Anggota Kelompok:

- Bunga Amelia Restuputri 115060800111114
- Anis Maulida Dyah Ayu Putri 115060801111087
- Inten Widi P 115060800111001
- Regita Ayu P 115060800111028

3.1 Training Algorithms For Pattern Association

3.1.1 Hebb Rule for Pattern Association

Aturan Hebb adalah metode yang termudah dan paling umum untuk menentukan bobot untuk memory assosiatif jaringan syaraf. Ini dapat digunakan dengan pola yang merepresentasikan binary maupun bipolar vektor. Kita menggunakan algoritma untuk input dan output training vektor dan memberikan prosedur umum untuk mencari bobot dari produk luar.

Algorithm

Step 0. Menginisialisasi semua bobot (*i*= 1,....., *n; j*= 1,.....,*m):*

Wij = 0.

Step 1 Untuk setiap input training output vektor s:t mengerjakan step 2-4

Step 2. Mengatur aktivasi untuk unit input ke input training saat ini

(*i* = 1 ,....,n):

x*i* = *si*

Step 3. Mengatur aktivasi untuk output unit ke target output saat ini

(*j* = 1, ..., *m)*:

Yj = tj.

Step 4. Menyesuaikan bobot (*i* = 1,....., *n ; j*= 1,.....,*m)*:

Wij (new) = *wij* (old) + *xiyj.*

Outer Products

Bobot berdasarkan aturan Hebb dapat dideskripsikan di outer product dari input vektor – output vektor. Outer product dengan 2 vektor

$$s = (S_1, \dots\dots\dots, S_i, \dots\dots\dots, S_n)$$

dan

$$t = (t_1, \ldots\ldots\ldots, t_j, \ldots\ldots\ldots, t_m)$$

matriks product dari n x 1 matriks S = S^T dan 1 x m matrils T = t

$$ST = \begin{bmatrix} s_1 \\ . \\ . \\ s_i \\ . \\ . \\ s_n \end{bmatrix} \begin{bmatrix} t_1 \ldots t_j \ldots t_m \end{bmatrix} = \begin{bmatrix} s_1t_1 \ldots . & s_1t_j & \ldots . s_1t_m \\ . & . & . \\ . & . & . \\ s_it_1 \ldots . s_it_j \ldots . & & s_it_m \\ . & . & . \\ . & . & . \\ s_nt_1 \ldots s_nt_j \ldots & & s_nt_m \end{bmatrix}$$

Ini hanya bobot matriks untuk menyimpan asosiasi s:t yang ditemukan menggunakan aturan hebb

Untuk menyimpan set dari asosiasi s(p) : t (p) , p =1,....,P, dimana

s(p) = (s1(p),....., si(p),.....,sn(p))

Dan

t(p) = (t1(p), ...,tj(p),....., tm(p)),

Untuk bobot matrik W = {Wij} didapat dari

$$\mathbf{W} = \sum_{p=1}^{P} \mathbf{s}^T(p)\mathbf{t}(p),$$

3.1.2 Delta Rule for Pattern Association

Aturan delta berasumsi bahwa fungsi aktivasi untuk unit output adalah fungsi identitas.Ekstensi sederhana mengijinkan untuk digunakan pada fungsi aktivasi yang berbeda. Tata nama aturan delta terdiri atas:

α learning rate

x training input vektor

t target output untuk input vektor x

Untuk bobot

$$w_{ij}(new) = w_{ij}(old) + \alpha(t_j - y_j)x_i \qquad (i = 1, \ldots . n; j = 1, \ldots . m).$$

Extended delta rule

Untuk memperbaharui bobot I input unit ke J output adalah

$$\Delta w_{ij} = \alpha(t_j - y_j)x_i f'(y\_in_j)$$

Derivation. Turunan dari aturan delta sudah didiskusikan pada sub bab 2.4.4. kesalahan kuadrat untuk pola pelatihan tertentu adalah

$$E = \sum_{j=1}^{m} (t_j - y_j)^2.$$

E adalah fungsi dari semua bobot. Gradien dari E adalah vektor yang terdiri atas turunan partial dari E untuk masing – masing bobot. Error dapat dikurangi dengan seringnya menyesuaikan bobot Wij pada $\partial E / \partial w_{ij}$. Untuk menemukan rumus $\partial E / \partial w_{ij}$ pada bobot w_{ij}. yang berubah ubah Wij,

$$\frac{\partial E}{\partial w_{IJ}} = \frac{\partial}{\partial w_{IJ}} \sum_{j=1}^{m} (t_j - y_j)^2$$

$$= \frac{\partial}{\partial w_{IJ}} (t_J - y_J)^2,$$

Bobot wij hanya berpengaruh pada error dari output unit Yj.

