BAB V
ALGORITMA PEMBELAJARAN DALAM JARINGAN SYARAF TIRUAN

Kompetensi :
1. Mahasiswa memahami konsep pembelajaran dalam JST

Sub Kompetensi :
1. Dapat mengetahui prinsip algoritma Perceptron
2. Dapat mengetahui prinsip pembelajaran pada algoritma Back Propagasi
3. Dapat mengetahui prinsip pembelajaran pada algoritma Kohonen Map

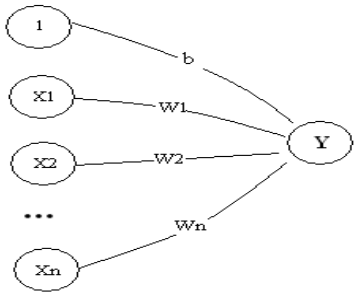
Dalam bab ini hanya akan dipaparkan 3 algoritma yang banyak digunakan dalam Jaringan Syaraf Tiruan, yaitu Algoritma Perceptron, Algoritma Backpropagasi dan Algoritma Kohonen. Algoritma perceptron dan Backpropagasi merupakan contoh algoritma dengan jenis pembelajaran yang terawasi (supervising) dan algoritma kohonen merupakan contoh algoritma yang tidak terawasi (unsupervising)

V.I. Algoritma Perceptron

Model jaringan perceptron ditemukan oleh Rosenblatt (1962) dan Minsky – Papert (1969). Model tersebut merupakan model yang memiliki aplikasi dan pelatihan yang paling baik pada era tersebut.

1. Arsitektur Jaringan

Arsitektur jaringan perceptron mirip dengan arsitektur jaringan Hebb.



Gambar 1. Arsitektur Jaringan Perceptron

Jaringan terdiri dari beberapa unit masukan (ditambah sebuah bias), dan memiliki sebuah unit keluaran. Hanya saja fungsi aktivasi merupakan fungsi biner (atau bipolar), tetapi memiliki kemungkinan nilai -1, 0 atau 1.

Untuk suatu harga threshold θ yang ditentukan :

$$f(net) = \begin{cases} 1 & \text{Jika net} > \theta \\ 0 & \text{jika } -\theta \leq \text{net} \leq \theta \\ -1 & \text{jika net} < -\theta \end{cases}$$

Secara geometris fungsi aktivasi membentuk 2 garis sekaligus, masing – masing dengan persamaan :

$$W_1X_1 + W_2X_2 + \ldots + W_nX_n + b = \theta \text{ dan}$$

$$W_1X_1 + W_2X_2 + \ldots + W_nX_n + b = -\theta$$

2. Pelatihan Perceptron

   Misalkan

   S adalah vector masukan dan t adalah target keluaran

   α adalah laju pemahaman (learning rate) yang ditentukan

   θ adalah threshold yang ditentukan.

   Algoritma pelatihan perceptron adalah sebagai berikut :

   a. Inisialisasi semua bobot dan bias (umumnya $w_i = b\ 0$)

      Tentukan laju pemahaman (=α). Untuk penyederhanaan biasanya α diberi nilai = 1

   b. Selama ada elemen vector masukan yang respon unit keluarannya tidak sama dengan target, lakukan :

      1) Set aktivasi unit masukan $x_i = s_i$ (i = 1,…n)
      2) Hitung respon unit keluaran : net = $\sum x_i w_i + b$

$$f(net) = \begin{cases} 1 & \text{Jika net} > \theta \\ 0 & \text{jika } -\theta \leq \text{net} \leq \theta \\ -1 & \text{jika net} < -\theta \end{cases}$$

3) Perbaiki bobot pola yang mengandung kesalahan ($y \neq t$) menurut persamaan :

