Benyamin Kusumoputro Ph.D
Computational Intelligence, Faculty of Computer Science
University of Indonesia

METODE PEMBELAJARAN

Sebelum suatu Jaringan Neural Buatan (JNB) digunakan untuk mengklasifikasikan pola, terlebih dahulu dilakukan proses pembelajaran untuk menentukan struktur jaringan, terutama dalam penentuan nilai bobot. Dalam bagian ini akan dijelaskan beberapa metode pembelajaran yaitu metode Hebb, Perceptron, Adaline/Madaline, Backpropagation,

1. HEBB

Jaringan Hebb adalah jaringan neural buatan yang mempunyai aturan pembelajaran yang sederhana. Hebb mengusulkan bahwa pembelajaran dilakukan dengan memodifikasi bobot dimana jika 2 neuron yang terhubung adalah *"on"* dalam waktu yang sama, maka bobot diantara keduanya harus ditingkatkan. Metode ini kemudian dikembangkan dengan menambah satu prinsip lainnya yaitu bobot juga akan ditingkatkan bila kedua neuron *"off"* dalam waktu yang sama.

Karakteristik

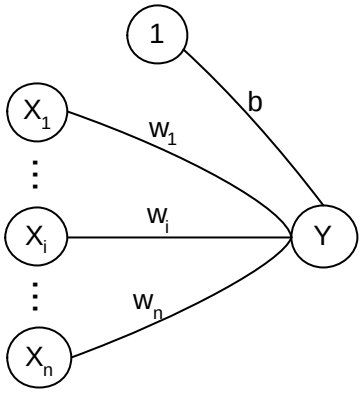
Jaringan Hebb mempunyai karakteristik sebagai berikut :

- Jaringan lapis tunggal
  - Jaringan terdiri dari satu atau lebih unit masukan dan satu unit keluaran.
  - Mempunyai sebuah bias yang berperilaku seperti bobot yang bisa disesuaikan yang terletak pada koneksi dari sebuah unit yang selalu mengeluarkan sinyal +1 agar bobot bias bisa dilatih seperti bobot lainnya dengan proses yang sama dalam algoritma pelatihan.
- Fungsi aktivasi

  Fungsi yang digunakan adalah fungsi identitas, artinya keluaran layer input sama dengan masukannya.

  $$F(y\_in) = y\_in \qquad (1)$$
- Arsitektur

  Arsitektur jaringan Hebb ditunjukkan pada gambar 1. Pada jaringan ini, terdapat n unit masukan, yaitu X_1, …, X_i, …, X_n dengan bobot w_1, …, w_i, …, w_n, dan sebuah unit keluaran, yaitu Y. Selain itu, terdapat sebuah unit yang selalu memberikan sinyal +1 dimana bobotnya diberi nama bias (b).



Gambar 1. Arsitektur Hebb

- Algoritma Pembelajaran

Untuk melatih jaringan, dilakukan langkah-langkah berikut ini :

Langkah 0. Inisialisasi seluruh bobot

$w_i = 0 \qquad i = 1, 2, \ldots, n$

Langkah 1. Untuk setiap pasangan vektor masukan pelatihan dan target keluaran, s:t (s adalah vektor masukan pelatihan dan t adalah vektor target keluaran), lakukan langkah 2-4

Langkah 2. Set aktivasi untuk unit masukan :

$x_i = s_i \qquad i = 1, 2, \ldots, n$

Langkah 3. Set aktivasi untuk unit keluaran :

$y = t$

Langkah 4. Sesuaikan bobot untuk

$w_i\,(new) = w_i\,(old) + x_i\,y \qquad i = 1, 2, \ldots, n$

Sesuaikan bias :

$b(new) = b(old) + y \qquad i = 1, 2, \ldots, n$

Perlu dicatat bahwa bias disesuaikan sama seperti sebuah bobot dari sebuah unit yang keluaran sinyalnya selalu +1. Perubahan bobot dapat juga diekspresikan dalam bentuk vektor sebagai

$$w_i\,(new) = w_i\,(old) + xy \qquad (2)$$

Hal ini sering ditulis dalam bentuk perubahan bobot, ?w, sebagai

$$?w = xy \qquad (3)$$

sehingga

$$w_i\,(new) = w_i\,(old) + ?w \qquad (4)$$

- Algoritma Pengujian

  Untuk menguji suatu masukan, dilakukan langkah-langkah berikut ini :

  Langkah 0. Inisialisasi bobot

  (digunakan nilai bobot yang diperoleh dari algoritma pelatihan)

  Langkah 1. Untuk setiap vektor masukan x, lakukan langkah 2-4

  Langkah 2. Set nilai aktifasi dari unit masukan, i = 1, …., n

  $$x_i = s_i$$

  Langkah 3. Hitung total masukan ke unit keluaran

  $$y\_in = b + \sum_i x_i w_i$$

  Langkah 4. Gunakan fungsi aktifasi

  F(y_in) = y_in

  Dimana nilai F(y_in) menjadi nilai keluaran dari unit keluaran (Y).