$$y\_in_J = \sum_{i=1}^{n} x_i w_{iJ} \quad \text{and}$$

$$y_J = f(y\_in_J),$$

Kita memiliki

$$\frac{\partial E}{\partial w_{IJ}} = -2(t_J - y_J) \frac{\partial y\_in_J}{\partial w_{IJ}}$$

$$= -2(t_J - y_J) x_I f'(y\_in_J).$$

Demikian lokal error akan berkurang dengan cepat apabila menyesuaikan bobot pada aturan delta

$$\Delta w_{IJ} = \alpha (t_J - y_J) x_I f'(y\_in_J).$$

3.4 Iterative Associative Net

3.4.1 Reccurent Linear Autoassociator

Merupakan salah satu iterative autoassociator jaringan syaraf tiruan paling sederhana, yang biasa disebut dengan linear autoassociator. Pada jaringan ini memiliki n neuron, yang masing-masing terhubung dengan neuron yang lainnya. Memiliki matriks penimbang (weight matrix) yang simetris dengan bobot wij, dan terdapat dua unit dalam fungsi aktivasi Xi dan Xj. Selain itu, pada n xn nonsingular matriks simetris satu sama lain memiliki orthogonal eigenvectors.

Hasilnya ketika input vektor x adalah x.W, dimana W adalah matriks penimbang (weight matriks). linear autoassociator untuk menghasilkan eigenvector dimana vektor input yang paling mirip.

3.4.2 Brain State In A Box Net

Misalkan W adalah matriks simetris terbesar eigenvalues memiliki komponen positif. Selain itu, W dirubah menjadi nilai positif. Misalkan x(0) menunjukkan vektor keadaan awal . Algoritma BSB didefinisikan dengan persamaan sebagai berikut:

- Definisi fungsi aktivasi (yi) sama dengan input vektor (x),

$$y_i = x_i$$

- Hitung net input(y_in),

$$y\_in = y_i + \propto \sum_j y_j w_{ji}$$

- Output sinya sesuai dengan fungsi aktivasi, sebagai berikut;

$$y_i = \begin{cases} 1, & if\ y\_in > 1 \\ y\_in, & if -1 \le y\_in \le 1 \\ -1, & if\ y\_in < -1 \end{cases}$$

- hitung perubahan bobot dengan rumus:

$$w_{ij}(new) = w_{ij}(old) + \beta y_i y_j$$

3.4.3 Autoassociator With Threshold Function

Fungsi threshold biasa digunakan pada fungsi aktivasi untuk iterative autoassociator net. Fungsi threshold dimplementasikan pada bipolar vektor (+1 atau -1) dan fungsi aktivasi dengan bobot simetris. Contohnya sebagai berikut, dimana bobot awal adalah 0

$$w_{ij} = w_{ji}, w_{ii} = 0$$

Algoritmanya:

- Definisikan fungsi aktivasi (x)
- Rubah tiap unit menggunakan fungsi berikut, dimana $\theta\ adalah\ threshold$;

$$x_i = \begin{cases} 1, & if \sum_j x_j w_{ij} > \theta \\ x_i, & if \sum_j x_j w_{ij} = \theta \\ -1, & if \sum_j x_j w_{ij} < \theta \end{cases}$$

- Lakukan perulangan (iterasi) pada langkah sebelumnya hingga vektor x hampir mirip dengan hasil iterasi vektor tersebut

Vector Masukan Pertama (1,0,0,0)

Iterasi 1

Langkah 4 : (1,0,0,0) W = (0,1,1,-1)

Langkah 5 : (0,1,1,-1) bukan vector tersimpan atau vector aktivasi sebelumnya sehingga harus diperbarui lagi

Iterasi 2

Langkah 4 : (0,1,1,-1) W = (3,2,2,2) -> (1,1,1,-1)

Langkah 5 : (1,1,1,-1) merupakan vector tersimpan sehingga iterasi dihentikan

Jaringan untuk vector input (1,0,0,0) menghasilkan (1,1,1,-1) sebagai respon setelah 2 iterasi.

Vector Masukan Kedua (0,1,0,0)

Iterasi 1

Langkah 4 : (0,1,0,0) W = (1,0,1,-1)

Langkah 5 : (1,0,1,-1) bukan vector tersimpan atau vector aktivasi sebelumnya sehingga harus diperbarui lagi

Iterasi 2

Langkah 4 : (1,0,1,-1) W = (2,3,2,2) -> (1,1,1,-1)

Langkah 5 : (1,1,1,-1) merupakan vector tersimpan sehingga iterasi dihentikan

Jaringan untuk vector input (0,1,0,0) menghasilkan (1,1,1,-1) sebagai respon setelah 2 iterasi.

