 - 90) + 0.6(100 - 90) = 6.$

$v_3(2) = -10 + 0.3(100 - 100) + 0.6(100 - 100) = -10$

$v_4(2) = 7.5 + 0.3(82.5 - 82.5) + 0.6(100 - 82.5) = 18$

Sedangkan untuk nilai x pada posisi atau koordinat partikel j pada iterasi ke-i digunakan dengan rumus sebagai berikut

$$x_j(i) = x_j(i-1) + V_j(i)$$

$x_1(2) = 85 + 14 = 99$

$x_2(2) = 90 + 6 = 96$

$x_3(2) = 100 - 10 = 90$

$x_4(2) = 82.5 + 18 = 100.5$

2. Evaluasi nilai fungsi tujuan sekarang pada partikel $x_j(2)$ atau pada iterasi kedua, dengan menggunakan rumus

$$f(x) = (100 - x)^2$$

$f_1(2) = f(99) = 1,$

$f_2(2) = f(96) = 16,$

$f_3(2) = f(90) = 100$

$f_4(2) = f(100.5) = 0.25.$

Jika dibandingkan dengan nilai f dari iterasi sebelumnya, ada nilai yang lebih baik dari nilai f sekarang yaitu f_3 (1) = 0, sehingga P_{best} untuk partikel 3 sama dengan 100, dan G_{best} dicari dari min (1,16,0,0.25) = 0 yang dicapai pada x_3 (1) = 100. Sehingga untuk iterasi berikutnya P_{best} = (99, 96, 100, 100.5) dan G_{best} = 100.

Cek apakah solusi sudah konvergen, dimana nilai x saling dekat. Jika belum konvergen, set i = 3, masuk ke iterasi berikutnya. Lanjutkan ke langkah berikutnya dengan menghitung kecepatan v dan ulangi langkah-langkah selanjunya sampai mencapai konvergen.

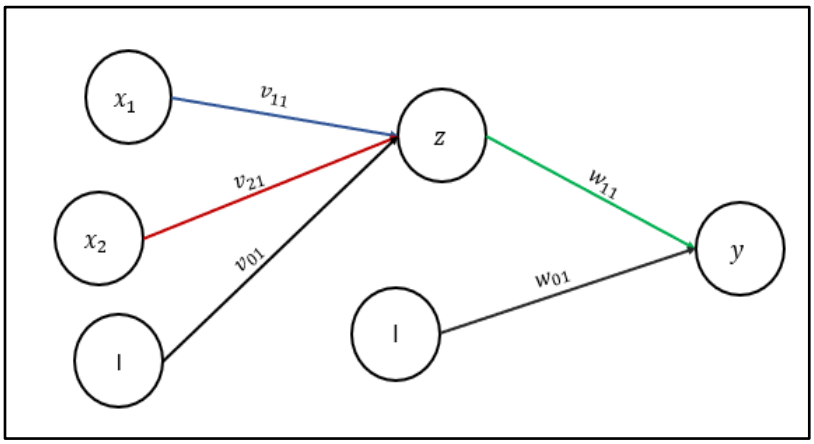
2.6.2. Studi Kasus Metode *Backpropagation*

Berikut ini adalah contoh studi kasus perhitungan dengan menggunakan metode *backpropagation* untuk menemukan bobot yang baru dan kemudian akan dihasilkan ouput jaringan.

Sebuah jaringan terdiri atas dua unit masukkan (*input unit)*, satu unit tersembunyi (*hidden unit)*, dan satu unit keluaran (*output unit)*. Fungsi aktivasi yang digunakan adalah Sigmoid Biner, dengan persamaan

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

Learning rate / alpha (α) = 0.01, toleransi error yang diperkenankan adalah 0.41. Jaringan digunakan untuk menyelesaikan fungsi XOR. Berikut akan ditampilkan arsitektur jaringan yang digunakan pada studi kasus ini.



Gambar 2.5 Studi Kasus Arsitektur Jaringan

Adapun data pelatihan yang digunakan pada studi kasus berikut ini, terdiri atas empat pasang masukkan dan keluaran yakni:

Tabel 2.1 Contoh Data Pelatihan

No.	Masukan 1	Masukan 2	Keluaran
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Langkah-langkah pada proses pelatihan adalah sebagai berikut :

Langkah 0 : Inisialisasi sembarang bobot dan bias, misalnya

Bias:

$v_{01} = 1{,}718946$

$w_{01} = -0{,}541180$

Bobot:

$v_{11} = -1{,}263178$

$v_{21} = -1{,}083092$

$w_{11} = 0{,}543960$

Langkah 1 : Dengan bobot sembarang tersebut, tentukan error untuk Dataset pelatihan secara keseluruhan. Dengan meng-gunakan rumus (5) dan (6)