2. PERCEPTRON

Aturan pembelajaran perceptron mempunyai kemampuan yang lebih baik daripada aturan pembelajaran Hebb dalam memecahkan permasalahan. Dengan beberapa asumsi, diantaranya adalah bobot yang ingin dicari harus ada, Perceptron dapat selalu menemukan bobot yang sesuai. Dimana bobot yang sesuai ialah bobot jaringan yang dapat menghasilkan keluaran yang benar untuk setiap pola masukan pelatihan.

Karakteristik

Jaringan Perceptron mempunyai karakteristik sebagai berikut :

- Jaringan lapis tunggal
  - Terdiri dari satu atau lebih unit masukan dan satu unit keluaran.
  - Mempunyai sebuah bias yang berperilaku seperti bobot yang bisa disesuaikan yang terletak pada koneksi dari sebuah unit yang selalu mengeluarkan sinyal +1 agar bobot bias bisa dilatih seperti bobot lainnya dengan proses yang sama dalam algoritma pelatihan.
- Fungsi aktivasi

Fungsi yang biasa digunakan adalah fungsi tangga bipolar dengan suatu nilai batas tetap (?)

$$f(y\_in) = \begin{cases} 1 & \text{jika} \quad y\_in > ? \\ 0 & \text{jika} \quad -? \le y\_in \le ? \\ -1 & \text{jika} \quad y\_in < -? \end{cases} \tag{5}$$

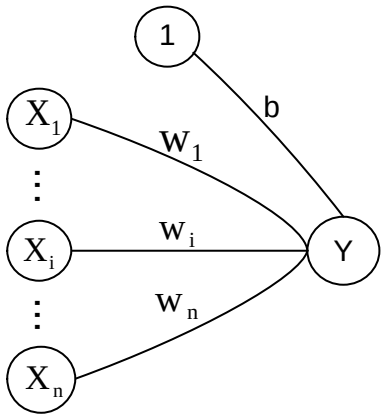
Apabila kesalahan terjadi untuk suatu pola masukan pelatihan, bobot akan diubah sesuai dengan formula

$$w_i\,(new) = w_i\,(old) + a\ t\ x_i \tag{6}$$

dimana nilai target t adalah +1 atau –1 dan a adalah laju pembelajaran. Jika kesalahan tidak terjadi, maka bobot tidak akan diubah. Pelatihan akan dilakukan terus sampai tidak ada kesalahan yang terjadi.

- Arsitektur

  Arsitektur jaringan Perceptron ditunjukkan pada gambar 1. Pada jaringan ini, terdapat n unit masukan, yaitu X_1, …, X_i, …, X_n dengan bobot w_1, …, w_i, …, w_n, dan sebuah unit keluaran, yaitu Y. Selain itu, terdapat sebuah unit yang selalu memberikan sinyal +1 dimana bobotnya diberi nama bias (b).



Gambar 2. Arsitektur Perceptron untuk melakukan klasifikasi 1 kelas

Tujuan dari jaringan dengan arsitektur seperti pada gambar 2 adalah untuk mengklasifikasi setiap pola masukan, apakah termasuk atau tidak termasuk ke dalam suatu kelas. Bila termasuk maka unit keluaran akan menghasilkan respon +1, dan bila tidak termasuk maka unit keluaran akan menghasilkan respon -1.

- Algoritma Pelatihan

  Untuk melatih jaringan, dilakukan langkah-langkah berikut ini :

  Langkah 0. Inisialisasi seluruh bobot dan bias

  (Agar sederhana, set bobot dan bias menjadi 0)

Set laju pembelajaran a ($0 < a = 1$))

(Agar sederhana, a bisa diset menjadi 1)

Langkah 1. Selama kondisi berhenti adalah salah, lakukan langkah 2-6

Langkah 2. Untuk setiap pasangan pelatihan s :t, lakukan langkah 3-5

Langkah 3. Set aktifasi dari unit masukan

$$x_i = s_i$$

Langkah 4. Hitung respon dari unit keluaran

$$y\_in = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{jika} \quad y\_in > ? \\ 0 & \text{jika} \quad -? \le y\_in \le ? \\ -1 & \text{jika} \quad y\_in < -? \end{cases}$$

Langkah 5. Sesuaikan bobot dan bias jika kesalahan terjadi untuk pola ini

Jika $y \neq t$, maka

w_i (new) = w_i (old) + a t x_i

b (new) = b (old) + a t

jika tidak, maka

w_i(new) = w_i(old)

b(new) = b(old)

Langkah 6. Tes kondisi berhenti :

Jika masih ada bobot yang berubah pada langkah 2, kembali ke langkah 1; Jika tidak, kembali ke langkah 2.

- Algoritma Pengujian

Setelah pelatihan, sebuah jaringan perceptron bisa digunakan untuk mengklasifikasi pola masukan. Langkah-langkah pengujian adalah sebagai berikut

Langkah 0. Inisialisasi bobot

(digunakan nilai bobot yang diperoleh dari algoritma pelatihan)

Langkah 1. Untuk setiap vektor masukan x, lakukan langkah 2-4

Langkah 2. Set nilai aktifasi dari unit masukan, i = 1, …., n

$x_i = s_i$

Langkah 3. Hitung total masukan ke unit keluaran

$$y\_in = b + \sum_i x_i w_i$$

Langkah 4. Gunakan fungsi aktifasi

$$f(y\_in) = \begin{cases} 1 & \text{jika} \quad y\_in > ? \\ 0 & \text{jika} \quad -? \le y\_in \le ? \\ -1 & \text{jika} \quad y\_in < -? \end{cases}$$

Dimana nilai f(y_in) menjadi nilai keluaran dari unit keluaran (Y).