Vector Masukan Ketiga (0, 0, 1,0)

Iterasi 1

Langkah 4 : (0, 0, 1,0) W = (1, 1, 0,-1)

Langkah 5 : (1, 1, 0,-1) bukan vector tersimpan atau vector aktivasi sebelumnya sehingga harus diperbarui lagi

Iterasi 2

Langkah 4 : (1, 1, 0,-1) W = (2, 2, 3,2) -> (1,1,1,-1)

Langkah 5 : (1,1,1,-1) merupakan vector tersimpan sehingga iterasi dihentikan

Jaringan untuk vector input (0, 0, 1,0) menghasilkan (1,1,1,-1) sebagai respon setelah 2 iterasi.

Vector Masukan Keempat (0, 0, 0, -1)

Iterasi 1

Langkah 4 : (0, 0,0,- 1) W = (1,1,1,0)

Langkah 5 : (1,1,1,0) bukan vector tersimpan atau vector aktivasi sebelumnya sehingga harus diperbarui lagi

Iterasi 2

Langkah 4 : (1,1,1,0) W = (2, 2,2, -3) -> (1,1,1,-1)

Langkah 5 : (1,1,1,-1) merupakan vector tersimpan sehingga iterasi dihentikan

Jaringan untuk vector input (0, 0, 1,0) menghasilkan (1,1,1,-1) sebagai respon setelah 2 iterasi.

Contoh 3.21 pengujian jaring autoassociative berulang: kesalahan dalam komponen pertama dan kedua dari vektor disimpan

Salah satu contoh yang dapat dibentuk dari vector tersimpan (1,1,1,-1) dengan kesalahan dua komponen adalah (-1,-1,1,-1)

Vektor masukan (-1,-1,1,1)

Langkah 4: (-1,-1,1,1) W=(1,1,-1,1)

Langkah 5: iterasi

Langkah 4: (1,1,-1,1) W = (-1,-1,1,-1)

Langkah 5: iterasi berhenti karena mengulang vector masukan.

Iterasi lebih lanjut hanya akan mengulang dua vector aktifasi yang sudah ada secara bergantian. Perilaku jaringan ini disebut fixed point cycle of length two.

3.4.4 Jaringan Diskrit Hopfield

Jaringan iterative associative mirip dengan jaringan yang didiskripsikan pada chapter ini yang dikembangkan oleh Hopfield. Jaringan Hopfield adalah jaringan syaraf yang saling berhubungan antara yang satu dengan yang lainnya.jaringan Hopfield memiliki bobot yang simetris yaitu

$w_{ij} = w_{ji}$ dan $w_{ii} = 0$

Perbedaan jaringan Hopfield dengan yang lainnya adalah sebagai berikut:

1. Hanya satu unit update vector aktivasi pada satu waktu.
2. Masing-masing unit terus menerima sinyal eksternal disamping sinyal dari unit lain.

Pembaruan unit asinkron dati fungsi sebelumnya, dikenal dengan energy atau fungsi Lyapunov. Adanya fungsi ini memungkinkan kita untuk memuktikan bahwa jaringan dapat menemukan set aktivasi yang stabil. Fungsi ini dikembangkan oleh insinyur dan matematikawan Rusia, Alexander Mikhailovich Lyapunov (1857-1918).

Formula original dari jaringan diskri Hopfield menunjukkan kegunaan jaringan sebagai pengalamatan isi memori.