W_i (baru) = w_i (lama) + Δ w (i=1,…n) dengan $\Delta w = \alpha t x_i$

b (baru) = b (lama) + Δ b dengan $\Delta b = \alpha t$

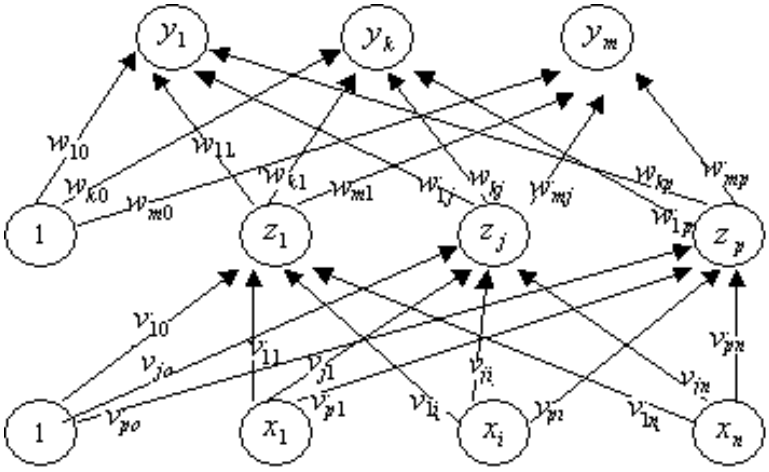
Ada beberapa hal yang perlu diperhatikan dalam algoritma tersebut :

a) Iterasi dilakukan terus hingga semua pola memiliki keluaran jaringan yang sama dengan targetnya (jaringan sudah memahami pola). Iterasi tidak berhenti setelah semua pola dimasukan seperti yang terjadi pada model Hebb.
b) Perubahan bobot hanya dilakukan pada pola yang mengandung kesalahan (keluaran jaringan ≠ target). Perubahan tersebut merupakan hasil kali unit masukan dengan target laju pemahaman. Perubahan bobot hanya akan terjadi kalau unit masukan ≠ 0.

V.2. Algoritma Backpropagasi Umpan Balik (Feed Forward Back Propagation)

Propagasi umpan balik berbasis jaringan syaraf tiruan merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot- bobot yang terhubung dengan neuron – neuron yang ada pada lapisan tersembunyinya. Propagasi umpan balik berbasis jaringan syaraf tiruan menggunakan *error output* untuk mengubah nilai bobot – bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron – neuron diaktifkan dengan menggunakan fungsi aktivasi yang dapat dideferensiasi.

Propagasi umpan balik berbasis jaringan syaraf tiruan memiliki beberapa unit yang ada dalam satu atau lebih layer tersembunyi. Gambar 2.3 adalah arsitektur propagasi umpan balik berbasis jaringan syaraf tiruan dengan *n* buah masukan (ditambah sebuah bias), sebuah layar tersembunyi yang terdiri dari *p* unit (ditambah sebuah bias), serta *m* buah unit keluaran (Jek, 2005:98).



Gambar 2.Arsitektur Propagasi Umpan Balik Berbasis Jaringan Syaraf Tiruan

v_{ji} merupakan bobot garis dari unit masukan x_i ke unit layar tersembunyi z_j (v_{jo} merupakan bobot garis yang menghubungkan bias di unit masukan ke unit layer tersembunyi z_j). w_{kj} merupakan bobot dari unit layar tersembunyi z_j ke unit keluaran y_k (w_{ko} merupakan bobot dari bias di layar tersembunyi ke unit keluaran z_k) (Jek, 2005:98).

1. Pelatihan Propagasi Umpan Balik Berbasis Jaringan Syaraf Tiruan

Pelatihan propagasi umpan balik berbasis jaringan syaraf tiruan meliputi 3 fase. Fase pertama adalah fase maju. Pola masukan dihitung maju mulai dari layar masukan hingga layar keluaran menggunakan fungsi aktivasi yang ditentukan. Fase kedua adalah fase mundur. Selisih antara keluaran jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasikan mundur, dimulai dari garis yang berhubungan langsung dengan unit-unit di layar keluaran. Fase ketiga adalah modifikasi bobot untuk menurunkan kesalahan yang terjadi (Jek, 2005:100).