3. ADALINE

Adaline (*Adaptive Linear Neuron*) dikembangkan oleh Widrow dan Hoff pada tahun 1960. Adaline dilatih dengan menggunakan aturan delta, yang juga dikenal sebagai aturan *least mean squares* (LMS) atau Widrow-Hoff.

Karakteristik

Jaringan Adaline mempunyai karakteristik sebagai berikut :

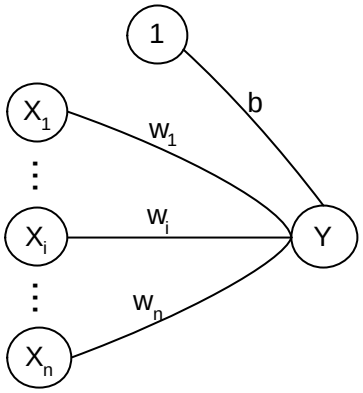
- Jaringan lapis tunggal
  - Jaringan terdiri dari satu atau lebih unit masukan dan satu unit keluaran.
  - Mempunyai sebuah bias yang berperilaku seperti bobot yang bisa disesuaikan yang terletak pada koneksi dari sebuah unit yang selalu mengeluarkan sinyal +1 agar bobot bias bisa dilatih seperti bobot lainnya dengan proses yang sama dalam algoritma pelatihan.
  - Beberapa jaringan Adaline yang menerima sinyal dari unit masukan yang sama dalam dikombinasikan menjadi sebuah jaringan lapis tunggal seperti perceptron.
  - Beberapa Adaline juga bisa dikombinasikan sehingga keluaran dari sebagian Adaline menjadi masukan untuk Adaline yang lain, dan akan membentukan jaringan lapis banyak yang disebut Madaline (*Many Adaptive Linear Neuron*).
- Fungsi aktivasi

  Fungsi yang digunakan adalah fungsi tangga

$$F(y\_in) = \begin{cases} 1 & \text{jika} \quad y\_in \ge 0 \\ -1 & \text{jika} \quad y\_in < 0 \end{cases} \qquad (7)$$

- Arsitektur

Arsitektur jaringan Adaline ditunjukkan pada gambar 3. Pada jaringan ini, terdapat n unit masukan, yaitu X_1, …, X_i, …, X_n dengan bobot w_1, …, w_i, …, w_n, dan sebuah unit keluaran, yaitu Y. Selain itu, terdapat sebuah unit yang selalu memberikan sinyal +1 dimana bobotnya diberi nama bias (b).



Gambar 3. Arsitektur Adaline

- Algoritma Pembelajaran

Langkah 0. Inisialisasi bobot

(biasanya digunakan bilangan acak yang kecil)

Set laju pembelajaran a

(0,1 = na = 1, dimana n adalah jumlah unit masukan)

Langkah 1. Selama syarat henti salah, lakukan langkah 2-6

Langkah 2. Untuk setiap pasangan pelatihan (masukan dan target) bipolar s:t, lakukan langkah 3-5

Langkah 3. Set nilai aktifasi dari unit masukan, i = 1, …., n

$$x_i = s_i$$

Langkah 4. Hitung total masukan ke unit keluaran

$$y\_in = b + \sum_i x_i w_i$$

Langkah 5. Perbarui bobot dan bias, i = 1, …., n

$$b(new) = b(old) + a\,(t - y\_in)$$

$$w_i(new) = w_i(old) + a\,(t - y\_in)\,x_i$$

Langkah 6. Uji syarat henti :

Jika perubahan bobot (a (t – y_in)) terbesar yang terjadi dalam langkah 2 adalah lebih kecil dari toleransi (e) yang telah ditentukan, maka selesai; jika tidak maka kembali ke langkah 1

Nilai toleransi (e) yang digunakan adalah 1 < e = 0.

Dalam menentukan nilai laju pembelajaran (a), umumnya digunakan nilai yang kecil (misalkan a = 0.1). Apabila nilai a terlalu besar, proses pembelajaran tidak akan konvergen. Jika terlalu kecil nilai yang dipilih, pembelajaran akan menjadi terlalu lambat. Agar praktis, kisaran nilai a yang bisa dipilih adalah 0,1 = na = 1 dimana n adalah jumlah unit masukan.

- Algoritma Pengujian

Setelah pelatihan, sebuah jaringan Adaline bisa digunakan untuk mengklasifikasi pola masukan. Bila nilai target adalah bivalen (biner atau bipolar), fungsi tangga bisa digunakan sebagai fungsi aktivasi dari unit keluaran. Prosedur umum ini adalah langkah-langkah yang digunakan apabila target adalah bipolar :

Langkah 0. Inisialisasi bobot

(digunakan nilai bobot yang diperoleh dari algoritma pelatihan)

Langkah 1. Untuk setiap vektor masukan x, lakukan langkah 2-4

Langkah 2. Set nilai aktifasi dari unit masukan, i = 1, …., n

$$x_i = s_i$$

Langkah 3. Hitung total masukan ke unit keluaran

$$y\_in = b + \sum_i x_i w_i$$

Langkah 4. Gunakan fungsi aktifasi

$$F(y\_in) = \begin{cases} 1 & \text{jika} \quad y\_in \geq 0 \\ -1 & \text{jika} \quad y\_in < 0 \end{cases}$$

Dimana nilai F(y_in) menjadi nilai keluaran dari unit keluaran (Y).