Arsitektur

Perluasan dari keadaan biasa pada jaringan Hopfield adalah

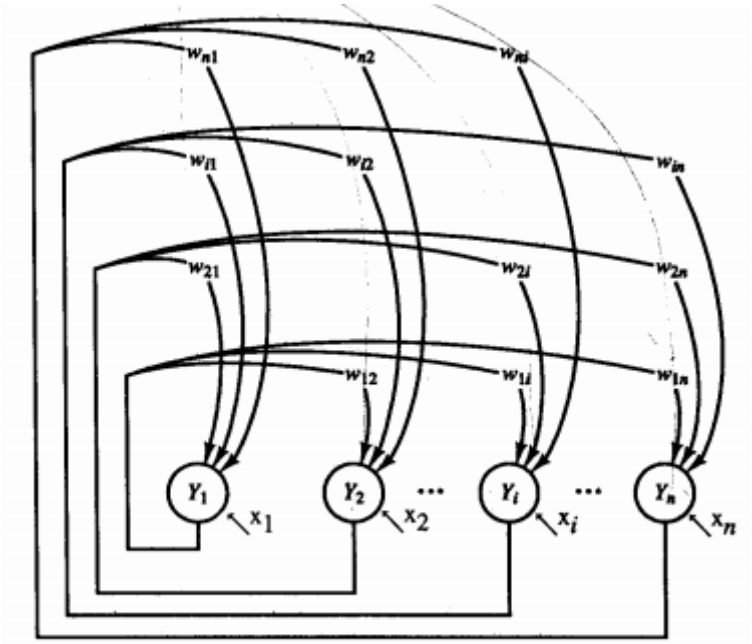

Figure 3.7 Discrete Hopfield net.

Algoritma

Terdapat beberapa versi dari jaringan diskrit Hopfield, yaitu:

1. Untuk menentukan pola binery
   S(p) = (s_1(p),…,s_i(p),…,s_n(p)) dimana P = 1,….,P
   W = $\{w_{ij}\}$ -> $w_{ij}=\sum_p[2si(P)-1][2sj(p)-1]$ untuk I ≠ j dan $w_{ii}=0$
2. Untuk menentukan pola bipolar
   S(p) = (s_1(p),…,s_i(p),…,s_n(p)) dimana P = 1,….,P
   W = $\{w_{ij}\}$ -> $w_{ij}=\sum_p si(p)sj(p)$ untuk I ≠ j dan $w_{ii}=0$

Aplikasi algoritma untuk jaringan diskrit Hopfield

Langkah 0 : inisialisasi bobot (W) -> gunakan aturan Hubb

Langkah 1 : untuk setian nilai vector x, lakukan langkah 2-6

Langkah 2 : set inisial aktivasi dari jaringan sebagai eksternal input vector x

$y_i = x_i$ (I = 1,…,n)

Langkah 3 : lakukan langkah 4-6 untuk masing-masing Y_i

Langkah 4 : hitung inputan jaringan

$Y\_in_i = x_i + \sum_j y_j w_{ji}$

Langkah 5 : tentukan aktivasi (sinyal output)

$$\begin{cases} 1 & jika\ y_{in} > \theta \\ y_i & jika\ y_{in} = \theta \\ 0 & jika\ y_{in} < \theta \end{cases}$$

Langkah 6 : menyebarkan nilai y_i ke semua unit

Langkah 7 : test

Threshold (θ) biasanya diberikan nilai 0. Urutan update unit acak, namun masing-masing unit harus di-update dengan nilai rata-rata sama.

Aplikasi

Sebuah jaringan Hopfield biner dapat digunakan untuk menentukan apakah masukan suatu vektor adalah vektor yang "diketahui" atau vektor yang "tidak diketahui". Jaringan mengenali sebuah vektor yang "diketahui" dengan memproduksi sebuah pola aktivasi pada unit jaringan yang sama dengan vektor yang disimpan dalam jaringan tersebut. Jika masukan vektor adalahvektor yang "tidak diketahui", vektor-vektor yang aktif diproduksi sebagai iterasi-iterasi jaringan yang akan berkumpul untuk sebuah vektor aktivasi yang bukan merupakan salah satu pola yang disimpan, pola seperti ini disebut *keadaan stabil palsu*.

Contoh 3.22 Pengujian Jaringan Hopfield diskrit : kesalahan dalam komponen pertama dan kedua dari vektor disimpan.

Pertimbangkan, vektor (1,1,1,0) (atau setara bipolar nya (1,1,1, -1) disimpan dalam jaringan.Masukan biner sesuai dengan input (0,0,1,0) .Meskipun jaringan Hopfield menggunakan vektor biner, matriks bobotnya adalah bipolar. Unit memperbarui aktivasi mereka dalam urutan acak. Untuk contoh ini urutan update adalah Y_1, Y_4, Y_3, Y_2.