- Fase I: Propagasi Maju

Selama propagasi maju, sinyal masukan ($=x_i$) dipropagasikan ke layer tersembunyi menggunakan fungsi aktivasi yang ditentukan. Keluaran dari setiap unit layar tersembunyi ($=z_j$) tersebut selanjutnya dipropagasikan maju lagi ke layer tersembunyi di atasnya menggunakan

fungsi aktivasi yang ditentukan. Demikian seterusnya hingga menghasilkan keluaran jaringan (= y_k).

Berikutnya, keluaran jaringan (= y_k) dibandingkan dengan target yang harus dicapai (= t_k). Selisih dari t_k terhadap y_k yaitu ($t_k - y_k$) adalah kesalahan yang terjadi. Jika kesalahan ini lebih kecil dari batas toleransi yang ditentukan, maka iterasi dihentikan. Akan tetapi apabila kesalahan masih lebih besar dari batas toleransinya, maka bobot setiap garis dalam jaringan akan dimodifikasi untuk mengurangi kesalahan yang terjadi.

- Fase II: Propagasi Mundur

Berdasarkan kesalahan $t_k - y_k$, dihitung faktor $\delta_k (k = 1,2,..., m)$ yang dipakai untuk mendistribusikan kesalahan di unit y_k ke semua unit tersembunyi yang terhubung langsung dengan y_k. δ_k juga dipakai untuk mengubah bobot garis yang berhubungan langsung dengan unit keluaran.

Dengan cara yang sama, dihitung faktor δ_j ($j = 1,2,\ldots, p$) di setiap unit di layar tersembunyi sebagai dasar perubahan bobot semua garis yang berasal dari unit tersembunyi di layar di bawahnya. Demikian seterusnya hingga semua faktor δ di unit tersembunyi yang berhubungan langsung dengan unit masukan dihitung.

- Fase III: Perubahan Bobot

Setelah semua faktor δ dihitung, bobot semua garis dimodifikasi bersamaan. Perubahan bobot suatu garis didasarkan atas faktor δ neuron di layar atasnya. Sebagai contoh, perubahan bobot garis yang menuju ke layar keluaran didasarkan atas δ_k yang ada di unit keluaran.

Ketiga fase terebut diulang-ulang terus hingga kondisi penghentian dipenuhi. Umumnya kondisi penghentian yang sering dipakai adalah jumlah iterasi atau kesalahan. Iterasi akan dihentikan jika jumlah iterasi yang dilakukan sudah melebihi jumlah maksimum iterasi yang ditetapkan, atau jika kesalahan yang terjadi sudah lebih kecil dari batas toleransi yang diijinkan.

Algoritma pelatihan untuk jaringan dengan satu layer tersembunyi (dengan fungsi aktivasi sigmoid biner) adalah sebagai berikut :

Langkah 0 : Inisialisasi semua bobot dengan bilangan acak kecil.

Langkah 1 : Jika kondisi penghentian belum terpenuhi, lakukan langkah 2 – 9.

Langkah 2 : Untuk setiap pasang data pelatihan, lakukan langkah 3 – 8.

Fase I : Propagasi maju

Langkah 3 : Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi di atasnya.

Langkah 4 : Hitung semua keluaran di unit tersembunyi z_j ($j = 1,2,\ldots, p$).

$$z\_net_j = v_{jo} + \sum_{i=1}^{n} x_i v_{ji}$$

$$z_j = f(z\_net_j) = \frac{1}{1+e^{-z\_net_j}}$$

Langkah 5 : Hitung semua keluaran jaringan di unit y_k ($k = 1,2,\ldots, m$).

$$y\_net_k = w_{ko} + \sum_{j=1}^{p} z_j w_{kj}$$

$$y_k = f(y\_net_k) = \frac{1}{1+e^{-y\_net_k}}$$

Fase II : Propagasi mundur

Langkah 6 : Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran y_k ($k = 1,2,\ldots, m$).

$$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_k - y_k)\, y_k (1 - y_k)$$

δk merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layer dibawahnya (langkah 7).

Hitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj}) dengan laju perceptron α

$$\Delta w_{kj} = \alpha \delta_k z_j \quad ; \quad k = 1,2,\ldots,m \; ; \; j = 0,1,\ldots, p$$

Langkah 7 : Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi z_j ($j = 1,2,\ldots, p$).

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k \, w_{kj}$$

Faktor δ unit tersembunyi :

$$\delta j = \delta\_netj \, f'(z\_netj) = \delta\_net\, j \, Zj \, (1 - Zj)$$

Hitung suku perubahan bobot Vji (yang akan dipakai nanti untuk merubah bobot Vji).

$$\Delta vji = \alpha \, \delta j \, xi \quad ; \; j = 1,2,\ldots,p \quad ; I = 0,1,\ldots,n$$

Fase III : Perubahan bobot

Langkah 8 : Hitung semua perubahan bobot.

Perubahan bobot garis yang menuju ke unit keluaran :

Wkj (baru) = Wkj (lama) + ΔWkj (k = 1,2,…,m ; j = 0,1,…,p)

Perubahan bobot garis yang menuju ke unit tersembunyi :

Vji (baru) = Vji (lama) + Δvji (j = 1,2…,p ; I = 0,1,…,n)

Setelah selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola. Dalam hal ini, hanya propagasi maju (langkah 4 dan 5) saja yang dipakai untuk menentukan keluaran jaringan.

Apabila fungsi aktivasi yang dipakai bukan sigmoid biner, maka langkah 4 dan 5 harus disesuaikan. Demikian juga turunannya pada langkah 6 dan 7.

2. Jaringan Propagasi Umpan Balik Pada MATLAB

Membangun Jaringan Propagasi Umpan Balik

Arsitektur jaringan yang sering digunakan oleh algoritma propagasi umpan balik adalah jaringan *feedforward* dengan banyak lapisan. Untuk membangun suatu jaringan *feedforward* digunakan instruksi *newff*.

Fungsi :

net = newff (PR,[S1 S2 . . . SN1],{TF1 TF2 . . . TFN1},BTF,BLF,PF)

Keterangan :

PR : matriks berukuran Rx2 yang berisi nilai *minimum* dan *maksimum*, dengan R adalah jumlah variable input .

Si : jumlah neuron pada lapisan ke-I, dengan I = 1,2,..,N1.

TFi : fungsi aktivasi pada lapisan ke- I, dengan I = 1,2,..,N1; (*default* : tansig)

BTF : fungsi pelatihan jaringan (*default* : trainlm)

BLF : fungsi pelatihan untuk bobot (*default* : learngdm)

PF : fungsi kinerja (*default* : mse)

Inisialisasi Bobot

Setiap kali membentuk jaringan propagasi umpan balik, MATLAB akan memberi nilai bobot dan bias awal dengan bilangan acak kecil. Bobot dan bias ini akan berubah setiap kali kita membentuk jaringan. Akan tetapi jika diinginkan memberi bobot tertentu, bisa dilakukan dengan memberi nilai pada net.IW, net.LW dan net.b.

Perbedaan antara net.IW dan net.LW. net.IW{j,i} digunakan sebagai variabel untuk menyimpan bobot dari unit masukan layar i ke unit tersembunyi (atau unit keluaran) layar j.

Karena dalam propagasi umpan balik, unit masukan hanya terhubung dengan layar tersembunyi paling bawah, maka bobotnya disimpan dalam net.IW {1,1}. Sebaliknya, net.LW{k,j} dipakai untuk menyimpan bobot dari unit di layar tersembunyi ke- j ke unit di layar tersembunyi ke- k. Sebagai contoh, net.LW {2,1} adalah penyimpan bobot dari layar tersembunyi paling bawah (layar tersembunyi ke- 1) ke layar tersembunyi diatasnya (layar tersembunyi ke- 2).