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

$$z_j = f(z_{in_j})$$

$v_{0j} = 1{,}718946$ $\quad x_1 = 0 \quad x_2 = 0$

$v_{11} = -1{,}263178$ $\quad x_3 = 0 \quad x_4 = 1$

$v_{21} = -1{,}083092$ $\quad x_5 = 1 \quad x_6 = 0$

$x_7 = 1 \quad x_8 = 1$

$z\_in_{11} = 1{,}718946 + \{(0 \times -1{,}263178) + (0 \times -1{,}083092)\} = 1{,}718946$

$z_{11} = f(z_{in_{11}}) = 0{,}847993$

$z\_in_{12} = 1{,}718946 + \{(0 \times -1{,}263178) + (1 \times -1{,}083092)\} = 0{,}635854$

$z_{12} = f(z_{in_{12}}) = 0{,}653816$

$z\_in_{13} = 1{,}718946 + \{(1 \times -1{,}263178) + (0 \times -1{,}083092)\} = 0{,}455768$

$z_{13} = f(z_{in_{13}}) = 0{,}612009$

$z\_in_{14} = 1{,}718946 + \{(1 \times -1{,}263178) + (1 \times -1{,}083092)\} = -0{,}627324$

$z_{14} = f(z_{in_{14}}) = 0{,}348118$

Dari perhitungan tersebut didapatlah nilai setiap *hidden unit* ($Z_j, j = 1, \dots, p$) dari penjumlahan sinyal-sinyal input yang sudah berbobot (termasuk biasnya)

Dimana indeks z_{jn} berarti bobot untuk hidden unit ke-j dan data training ke-n, dengan rumus (7) dan (8)

$$y\_in_k = w_{0k} + \sum_{i=1}^{n} z_i w_{jk}$$

$$y_k = f(y_{in_k})$$

Diketahui:

$w_{01} = -0{,}541180$ $\quad z_{11} = 0{,}847993$

$w_{11} = 0{,}543960$ $\quad z_{12} = 0{,}653816$

$z_{13} = 0{,}612009$

$z_{14} = 0{,}348118$

$$y_{in_{11}} = -0{,}541180 + (0{,}847993 \times 0{,}543960)$$
$$= -0{,}079906$$
$$y_{11} = f(y\_in_{11}) = 0{,}480034$$
$$y_{in_{12}} = -0{,}541180 + (0{,}653816 \times 0{,}543960)$$
$$= -0{,}185530$$
$$y_{12} = f(y\_in_{12}) = 0{,}453750$$
$$y_{in_{13}} = -0{,}541180 + (0{,}612009 \times 0{,}543960)$$
$$= -0{,}208271$$
$$y_{13} = f(y\_in_{13}) = 0{,}448119$$
$$y_{in_{14}} = -0{,}541180 + (0{,}348118 \times 0{,}543960)$$
$$= -0{,}351818$$
$$y_{14} = f(y\_in_{14}) = 0{,}412941$$

Pada perhitungan, tersebut didapatlah nilai tiap-tiap *unit output* (Yk, k = 1 ,…, m) yang menjumlahkan bobot sinyal input didapat:

Sehingga,

$$E = 0{,}5 \times \{(0 - 0{,}480034)^2 + (1 - 0{,}453750)^2 + (1 - 0{,}448119)^2 + (0 - 0{,}412941)^2\} = 0{,}501957$$

Setelah melakukan perhitungan kesalahan (*error*), sehingga didapat nilai kesalahan sebesar 0,501957. Nilai ini masih lebih besar jika dibandingkan dengan toleransi kesalahan, maka dari itu harus dilakukan pelatihan selanjutnya.

Langkah 2 : Karena, data *error training* masih lebih besar dari toleransi yakni 0.41. Maka, pelatihan dilanjutkan pada langkah 3-8.

Langkah 3 : $x_1 = 0, x_2 = 0$; (Pelatihan untuk data pertama)

Langkah 4 : Setiap *hidden unit* akan dihitung perjumlahan sinyal-sinyal input yang masuk ke *hidden layer*

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

$$z_j = f(z_{in_j})$$

Diketahui:

$v_{0j} = 1{,}718946$ $\quad$ $v_{21} = -1{,}083092$

$v_{11} = -1{,}263178$ $\quad$ $x_1 = 0$ $\quad$ $x_2 = 0$

$$z\_in_{11} = 1{,}718946 + \{(0 \times -1{,}263178) + (0 \times -1{,}083092)\} = 1{,}718946$$

$$z_{11} = f(z\_in_{11}) = 0{,}847993$$

Pada langkah ke-4 menghasilkan nilai dan sinyal output yang berasal *hidden unit*, yang nantinya akan digunakan pada tahap selanjutnya.

