

BAB V

METODE BELAJAR HEBBIAN

- Aturan Hebb merupakan salah satu hukum pembelajaran jaringan neural yang pertama. Dikemukakan oleh Donald Hebb (1949). Hebb lahir di Chester, Nova Scotia, pada pergantian abad.
- Isinya menyangkut kemungkinan mekanisme modifikasi sinaptik dalam otak, yang kemudian digunakan untuk melatih JST.
- Pada tahun 1949 Hebb merangkum hasil penelitian yang telah dilakukannya selama 20 tahun ke dalam buku yang diberi judul "*The Organization of Behavior*", yang pada intinya mengatakan bahwa perilaku dapat dijelaskan melalui aksi-aksi neuron.
- Ide yang paling terkenal di dalam buku Hebb di atas adalah sebuah postulat yang kemudian dikenal dengan nama metode belajar Hebb :

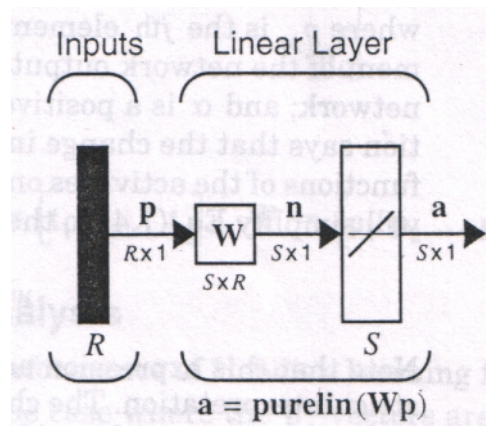
"Jika axon sebuah sel A cukup dekat untuk bisa mengeksitasi sel B dan secara berulang atau terus menerus melakukan penembakan, beberapa proses atau perubahan metabolisme akan terjadi pada satu atau kedua sel, sehingga efisiensi sel A, sebagai salah satu sel penembak B, akan meningkat."

Postulat ini diusulkan sebagai mekanisme fisik untuk proses pembelajaran pada tingkat selular. Walaupun Hebb tidak pernah memberikan bukti fisiologis yang signifikan mengenai teorinya, namun penelitian-penelitian selanjutnya menunjukkan bahwa sejumlah sel menunjukkan pola belajar Hebbian.

5.1 Asosiator Linier

- Hukum pembelajaran Hebb dapat diimplementasikan pada berbagai arsitektur jaringan neural.

- Salah satu arsitektur jaringan neural yang sederhana adalah asosiator linier (*linear associator*), seperti terlihat pada gambar berikut ini :



- Keluaran vektor **a** dihitung dari vektor masukan **p** sbb. :

$$\mathbf{a} = \mathbf{Wp}$$

atau

$$a_i = \sum_{j=1}^Q w_{ij} p_j$$

- Asosiator linear merupakan contoh dari jaringan neural yang dinamakan memori asosiatif (*associative memory*).
- Memori asosiatif didisain untuk mempelajari Q pasang vektor contoh masukan / keluaran.

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

Dengan kata lain, jika jaringan menerima input $\mathbf{p} = \mathbf{p}_q$ maka ia harus menghasilkan keluaran $\mathbf{a} = \mathbf{t}_q$, dengan $q = 1, 2, \dots, Q$. Jika masukan sedikit berubah (mis., $\mathbf{p} = \mathbf{p} + \delta$), maka keluaran juga sedikit berubah (mis., $\mathbf{a} = \mathbf{t}_q + \epsilon$).

5.2 Aturan Hebb

- Merupakan interpretasi matematis postulat Hebb.

- Dari rumus keluaran neuron $a_i = \sum_{j=1}^Q w_{ij} p_j$, terlihat bahwa hubungan (sinapsis) antara masukan p_j dan keluaran a_i berupa bobot w_{ij} .

- Postulat Hebb mengatakan bahwa jika p_j positif menghasilkan a_i positif, maka w_{ij} harus naik. Ekspresi matematikanya :

$$w_{ij}^{new} = w_{ij}^{old} + \alpha f_i(a_{iq}) g_j(p_{jq})$$

dengan p_{jq} adalah elemen ke- j dari vektor masukan \mathbf{p}_q ke- q ; a_{iq} adalah elemen ke- i dari keluaran jaringan pada saat vektor masukan ke- q digunakan jaringan; dan α adalah konstanta positif, dinamakan *learning rate*.