Metode Penurunan Gradien Dengan Momentum (Traingdm)

Modifikasi metode penurunan tercepat dilakukan dengan menambah momentum. Dengan momentum, perubahan bobot tidak hanya didasarkan atas error yang terjadi pada epoch pada waktu itu. Perubahan bobot saat ini dilakukan dengan memperhitungkan juga perubahan bobot pada epoch sebelumnya. Dengan demikian kemungkinan terperangkap ke titik minimum lokal dapat dihindari.

Dalam MATLAB, pelatihan propagasi umpan balik dengan menggunakan metode penurunan gradient dengan momentum dilakukan dengan mendefinisikan fungsi pelatihan ‘ traingdm’ dalam pembentukan jaringannya.

Ada beberapa parameter yang harus diset untuk pelatihan ini, yaitu :

- Maksimum epoh

  Maksimum epoh adalah jumlah epoh maksimum yang boleh dilakukan selama proses pelatihan. Iterasi akan dihentikan apabila nilai epoh melebihi maksimum epoh.

  Instruksi : **net.trainParam.epochs = MaxEpoh** (default = 10).
- Kinerja tujuan

  Kinerja tujuan adalah target nilai fungsi kinerja. Iterasi akan dihentikan apabila nilai fungsi kinerja kurang dari atau sama dengan kinerja tujuan.

  Instruksi : **net.trainParam.goal = Targeterror** (default = 0).
- Learning rate

  Learning rate adalah laju pembelajaran. Semakin besar nilai learning rate akan berimplikasi pada semakin besarnya langkah pembelajaran. Jika learning rate diset terlalu besar, maka algoritma akan menjadi tidak stabil. Sebaliknya, jika learning rate diset terlalu kecil, maka algoritma akan konvergen dalam jangka waktu yang sangat lama.

  Instruksi : **net.trainParam.lr = LearningRate**(default = 0.01).
- Faktor laju momentum

  Instruksi : **net.trainParam.mc = Momentum** (default = 0.9).
- Jumlah epoh yang akan ditunjukkan kemajuannya

  Menunjukkan berapa jumlah epoh berselang yang akan ditunjukkan kemajuannya.

  Instruksi : **net.trainParam.show = EpohShow** (default = 25).

Pelatihan Propagasi Umpan Balik

Untuk menggunakan pelatihan propagasi umpan balik digunakan fungsi train.

Fungsi :

[net,tr] = train (net,P,T)

Keterangan :

net : jaringan syaraf

tr : informasi pelatihan (epoh dan fungsi kinerja)

P : matriks data input

T : matriks data target (default = 0)

V.3. Algoritma Kohonen Map

Kohonen Map atau bisa disebut *Self Organizing Map* diperkenalkan pertama kali oleh Prof. Teuvo Kohonen dari Finlandia pada tahun 1982. Kohonen map merupakan salah satu algoritma jaringan syaraf tiruan terbaik, metoda ini cukup unik karena membangun sebuah *topology preserving map* dari ruang berdimensi tinggi ke dalam neuron-neuron sebagai representasi dari datapoint-datapoint yang ada.