Step 0, Inisialisasi bobot untuk pola penyimpanan :

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}.$$

Step 1, vektor input x = (0,0,1,0). Untuk vektor ini,

Step 2, y = (0,0,1,0)

Step 3, pilih satuan Y_1 untuk memperbarui aktivasi:

Step 4. $y\_in_1 = x_1 + \sum_j y_j w_{j1} = 0 + 1.$

Step 5. $y\_in_1 > 0 \rightarrow y_1 = 1.$

Step 6. $\mathbf{y} = (1, 0, 1, 0).$

Step 3, pilih satuan Y_4 untuk memperbarui aktivasi:

Step 4. $y\_in_4 = x_4 + \sum_j y_j w_{j4} = 0 + (-2).$

Step 5. $y\_in_4 < 0 \rightarrow y_4 = 0.$

Step 6. $\mathbf{y} = (1, 0, 1, 0).$

Step 3, pilih satuan Y_3 untuk memperbarui aktivasi:

Step 4. $y\_in_3 = x_3 + \sum_j y_j w_{j3} = 1 + 1.$

Step 5. $y\_in_3 > 0 \rightarrow y_3 = 1.$

Step 6. $\mathbf{y} = (1, 0, 1, 0).$

Step 3, pilih satuan Y_2 untuk memperbarui aktivasi:

Step 4. $y\_in_2 = x_2 + \sum_j y_j w_{j2} = 0 + 2.$

Step 5. $y\_in_2 > 0 = y_2 = 1.$

Step 6. $y = (1, 1, 1, 0).$

Step 7, tes untuk konvergensi

karena beberapa aktivasi telah berubah selama siklus pembaruan ini, setidaknya harus dibuat satu lagi untuk melewati semua vektor-vektor masukan. Pembaca dapat mengkonfirmasi bahwa iterasi selanjutnya tidak mengubah aktivasi setiap unit. Jaringan telah berkumpul untuk vektor disimpan.

Analisis

Fungsi Energi. Hopfield [1984] membuktikan bahwa jaringan diskrit akan berkumpul ke sebuah titik limit yang stabil dengan mempertimbangkan energi (atau Lyapunov) fungsi untuk sistem. Fungsi energi adalah fungsi yang dibatasi di bawah dan merupakan fungsi nonincreasing(Fungsi Tidak Bertambah) dari keadaan sistem. Untuk jaringan saraf, keadaan dari sistem adalah vektor aktivasi dari unit. Jadi, jika fungsi energi dapat ditemukan untuk jaring saraf berulang, maka jaringan akan berkumpul untuk satu set aktivasi yang stabil. Fungsi energi untuk jaringan diskrit Hopfield diberikan oleh:

$$E = -.5 \sum_{i \neq j} \sum_j y_i y_j w_{ij} - \sum_i x_i y_i + \sum_i \theta_i y_i.$$

Jika aktivasi dari jaringan berubah dengan jumlah Δy_i, maka perubahan energi dengan jumlah yang

$$\Delta E = -\left[\sum_j y_j w_{ij} + x_i - \theta_i\right] \Delta y_i.$$

Kita sekarang mempertimbangkan dua kasus di mana perubahan Δy_i akan terjadi dalam aktivasi neuron y_i.

jika y_i positif, hal itu akan berubah menjadi nol jika

$$x_i + \sum_j y_j w_{ji} < \theta_i.$$

ini memberikan perubahan negatif untuk y_i, Dalam hal ini, $\Delta E < 0$.

Jika y_i adalah nol, itu akan berubah menjadi positif jika

$$x_i + \sum_j y_j w_{ji} > \theta_i.$$

ini memberikan perubahan positif bagi y_i, Dalam hal ini, $\Delta E < 0$.

Jadi Δy_i positif hanya jika $[\Sigma y_j w_{ji} + x_i - \theta_i]$ *positif*, dan Δy_i negatif hanya jikarumus yang sama ini adalah negatif. Oleh karena itu, energi tidak dapat meningkat. Dikarenakan energi dibatasi, jaringan harus mencapai keseimbangan stabil sehingga energi tidak berubah dengan iterasi lebih lanjut.

Bukti ini menunjukkan bahwa aktivasi tidak perlu menjadi biner. Hal ini juga tidak penting apakah sinyal eksternal diselenggarakan selama iterasi. Aspek penting dari algoritma adalah bahwa perubahan energi hanya tergantung pada perubahan aktivasi satu unit dan bobot matriks akan simetris dengan nol pada diagonal.

Kapasitas Penyimpanan. Hopfield menemukan eksperimental bahwa jumlah pola biner yang dapat disimpan dan dipanggil kembali dalam jaringan dengan cukup akurat, diberikan oleh

$$P = 0.15n,$$

di mana *n* adalah jumlah neuron di jaringan.

Abu-Mostafa dan St Jacques (1985) telah melakukan analisis teoritis dengan rinci tentang kapasitas informasi dari jaringan Hopfield. Untuk jaringan serupa dengan menggunakan pola bipolar, McEliece, Posner, Rodemich, dan Venkatesh (1987) menemukan bahwa

$$P \approx \frac{n}{2 \log_2 n}.$$