4. MADALINE

Madaline (*Many Adaptive Linear Neuron*) merupakan kombinasi dari beberapa Adaline yang diatur sedemikian rupa sehingga membentuk jaringan lapis banyak.

Karakteristik

Jaringan Madaline mempunyai karakteristik sebagai berikut :

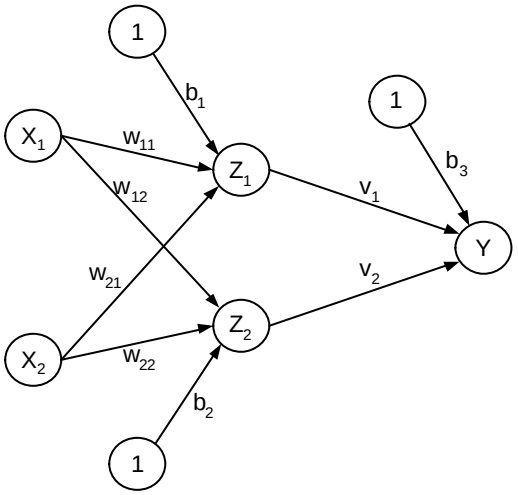
- Jaringan lapis banyak
- Fungsi aktivasi

  Fungsi aktivasi digunakan untuk menghitung sinyal keluaran dari setiap jaringan Adaline yang membentuk jaringan Madaline. Dimana Fungsi yang digunakan adalah fungsi tangga

$$F(y\_in) = \begin{cases} 1 & \text{jika} \quad y\_in \geq 0 \\ -1 & \text{jika} \quad y\_in < 0 \end{cases} \tag{8}$$

- Arsitektur

  Arsitektur Madaline tergantung kepada kombinasi Adaline yang digunakan. Pada gambar 4 ditunjukkan sebuah Madaline sederhana yang mempunyai 2 Adaline tersembunyi dan sebuah Adaline keluaran. Keluaran dari 2 Adaline tersembunyi, Z_1 dan Z_2, ditentukan oleh sinyal yang berasal dari unit masukan yang sama yaitu X_1 dan X_2. Penggunaan lapisan tersembunyi, Z_1 dan Z_2, memberikan kemampuan komputasi jaringan yang tidak ditemukan pada jaringan lapis tunggal, tetapi juga menyebabkan pelatihan menjadi lebih rumit. Persamaan (8) digunakan sebagai fungsi aktivasi oleh setiap unit keluaran Adaline (Z_1, Z_2, dan Y).



Gambar 4. Arsitektur Madaline dengan 2 Adaline tersembunyi dan sebuah Adaline keluaran

- Algoritma Pembelajaran

  Terdapat 2 macam algoritma pembelajaran untuk Madaline yang disebut MRI dan MRII. Dalam algoritma MRI, hanya bobot-bobot pada Adaline tersembunyi saja yang disesuaikan, sedangkan bobot pada unit keluaran nilainya tetap. Pada

algoritma MRII terdapat metode untuk menyesuaikan seluruh bobot dalam jaringan. Untuk menjelaskan langkah-langkah dalam MRI dan MRII, digunakan contoh arsitektur Madaline yang terdapat pada gambar 4.

Algoritma MRI

Pada MRI, bobot v_1 dan v_2 serta bias b_3, yang memberikan sinyal masukan ke unit keluaran Y, ditentukan sedemikian rupa sehingga respon dari unit Y adalah 1 jika sinyal Z_1 atau/dan Z_2 adalah 1, dan adalah -1 jika Z_1 dan Z_2 mengirimkan sinyal -1. Dengan kata lain, unit Y melakukan fungsi logika OR terhadap sinyal yang diterima dari Z_1 dan Z_2. Sehingga bobot-bobot yang menuju ke Y adalah :

$$v_1 = \frac{1}{2} \quad , \quad v_2 = \frac{1}{2} \quad , \quad b_3 = \frac{1}{2}$$

Bobot pada Adaline tersembunyi pertama (w_{11} dan w_{21}) dan bobot pada Adaline tersembunyi kedua (w_{12} dan w_{22}) disesuaikan menurut langkah-langkah berikut ini :

Fungsi aktivasi untuk unit Z_1, Z_2, dan Y adalah

$$f(x) = \begin{cases} 1 & \text{jika} \quad x \geq 0 \\ -1 & \text{jika} \quad x < 0 \end{cases}$$

Langkah 0. Inisialisasi bobot

Set bobot v_1, v_2 dan b_3 menggunakan nilai yang telah dijelaskan sebelumnya.