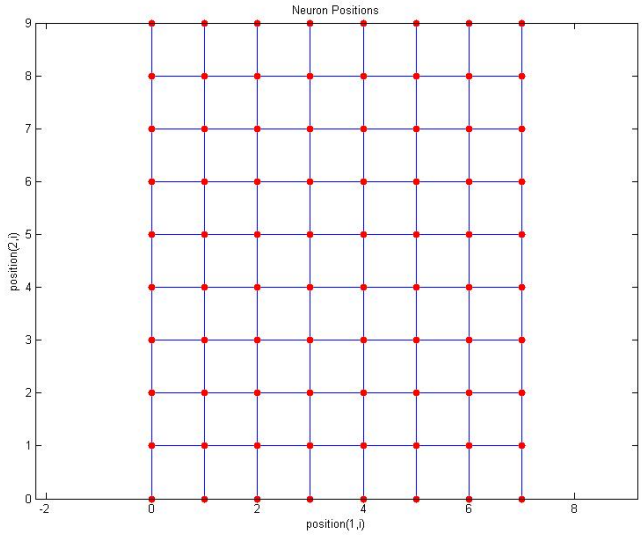
Kohonen map merupakan salah satu metoda jaringan syaraf tiruan *unsupervised* (tidak terawasi), jaringan ini tidak mendapatkan target, sehingga JST mengatur bobot interkoneksi sendiri. Belajar tanpa pengawasan kadang-kadang diacu sebagai *Self Organizing learning*, yakni belajar mengklasifikasikan tanpa dilatih. Pada proses belajar tanpa pengawasan, JST akan mengklasifikasikan contoh pola-pola masukan yang tersedia ke dalam kelompok yang berbeda-beda. Ketika data diberikan ke dalam jaringan syaraf, data akan mengatur struktur dirinya sendiri untuk merefleksikan dari pola yang diberikan. Pada kebanyakan model ini, batasan mengacu pada determinasi kekuatan antar neuron.

1. Topologi Jaringan Kohonen Map

J. J. Siang (2004:292); Dalam jaringan kohonen, neuron target tidak diletakan dalam sebuah baris, tetapi neuron target diletakan dalam 2 dimensi yang topologinya dapat diatur. Untuk mendefinisikan topologi jaringan terdapat 3 macam topologi yang dapat dibuat yaitu *gridtop*, *hextop*, dan *randtop.*

1. Gridtop (*Grid Topology*)

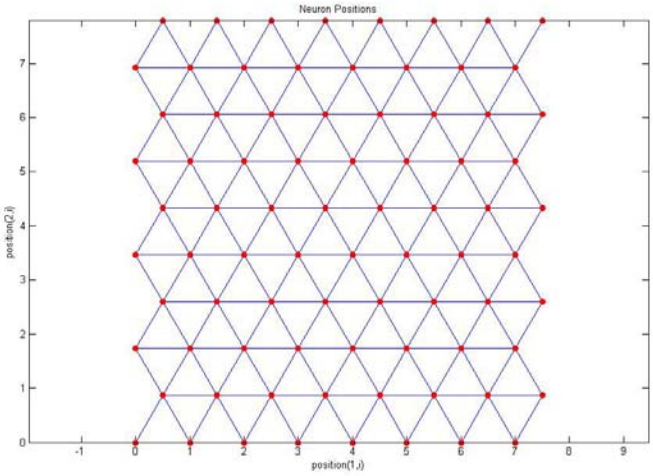
   Topologi dengan posisi neuron membentuk pola menyerupai segi empat. Ditentukan dengan membuat koordinat neuron dalam n baris dan n kolom.



Gambar 3. Grid Topology

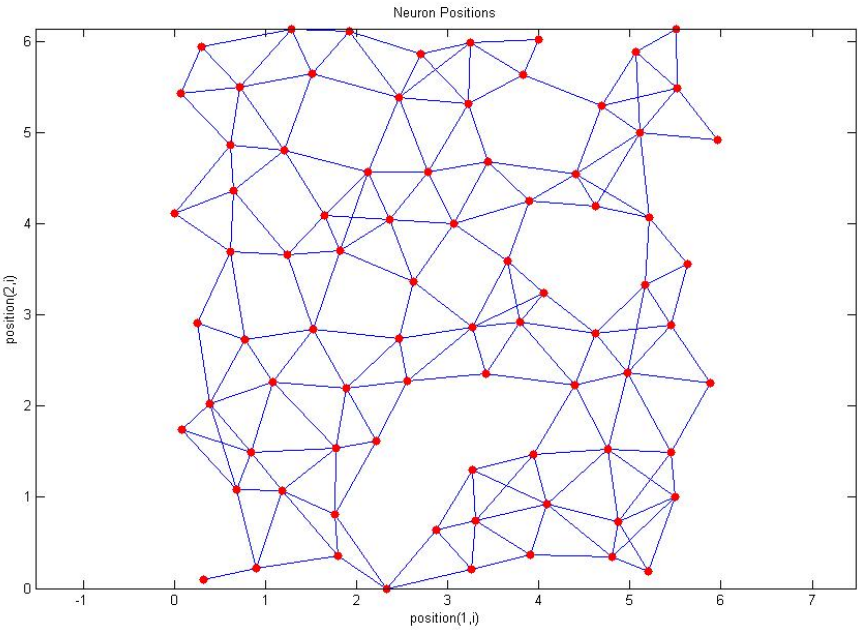
2. Hextop (*Hexagonal Topology*)