Langkah 5 : Sama seperti pada langkah 4, pada tahap ini sinyal-sinyal input akan dihitung perjumlahannya.

$$\boxed{y_{-in_k} = w_{0k} + \sum_{i=1}^{n} z_i w_{jk}} \qquad \boxed{y_k = f(y_{in_k})}$$

Diketahui:

$w_{01} = -0{,}541180$ $\quad$ $z_{11} = 0{,}847993$

$w_{11} = 0{,}543960$

$$y\_in_{11} = -0{,}541180 + (0{,}847993 \times 0{,}543960)$$
$$= -0{,}079906$$

$$y_{11} = f(y\_in_{11}) = 0{,}480034$$

Didapatlah nilai dan sinyal *output* yang berasal dari *unit output*. Sinyal output ini selanjutnya dikirim ke seluruh unit pada output

Langkah 6 : Masuk ke tahap propagasi balik, pada tahap ini setiap unit *output* menerima suatu *target* (*output* yang diharapkan) yang akan dibandingkan dengan *output* yang dihasilkan.

$$\boxed{\delta = (t_k - y_k) f'(y_{in_k})} \qquad \boxed{\Delta w_{jk} = \alpha \delta_k Z_j}$$

Diketahui:

$t_k = 0$ $\quad$ $\alpha = 0.01$

$y\_in_{11} = 0{,}079906$ $\quad$ $Z_j = 0{,}847993$

$$\delta_1 = (0 - 0{,}480034)f'(0{,}480034) = -0{,}119817$$

Selanjutnya akan dihitung koreksi error, untuk memperbarui w_{jk}

$$\Delta w_{11} = 0{,}01 \times -0{,}119817 \times 0{,}847993$$
$$= -0{,}001016$$

Dan dihitung pula koreksi bias, yang digunakan untuk memperbarui w_{0k}

$$\boxed{\Delta w_{0k} = \alpha\delta_k}$$

$$\Delta w_{01} = 0{,}01 \times -0{,}119817 = -0{,}00119817$$

Langkah 7 : Setiap *hidden unit* menghitung perjumlahan *input delta* (yang dikirim dari *layer* pada langkah 6) yang sudah berbobot

$$\boxed{\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk}} \quad \boxed{\delta_j = \delta\_in_j f'(z_{in_j})}$$

Diketahui:

$\delta_1 = -0{,}119817$

$w_{11} = 0{,}543960$

$Z_{inj} = 1{,}718946$

$$\delta\_in_1 = -0{,}119817 \times 0{,}543960 = -0{,}065176$$

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi *error* δ_j

$$\delta_1 = -0{,}065176 \times f'(0{,}847993) = -0{,}008401$$

Faktor δ_j ini digunakan untuk menghitung koreksi *error* (Δv_{ij}) yang nantinya akan dipakai untuk memperbaharui v_{ij},

$$\Delta v_{11} = 0{,}01 \times -0{,}008401 \times 0 = 0$$

$$\Delta v_{21} = 0{,}01 \times -0{,}008401 \times 0 = 0$$

Selain itu juga dihitung koreksi bias Δv_{0j} yang nantinya akan dipakai untuk memperbaharui v_{0j}

$$\Delta v_{01} = 0{,}01 \times -0{,}008401 = -0{,}00008401$$

Langkah 8 : Pada langkah ini masuk ke proses pembaruan bobot dan bias. Pertama, pada *hidden layer* terlebih dahulu

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

Diketahui:

$w_{0k} = -0{,}541180$ $\quad w_{jk} = 0{,}543960$

$\Delta w_{0k} = -0{,}00119817$ $\quad \Delta w_{jk} = -0{,}001016$

$$w_{o1}(baru) = -\,0{,}541180 + (-0{,}00119817)$$
$$= -0{,}542378$$

$$w_{11}(baru) = 0{,}543960 + (-0{,}001016\ )$$
$$= 0{,}542944$$

Dan yang kedua, pada *output layer* akan dihitung pula pembaruan bobot dan biasnya

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

$$v_{o1}(baru) = 1{,}718946 + (-0{,}00008401)$$
$$= 1{,}718862$$

$$v_{11}(baru) = 1{,}263178 + 0 = 1{,}263178$$

$$v_{21}(baru) = 1{,}083092 + 0 = 1{,}083092$$

Setelah langkah 3-8 untuk data training pertama dikerjakan, ulangi kembali langkah 3-8 untuk data training ke-2,3 dan 4. Setelah seluruh data training dikerjakan itu berarti satu iterasi telah diproses. Bobot yang dihasilkan pada iterasi pertama untuk data training ke-2,3, dan 4 adalah :