Set bobot lainnya (w_{11}, w_{21}, w_{12}, w_{22}, b_1, dan b_2) menggunakan bilangan acak kecil.

Set laju pembelajaran a dengan bilangan acak kecil (0,1 = na = 1, dimana n adalah jumlah unit masukan).

Langkah 1. Selama syarat henti salah, lakukan langkah 2-8

Langkah 2. Untuk setiap pasangan pelatihan (masukan dan target) bipolar s:t, lakukan langkah 3-7

Langkah 3. Set aktifasi dari unit masukan, i = 1, …., n

$$x_i = s_i$$

Langkah 4. Hitung masukan jaringan ke setiap unit Adaline tersembunyi

$z\_in_1 = b_1 + x_1 w_{11} + x_2 w_{21}$

$z\_in_2 = b_2 + x_1 w_{12} + x_2 w_{22}$

Langkah 5. Tentukan keluaran dari setiap unit Adaline tersembunyi

$z_1 = f(z\_in_1)$

$z_2 = f(z\_in_2)$

Langkah 6. Tentukan keluaran dari jaringan

$y\_in = b_3 + z_1 v_1 + z_2 v_2$

$y = f(y\_in)$

Langkah 7. Tentukan kesalahan dan penyesuaian bobot :

Jika t = y, tak ada penyesuaian bobot.

Sebaliknya,

jika t = 1, maka sesuaikan bobot pada Z_J, yaitu unit yang masukan jaringannya paling mendekati 0,

$b_J(new) = b_J(old) + a\ (1 - z\_in_J)$

$w_{iJ}(new) = b_J(old) + a\ (1 - z\_in_J)$

jika t = -1, maka sesuaikan bobot pada seluruh unit Z_k yang memiliki masukan jaringan yang positif,

$b_k(new) = b_k(old) + a\ (-1 - z\_in_k)$

$w_{ik}(new) = w_{ik}(old) + a\ (-1 - z\_in_k)\ x_i$

Langkah 8. Uji syarat henti.

Jika tidak ada perubahan bobot (atau telah mencapai level yang cukup), atau jika telah mencapai jumlah maksimum dari iterasi perubahan bobot (pada langkah 2), maka berhenti; jika tidak lanjutkan.

Algoritma MRII

Pada aturan pembelajaran ini, seluruh bobot dari setiap lapisan dalam jaringan akan dilatih. Langkah-langkah yang dilakukan adalah sebagai berikut :

Langkah 0. Inisialisasi bobot

Set laju pembelajaran a dengan bilangan acak kecil (0,1 = na = 1, dimana n adalah jumlah unit masukan).

Langkah 1. Selama syarat henti salah, lakukan langkah 2-8

Langkah 2. Untuk setiap pasangan pelatihan (masukan dan target) bipolar s:t, lakukan langkah 3-7

Langkah 3. Set aktifasi dari unit masukan, i = 1, …., n

$$x_i = s_i$$

Langkah 4. Hitung masukan jaringan ke setiap unit Adaline tersembunyi

$$z\_in_1 = b_1 + x_1 w_{11} + x_2 w_{21}$$

$$z\_in_2 = b_2 + x_1 w_{12} + x_2 w_{22}$$

Langkah 5. Tentukan keluaran dari setiap unit Adaline tersembunyi

$$z_1 = f(z\_in_1)$$

$$z_2 = f(z\_in_2)$$

Langkah 6. Tentukan keluaran dari jaringan

$$y\_in = b_3 + z_1 v_1 + z_2 v_2$$

$$y = f(y\_in)$$

Langkah 7. Tentukan kesalahan dan sesuaikan bobot jika perlu :

Jika t ? y, lakukan langkah 7a-b untuk setiap unit tersembunyi yang masukan jaringannya adalah cukup dekat dengan 0 (katakan, antara –0,25 dan 0,25). Mulai dengan unit yang masukan jaringannya adalah paling dekat dengan 0, lalu untuk yang paling dekat berikutnya, dan seterusnya.

Langkah 7a : Ubah keluaran unit (dari +1 menjadi –1, atau sebaliknya)

Langkah 7b : Hitung kembali respon dari jaringan.

Jika kesalahan berkurang :

Sesuaikan bobot pada unit ini (gunakan nilai keluaran yang baru sebagai target dan lakukan aturan Delta).

Langkah 8 : Uji syarat henti.

Jika tidak ada perubahan bobot (atau telah mencapai level yang cukup), atau jika telah mencapai jumlah maksimum dari iterasi perubahan bobot (pada langkah 2), maka berhenti; jika tidak lanjutkan.

- Algoritma Pengujian

  Setelah pelatihan, sebuah jaringan Madaline bisa digunakan untuk mengklasifikasi pola masukan. Bila nilai target adalah bivalen (biner atau bipolar), fungsi tangga bisa digunakan sebagai fungsi aktivasi dari unit keluaran. Prosedur umum ini adalah langkah-langkah yang digunakan apabila target adalah bipolar :

  Langkah 0 : Inisialisasi bobot

  (digunakan nilai bobot yang diperoleh dari algoritma pelatihan)

  Langkah 1 : Untuk setiap jaringan Adaline, lakukan langkah 2-5

  Langkah 2 : Untuk setiap vektor masukan x, lakukan langkah 2-4

  Langkah 3 : Set nilai aktifasi dari unit masukan, i = 1, …., n

  $$x_i = s_i$$

  Langkah 4 : Hitung total masukan ke unit keluaran

  $$y\_in = b + \sum_i x_i w_i$$

  Langkah 5 : Gunakan fungsi aktivasi

  Dimana nilai F(y_in) menjadi nilai keluaran dari unit keluaran jaringan.