   Topologi dengan posisi neuron membentuk pola menyerupai segi enam. Ditentukan dengan membuat koordinat neuron dalam n baris dan n kolom.



Gambar 4. Hexagonal Topology

3. Randtop (*Random Topology*)
   Topologi dengan posisi neuron membentuk pola secara acak. Ditentukan dengan membuat koordinat neuron dalam n baris dan n kolom.



Gambar 5. Random Topology

2. Jarak Antar Neuron Jaringan Kohonen Map

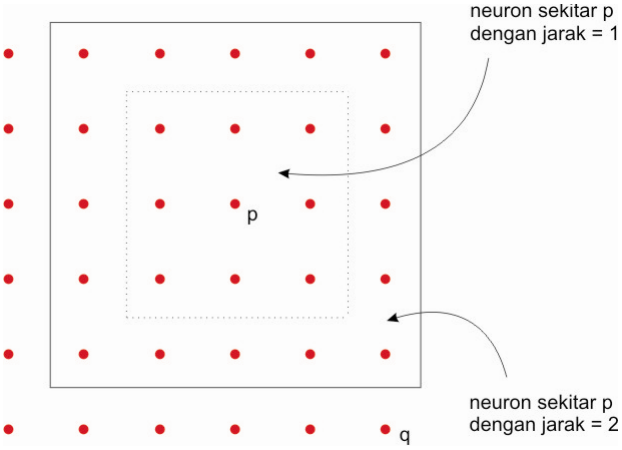
J. J. Siang (2004:297); Dalam jaringan kohonen, perubahan bobot tidak hanya dilakukan pada garis yang terhubung ke neuron pemenang saja, tetapi juga pada bobot garis neuron-neuron sekitarnya. Neuron sekitar neuron pemenang ditentukan berdasarkan jaraknya dari neuron pemenang (*default* jarak = 1). Ada 4 macam definisi jarak antar 2 neuron, yaitu jarak *Euclidist*, *Boxdist, linkdist, dan mandist*.

Jarak *Euclidist* adalah jarak yang antara 2 titik dalam posisi berbeda yang kita kenal sehari-hari. Misal (x_1, y_1) dan (x_2, y_2) adalah koordinat 2 buah neuron. Jarak neuron didefinisikan sebagai berikut:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

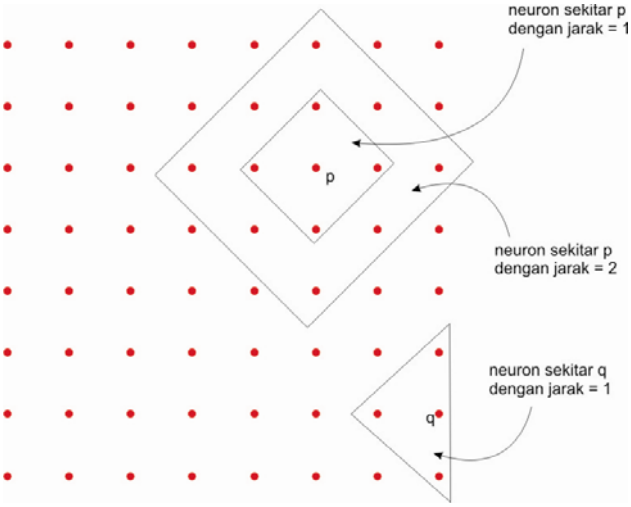
Jarak persegi *(boxdist)* sebuah neuron adalah neuron-neuron disekitarnya secara langsung. Apabila topologi neuron adalah *gridtop*, maka paling banyak terdapat 8 buah neuron dengan *boxdist* = 1, 16 buah neuron dengan *boxdist* = 2 (lihat gambar 2.11). jika neuron