- Dari persamaan terlihat bahwa perubahan bobot w_{ij} proporsional terhadap perkalian fungsi aktivitas pada kedua sisi sinapsis. Persamaan di atas dapat disederhanakan menjadi :

$$w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq}$$

Ekspresi ini merupakan perluasan dari postulat Hebb.

- Aturan belajar Hebb di atas termasuk aturan belajar *tanpa supervisi* (*unsupervised learning rule*), yang tidak membutuhkan informasi apapun mengenai target keluaran.

- Terdapat juga aturan Hebb dengan supervisi (*supervised Hebb rule*), dimana pada persamaannya, keluaran aktual digantikan dengan keluaran target. Persamaannya :

$$w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq}$$

dengan t_{iq} adalah elemen ke - i dari vektor target t ke - q . (Untuk memudahkan, *learning rate* diset bernilai satu)

Dalam bentuk notasi vektor :

$$W^{new} = W^{old} + t_q p_q^T$$

Jika diasumsikan bahwa matriks bobot diinisialisasi dengan nilai 0, dan pasangan masukan / keluaran Q digunakan satu kali, maka :

$$W = t_1 p_1^T + t_2 p_2^T + \dots + t_Q p_Q^T = \sum_{q=1}^Q t_q p_q^T$$

Ini juga dapat direpresentasikan dalam bentuk matriks :

$$W = [t_1 t_2 \dots t_Q] \begin{bmatrix} p_1^T \\ p_2^T \\ \dots \\ p_Q^T \end{bmatrix} = TP^T$$

dengan $T = [t_1 t_2 \dots t_Q]$, $P = [p_1 p_2 \dots p_Q]$

5.3 Analisis Performa

- Jika p_q adalah vektor-vektor orthonormal (orthogonal dan bernilai 1) dan p_k adalah masukan jaringan, maka keluaran jaringan dapat dihitung :

$$a = Wp_k = \left(\sum_{q=1}^Q t_q p_q^T \right) p_k = \sum_{q=1}^Q t_q (p_q^T p_k)$$

Karena p_q orthonormal,

$$(p_q^T p_k) = 1 \text{ jika } q = k$$

$$= 0 \text{ jika } q \neq k$$

Oleh karena itu persamaan di atas dapat ditulis :

$$a = Wp_k = t_k$$

yang berarti keluaran jaringan sama dengan keluaran target. Hal ini menunjukkan bahwa *jika contoh vektor masukan bersifat orthonormal, aturan Hebb akan menghasilkan keluaran yang benar untuk setiap input.*

- Jika vektor masukan bersifat *non-orthogonal* (namun bernilai satu) :

$$a = Wp_k = t_k + \boxed{\text{Error !}} \sum_{q \neq k} t_q (p_q^T p_k)$$

Karena tidak bersifat orthogonal, jaringan tidak akan menghasilkan keluaran yang benar. Besarnya kesalahan akan tergantung pada besar korelasi antara pola-pola masukan contoh.

- Ada sejumlah prosedur yang dapat mengurangi kesalahan ini, antara lain *pseudoinverse rule* :

$$\mathbf{W} = \mathbf{TP}^+$$

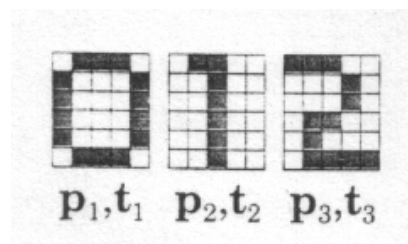
Jika jumlah baris (R) pada \mathbf{P} lebih besar dari jumlah kolom (Q) \mathbf{P} dan kolom-kolom \mathbf{P} bersifat independen, maka *pseudoinverse rule* dapat dihitung dengan :

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1} \mathbf{P}^T$$

5.4 Aplikasi

Berikut ini ditunjukkan penggunaan aturan Hebb secara praktis, yaitu pada masalah pengenalan pola (yang sangat disederhanakan). Untuk masalah ini, digunakan jenis khusus dari memori asosiatif – *autoassociative memory*. Pada *autoassociative memory* vektor keluaran yang diinginkan sama dengan vektor input (yaitu $\mathbf{t}_q = \mathbf{p}_q$). Di sini *associative memory* digunakan untuk menyimpan satu set pola dan pola tersebut akan dipanggil kembali, termasuk dengan menggunakan masukan yang rusak.