5. BACKPROPAGATION

Keterbatasan jaringan neural lapis tunggal menyebabkan penurunan minat dalam JNB pada tahun 1970-an. Pada sekitar tahun 1985 minat tersebut mulai bangkit kembali setelah penemuan metode pembelajaran yang efektif untuk jaringan neural lapis banyak. Jaringan Propagasi-Balik dikembangkan oleh Rumelhart, Hinton dan Williams dan dipopulerkan pada buku *Parallel Distributed Processing* (Rumelhart and McLelland, 1986).

Prinsip dasar algoritma propagasi-balik memiliki tiga fase:

- Fase *feedforward* pola input pembelajaran
- Fase kalkulasi dan *backpropagation* error yang didapat.
- Fase penyesuaian bobot.

Arsitektur yang digunakan adalah jaringan perceptron lapis banyak (*multi-layer perceptrons*). Hal ini merupakan generalisasi dari arsitektur perceptron lapis tunggal (*single-layer perceptron*). Secara umum, algoritma jaringan ini membutuhkan

waktu pembelajaran yang memang lambat, namun setelah pembelajaran/pelatihan selesai, aplikasinya akan memberikan output yang sangat cepat.

Karakteristik

Jaringan Backpropagation mempunyai karakteristik sebagai berikut :

- Jaringan lapis banyak
  - Terdiri dari satu lapisan unit-unit masukan, satu atau lebih lapisan tersembunyi dan satu lapisan unit keluaran. Arsitektur jaringannya pada dasarnya serupa dengan *perceptron*, namun memiliki lapisan tersembunyi (*hidden layers*), sehingga disebut *multi-layer perceptrons*.
  - Setiap neuron pada suatu lapisan dalam jaringan Propagasi-Balik mendapat sinyal masukan dari semua neuron pada lapisan sebelumnya beserta satu sinyal bias.
- Fungsi aktivasi
  - Fungsi aktivasi untuk jaringan propagasi-balik harus memiliki beberapa karakteristik : kontinyu, dapat didiferensiasikan, dan monoton tidak turun. Juga lebih diinginkan demi efisiensi komputasi, turunan fungsinya juga mudah dihitung. Biasanya, fungsinya diharapkan untuk bersaturasi atau mendekati maksimum dan minimumnya secara asimtotik.
  - Fungsi yang sering digunakan adalah :
    - fungsi sigmoid biner (range : [0,1]),

$$f(x) = \frac{1}{1+e^{-x}} \tag{9}$$

dengan turunannya,

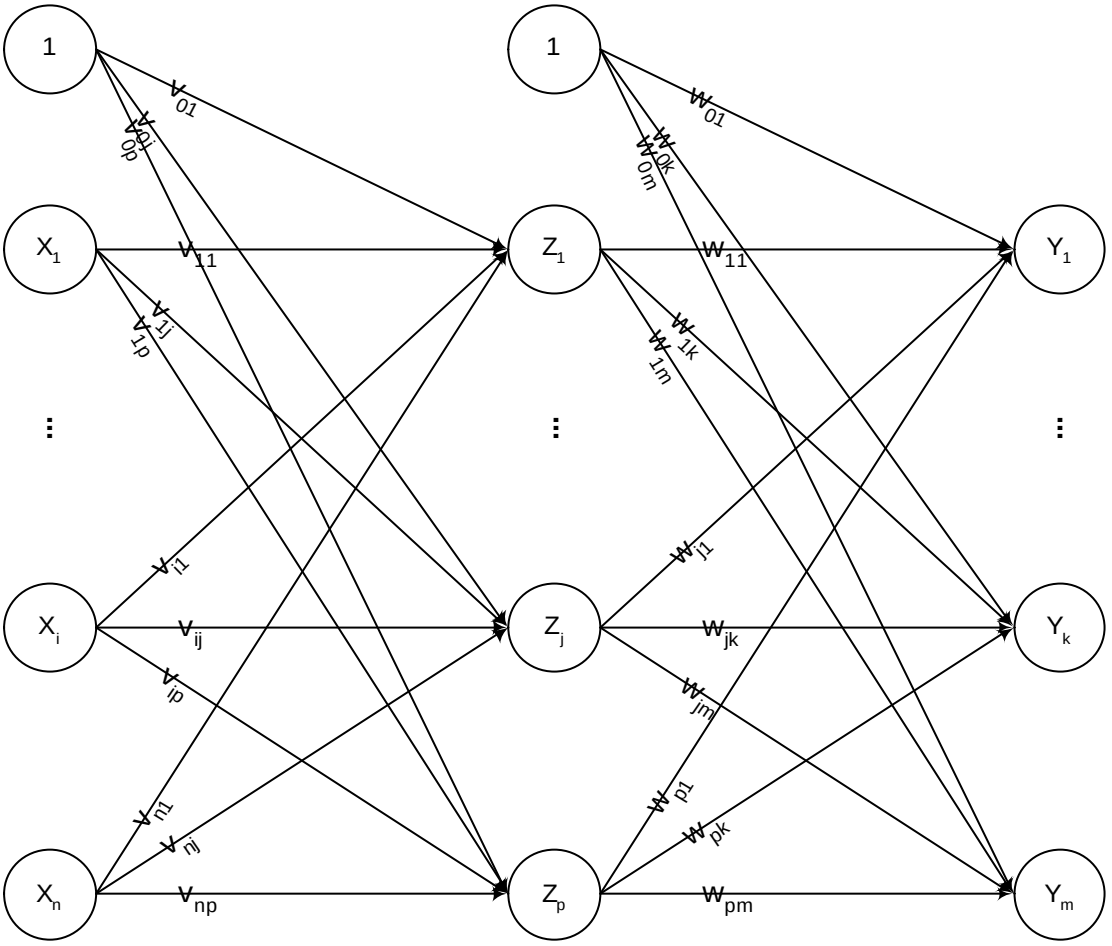
$$f'(x) = f(x)\left[1 - f(x)\right] \tag{10}$$