Data Training ke-2

$v_{01} = -0{,}541023$ $\quad w_{01} = -1{,}263178$

$v_{11} = 0{,}543830$ $\quad w_{11} = -1{,}083092$

$v_{21} = 1{,}718862$

Data Training ke-3

$v_{01} = -0{,}539659$ $\quad w_{01} = -1{,}263002$

$v_{11} = 0{,}544665$ $\quad w_{11} = -1{,}082925$

$v_{21} = 1{,}719205$

Data Training ke-4

$v_{01} = -0{,}540661$ $\quad$ $w_{01} = -1{,}263126$

$v_{11} = 0{,}544316$ $\quad$ $w_{11} = -1{,}083049$

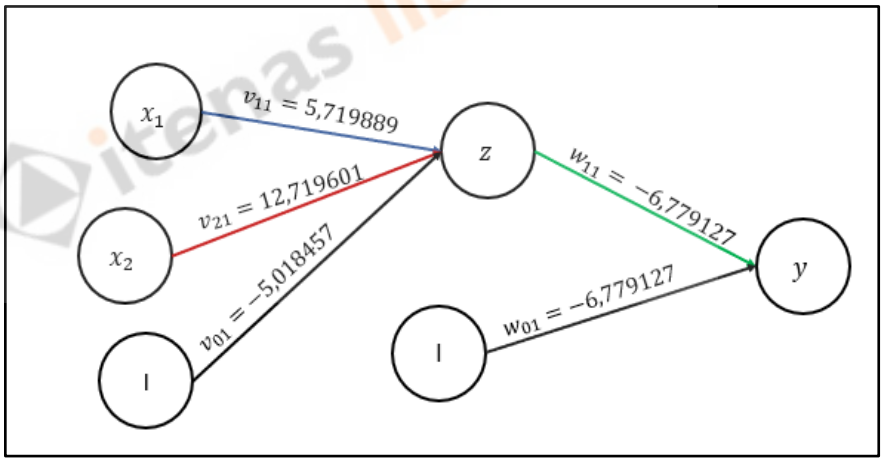
$v_{21} = 1{,}719081$

Setelah sampai pada data training ke-4, maka iterasi pertama selesai dikerjakan. Proses training dilanjutkan pada langkah ke-9 yaitu memeriksa kondisi STOP dan kembali pada langkah ke-2. Demikian seterusnya sampai error yang dihasilkan memenuhi toleransi error yang ditentukan. Setelah proses training selesai, bobot akhir yang diperoleh untuk contoh XOR adalah sebagai berikut

$v_{01} = -5{,}018457$ $\quad$ $w_{01} = -6{,}779127$

$v_{11} = 5{,}719889$ $\quad$ $w_{11} = -6{,}779127$

$v_{21} = 12{,}719601$



Gambar 2.6 Hasil Arsitektur Jaringan yang Dilatih

Setelah memperoleh bobot akhir dari proses training tersebut, didapatlah hasil arsitektur jaringan yang dapat dilihat pada gambar 10. Pada *input layer* nilai $v_{11} = 5{,}719889$ dan $v_{21} = 12{,}719601$ dengan bias $v_{01} = -5{,}018457$. Hidden layer dengan nilai $w_{11} = -6{,}779127$ dengan bias $w_{01} = -6{,}779127$.

Jika terdapat masukkan baru, misalnya $x_1 = 0{,}2$ dan $x_2 = 0{,}9$ maka keluarannya dapat dicari dengan menggunakan langkah-langkah propagasi maju berikut ini:

Langkah 0 : Bobot yang digunakan adalah bobot akhir hasil pelatihan di atas.

Langkah 1 : Perhitungan dilakukan pada langkah 2 – 4.

Langkah 2 : Dalam contoh ini, bilangan yang digunakan telah berada dalam interval 0 dan 1, jadi tidak perlu diskalakan lagi.

Langkah 3 : $z\_in_{11} = 12{,}719601 + \{(0{,}2 \times -6{,}779127) + (0{,}9 \times -6{,}779127)\} = 5{,}262561$

$$z_{11} = f(z\_in_{11}) = 0{,}994845$$

Langkah 4 : $y\_in_{11} = -5{,}018457 + (5{,}719889 \times 0{,}994845) = -0{,}671944$

$$y_{11} = f(y\_in_{11}) = 0{,}661938$$

Jadi, jika input data adalah $x_1 = 0{,}2$ dan $x_2 = 0{,}9$ output jaringan yang dihasilkan adalah 0,661938