Pola yang akan disimpan adalah sebagai berikut :



Yang merupakan vektor input sekaligus target (karena di sini digunakan *autoassociative memory*). Vektor-vektor tersebut merepresentasikan bilangan (0,1,2) dalam kisi 6 x 5. Digit ini harus dikonversi menjadi vektor, untuk dijadikan pola contoh bagi jaringan. Setiap kotak putih direpresentasikan dengan "-1" dan kotak hitam direpresentasikan dengan "1". Selanjutnya, untuk membuat vektor

input, kisi 6 x 5 tersebut dibaca per kolom. Sebagai contoh, vektor pola pertama adalah

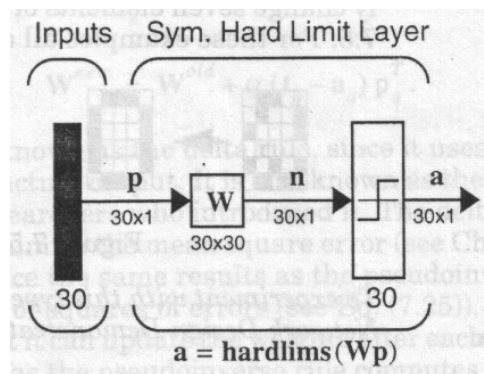
$$\mathbf{p}_1 = [-1 \ 1 \ 1 \ 1 \ 1 \ 1 \ -1 \ 1 \ -1 \ -1 \ -1 \ -1 \ 1 \ 1 \ -1 \ \dots \ 1 \ -1]^T$$

Vektor \mathbf{p}_1 adalah untuk digit "0", \mathbf{p}_2 untuk "1", dan \mathbf{p}_3 untuk "2". Dengan menggunakan aturan Hebb, bobot matriks dihitung

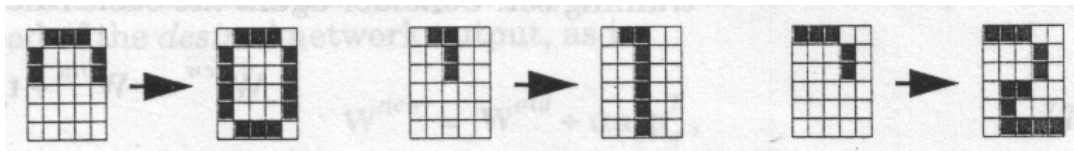
$$\mathbf{W} = \mathbf{p}_1 \mathbf{p}_1^T + \mathbf{p}_2 \mathbf{p}_2^T + \mathbf{p}_3 \mathbf{p}_3^T$$

(Perhatikan bahwa \mathbf{p}_q menggantikan posisi \mathbf{t}_q , karena ini adalah *autoassociative memory*).

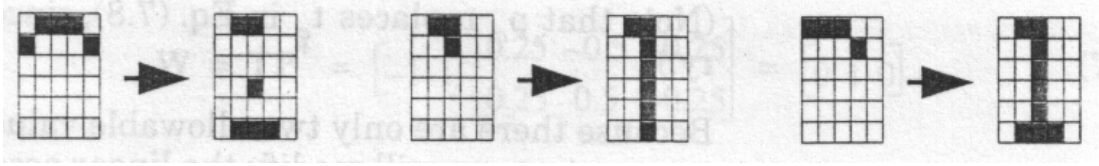
Karena hanya ada elemen-elemen vektor hanya memiliki dua kemungkinan nilai ("-1" dan "1") maka fungsi transfer linier digantikan dengan fungsi transfer *symmetrical hard limit*.



Selanjutnya dapat dilihat bahwa jika pada jaringan diberi masukan berupa pola 50 % (lihat gambar di bawah ini), ternyata jaringan dapat memberi keluaran yang benar.



Jika pada jaringan diberi masukan berupa pola yang mengalami kerusakan 67%, hasilnya adalah sebagai berikut.



Jika jaringan diberi masukan berupa pola terdistorsi, keluaran jaringan adalah sebagai berikut.