    - fungsi sigmoid bipolar (range : [-1,1]),

$$f(x) = \frac{2}{1+e^{-x}} - 1 \tag{11}$$

dengan turunannya,

$$f'(x) = \frac{1}{2}\left[1 + f(x)\right]\left[1 - f(x)\right] \tag{12}$$



Gambar 5. Arsitektur Backpropagation

- Arsitektur

  Suatu jaringan neural lapis-banyak (MLP) dengan satu lapis tersembunyi ditunjukkan oleh gambar 5. Pada gambar tersebut, lapis masukan (*input layer*) ditunjukkan oleh unit-unit X_i, sementara lapis keluaran (*output layer*) ditunjukkan oleh unit-unit Y_k. Lapis tersembunyi (*hidden layer*) ditunjukkan oleh unit-unit Z_j. Dalam hal ini, lapis tersembunyi dapat terdiri lebih dari satu lapisan.

  Bias untuk suatu unit output Y_k, diberikan oleh w_{0k}. Bias ini bertindak seolah sebagai bobot pada koneksi yang berasal dari suatu unit/neuron yang keluarannya selalu 1. Unit-unit tersembunyi juga dapat memiliki bias. Aliran sinyal pada gambar dinyatakan dengan arah panah. Sedangkan, pada fase propagasi-balik, sinyal dikirim pada arah yang berlawanan.

- Algoritma
  - Algoritma ini didasarkan pada aturan *error-correction learning rule*.
    Pelatihannya terdiri dari tiga tahap:
    - proses pelatihan pola masukan secara *feedforward*;
    - penghitungan dan propagasi-balik dari *error* yang didapat;
    - penyesuaian bobot.
  - Penerapan JNB cukup dengan melibatkan proses *feedforward*-nya saja.
  - Algoritma ini mengasumsikan hanya terdapat satu lapisan tersembunyi (*hidden layer*) saja. Untuk jaringan dengan lebih dari satu lapisan tersembunyi, algoritma di bawah cukup dimodifikasikan sedikit. Pada algoritma di bawah, pada fase *feedforward* langkah 4 dilakukan berulang-ulang untuk setiap lapisan tersembunyi dengan menganggap sinyal masukan terbobot berasal dari unit di lapisan sebelumnya. Sedangkan pada fase *backpropagation*, langkah 7 dilakukan berulang-ulang untuk setiap lapisan tersembunyi.
    Pada dasarnya satu lapisan tersembunyi sudah cukup untuk sembarang pemetaan kontinyu dari pola input ke pola output pada sembarang tingkat akurasi. Meskipun, dua lapis tersembunyi bisa membuat pembelajaran lebih mudah pada beberapa situasi.
  - Algoritma Pembelajaran

    Langkah 0. Inisialisasi bobot (biasanya digunakan nilai acak yang kecil)
    Set laju pembelajaran α

    Langkah 1. Selama syarat henti salah, lakukan langkah 2 – 9

    Langkah 2. Untuk setiap pasangan pelatihan (masukan dan target), lakukan langkah 3 – 8.

    Feedforward:

    Langkah 3. Setiap unit masukan (X_i, $i = 1, \ldots, n$) menerima sinyal masukan x_i dan meneruskannya ke seluruh unit pada lapisan di atasnya (*hidden units*).

    Langkah 4. Setiap unit tersembunyi (Z_j, $j = 1, \ldots, p$) menghitung total sinyal masukan terbobot,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij},$$

lalu menghitung sinyal keluarannya dengan fungsi aktivasi,

$$z_j = f(z\_in_j),$$

dan mengirimkan sinyal ini ke seluruh unit pada lapisan di atasnya (lapisan output).