pemenang berada di pinggir (seperti titik q pada gambar 2.11), maka hanya ada 5 neuron disekitarnya.



Gambar 6. Box Distance

Jarak *link (linkdist)* dari sebuah neuron adalah jumlah langkah yang dibutuhkan untuk menuju neuron tersebut. Jika dalam jaringan kohonen menggunakan topologi *gridtop* dengan *linkdist* = 1 (lihat gambar 2.12), berarti hanya neuron-neuron yang berhubungan langsung dengan neuron pemenang saja yang diubah bobotnya.



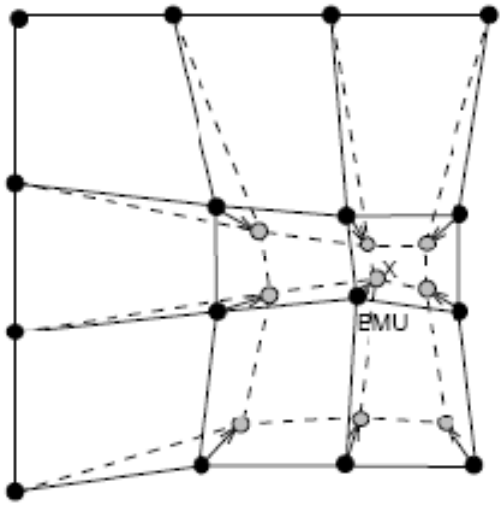
Gambar 7. Link Distance

Jarak Manhattan *(mandist)* antar vektor x = (x_1, x_2, ... x_n) dan vektor y = (y_1, y_2, ... y_n) didefinisikan sebagai: sum (abs(x-y)) = $\sum_{i=1}^{n} |x_i - y_i|$

Jika x = (x_1, x_2) dan y = (y_1, y_2) menyatakan koordinat neuron yang dibentuk melalui topologi tertentu, maka jarak manhattan antara neuron x dan y adalah:

$$D = |x_1 - y_1| + |x_2 - y_2|$$

Juha Vesanto (1999:1) Kohonen dapat berfikir sebagai jaringan yang meyebar dari data masukan. Algoritma pelatihan kohonen memindahkan bobot vektor sehingga jaringan hasil pembelajaran dapat menyerupai pola masukan dan oleh karena itu jaringan dapat terorganisir, neuron yang berdekatan pada jaringan dapat menyrupai bobot vektor. Pada contoh pelatihan sederhana, pola masukan dihubungka satu persatu, dan algoritma hasil pelatihan secara berturut – turut memindahkan bobot vektor ke arah jaringan hasil pelatihan, ditunjukan pada gambar 2.13. Pada proses pelatihan, seperangkat data dipelajari oleh kohonen secara keseluruhan, dan vektor bobot baru menimbang dari rata-rata vektor masukan. Keduanya adalah algoritma iteratif dengan kumpulan yang banyak, sejak operasi matrix dapat menggunakan matlab membuatnya jadi lebih efisien.



Gambar 9. Bobot Baru Jaringan Kohonen

Bobot baru mencocokan pada bobot masukan, dan tetangga dari vektor masukan di tandai dengan x.Garis padat dan garis putus-putus sesuai dengan situasi sebelum dan setelah pembelajaran pada masing – masing neuron.