Langkah 5. Setiap unit output (Y_k, k = 1, …, m) menghitung total sinyal masukan terbobot,

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} x_j w_{jk},$$

lalu menghitung sinyal keluaran dengan fungsi aktivasi,

$$y_k = f(y\_in_k).$$

Backpropagation of error:

Langkah 6. Setiap unit output (Y_k, k = 1, …, m) menerima sebuah pola target yang sesuai dengan pola masukan pelatihannya. Unit tersebut menghitung informasi kesalahan,

$$d_k = (t_k - y_k) f'(y\_in_k)$$

kemudian menghitung koreksi bobot (digunakan untuk mengubah w_{jk} nanti),

$$\Delta w_{jk} = \alpha d_k z_j$$

dan menghitung koreksi bias

$$\Delta w_{0k} = \alpha d_k,$$

serta mengirimkan nilai d_k ke unit pada lapisan di bawahnya.

Langkah 7. Setiap unit tersembunyi (Z_j, j = 1, …, p) menghitung selisih input (dari unit-unit pada layer di atasnya),

$$d\_in_j = \sum_{k=1}^{m} d_k w_{jk},$$

lalu mengalikannya dengan turunan fungsi aktivasi untuk menghitung informasi errornya,

$$d_j = d\_in_j f'(z\_in_j),$$

selanjutnya menghitung koreksi bobot untuk mengubah v_{ij} nanti,

$$\Delta v_{ij} = a d_j x_i ,$$

dan menghitung koreksi biasnya,

$$\Delta v_{0j} = a d_j .$$

Perubahan bobot dan bias:

Langkah 8. Setiap unit output (Y_k, k = 1, …, m) mengubah bias dan bobot-bobotnya (j = 0, …, p):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} .$$

Setiap unit tersembunyi (Z_j, j = 1, …, p) mengubah bias dan bobotnya (i = 1, …, n):

$$v_{ij}(new) = v w_{ij}(old) + \Delta v_{ij}$$

Langkah 9. Uji syarat henti :

Jika besar *total squared-error* $\sum_{k=1}^{n}(t_k - y_k)^2$ lebih kecil dari toleransi yang telah ditentukan atau jumlah epoh pelatihan sudah mencapai epoh maksimum, maka selesai; jika tidak maka kembali ke langkah 1

Nilai toleransi (ε) yang digunakan adalah $1 < \varepsilon \leq 0$.

- Algoritma Pengujian

Langkah 0. Inisialisasi bobot

(digunakan nilai bobot yang diperoleh dari algoritma pelatihan)

Langkah 1. Untuk setiap vektor masukan x, lakukan langkah 2-4

Langkah 2. Set nilai aktivasi dari unit masukan, i = 1, …., n

$$x_i = s_i$$

Langkah 3. Untuk j = 1, …, p

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} ,$$

$$z_i = f(z\_in_j).$$

Langkah 4. Untuk k = 1, …, m

$$y\_in_k = w_{0k} + \sum_{j=1}^{m} z_j w_{jk} ,$$

$$y_k = f(y\_in_k).$$

- Pemilihan bobot awal dan bias
  - Pemilihan bobot awal mempengaruhi apakah jaringan akan mencapai error minimum global (atau lokal), dan jika tercapai, seberapa cepat konvergensinya.
  - Update bobot tergantung pada fungsi aktivasi unit yang lebih dalam (pemberi sinyal input) dan turunan fungsi aktivasi unit yang lebih luar (penerima sinyal input), sehingga perlu dihindari pemilihan bobot awal yang menyebabkan keduanya bernilai 0.
  - Jika menggunakan fungsi sigmoid, nilai bobot awal tidak boleh terlalu besar karena dapat menyebabkan nilai turunannya menjadi sangat kecil (jatuh di daerah saturasi). Sebaliknya juga tidak boleh terlalu kecil, karena dapat menyebabkan net input ke unit tersembunyi atau unit output menjadi terlalu dekat dengan nol, yang membuat pembelajaran terlalu lambat.
  - Di sini diberikan dua contoh model inisialisasi bobot dan bias.
    - Inisialisasi Acak.

      Bobot dan bias diinisialisasi nilai acak antara -0.5 dan 0.5 (atau antara -1 dan 1, atau pada interval lain yang sesuai).
    - Inisialisasi Nguyen-Widrow.

      Cara ini memberikan laju pembelajaran yang lebih cepat. Berikut contohnya untuk arsitektur dengan satu lapis tersembunyi.
      - Bobot dari unit/neuron tersembunyi ke unit/neuron output diinisialisasi dengan nilai acak antara -0.5 dan 0.5.
      - Bobot dari neuron input ke neuron tersembunyi diinisalisasi sebagai berikut :

        (1). Set :

        n = jumlah unit input

        p = jumlah unit tersembunyi

        ß = faktor skala = $0.7(p)^{1/n} = 0.7\sqrt[n]{p}$.

        (2). Untuk setiap unit tersembunyi ($j = 1, \ldots, p$). lakukan (3) – (6).

        (3). Untuk $i = 1, \ldots, n$ (semua unit input), v_{ij}(old) = bilangan acak antara -0.5 dan 0.5 (atau antara –? dan ?).

        (4). Hitung nilai norm $\|\mathbf{v}_j(\text{old})\|$.

(5). Inisialisasi ulang bobot-bot dari unit input (i = 1, …, n) :

(6). Set bias : v_{0j} = bilangan acak antara –ß dan ß.