BAB 3
PATTERN MATCHING BERBASIS JARAK EUCLID, PATTERN MATCHING BERBASIS JARAK MAHALANOBIS, DAN JARINGAN SYARAF TIRUAN BERBASIS PROPAGASI BALIK

Proses pengenalan dilakukan dengan beberapa metode. Pertama dengan menggunakan Jaringan Syaraf Tiruan Berbasis Propagasi Balik. Kedua dengan menggunakan metode *Pattern Matching* Berbasis Jarak Euclid. Yang terakhir adalah menggunakan metode *Pattern Matching* Berbasis Jarak Mahalanobis. Bab ini akan membahas mengenai ketiga metode pengenalan tersebut.

3.1 *Pattern Matching* Berbasis Jarak Euclid

Metode *Pattern Matching* Berbasis Jarak Euclid atau ED digunakan sebagai metode pengenalan iris. Penggunaan metode ini disarankan oleh pembimbing dengan pertimbangan bahwa prosesnya tidak terlalu rumit. Penggunaan metode ini ditujukan untuk membandingkan hasilnya dengan penggunaan BPNN.

Metode ini menggunakan jarak euclid sebagai kriteria pengenalan. Subbab ini akan menjelaskan tentang jarak euclid, hubungan jarak euclid dengan pengenalan, dan algoritma yang digunakan untuk pengenalan dengan jarak euclid.

Jarak euclid adalah suatu nilai yang didapatkan ketika kita mengukur seberapa jauhnya suatu titik X dari titik lain Y. Dengan kata lain, jarak euclid adalah panjang garis lurus yang menghubungkan dua buah titik X dan Y dalam suatu ruang euclid. Panjang garis antara dua buah dapat dihitung dengan cara sebagai berikut.

- Terdapat dua buah titik dalam ruang euclid yaitu $X = [x_1, x_2, \ldots, x_n]$ dan $Y = [y_1, y_2, \ldots, y_n]$
- Jarak euclid antara X dan Y dapat dihitung dengan persamaan 3.1

$$ED(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n \quad y_n)^2} \tag{3.1}$$

Dengan metode ini suatu gambar objek dianalogikan sebagai suatu titik dalam ruang gambar (ruang euclid). Oleh karena itu untuk menentukan apakah dua buah gambar

berasal dari kelas yang sama dapat dilakukan dengan menghitung jarak dari kedua buah gambar tersebut dalam ruang euclid (jarak euclid). Semakin mirip kedua buah gambar maka semakin kecil jarak euclidnya.

Pengenalan dengan jarak euclid ini dilakukan melalui dua metode. Yang pertama adalah dengan menggukanan vektor rata-rata dan yang kedua adalah dengan perbandingan langsung.

3.1.1 Vektor Rata-rata

Dengan metode ini, setiap kelas memiliki vektor rata-rata pada ruang euclid. Vektor rata-rata tersebut dibuat dari data pelatihan.

Untuk menentukan apakah suatu gambar dikenali atau tidak, gambar tersebut dihitung jarak euclidnya terhadap setiap vektor rata-rata. Jika jarak minimum gambar tersebut didapatkan terhadap vektor rata-rata kelas yang sama maka gambar tersebut dikenali.

1. Bagi data gambar menjadi dua bagian, data pelatihan $A=[a_1,a_2,\ldots,a_p]$ dan data pengujian $B=[b_1,b_2,\ldots,b_p]$ dimana $a_i=[a_{i1},a_{i2},\ldots,a_{in}]$ dan $b_i=[b_{i1},b_{i2},\ldots,b_{im}]$, $i=[1,2,\ldots,p]$.
   $p=$ banyaknya kelas
   $n=$ banyaknya gambar untuk suatu kelas i
   $m=$ banyaknya gambar untuk suatu kelas i
2. Buat vektor rata-rata $\overline{A}=[\overline{a}_1,\overline{a}_2,\ldots,\overline{a}_p]$ untuk masing-masing kelas dari data pelatihan dengan persamaan 3.2.

$$\overline{a}_i=\frac{a_{i1}+a_{i2}+\ldots+a_{in}}{n} \tag{3.2}$$

3. Untuk masing-masing data pengujian $D=[d_1,d_2,\ldots,d_p]$, hitung jaraknya terhadap vektor rata-rata setiap kelas seperti pada persamaan 3.3.

$$d_h=ED(\overline{a}_i,b_{jk}) \tag{3.3}$$

$h = [1,2,\ldots,p]$

$i = [1,2,\ldots,p]$

$j = [1,2,\ldots,p]$

$k = [1,2,\ldots,m]$

4. Cari jarak minimum dan indeks dari jarak minimum tersebut dengan persamaan 3.24 dan 3.5.

$$MD = \min(D) \quad (3.4)$$

$$x = index(MD) \quad (3.5)$$

5. Jika jarak minimum yang didapatkan adalah terhadap vektor rata-rata dari kelas yang sama dengan data pengujian yaitu untuk data b_{jk} , nilai $x = j$ maka data b_{jk} tersebut dikenali.
6. Jika jarak minimum yang didapatkan adalah bukan terhadap vektor rata-rata dari kelas yang sama dengan data pengujian yaitu untuk data b_{jk} , nilai $x \neq j$ maka data b_{jk} tersebut tidak dikenali.

3.1.2 Perbandingan Langsung

Berbeda dengan metode sebelumnya, metode ini tidak mememrlukan vektor rata-rata. Setiap gambar pengujian dicari jaraknya dengan setiap gambar pelatihan. Jika jarak minimum didapatkan terhadap gambar pelatihan yang memiliki kelas yang sama denga gambar pengujian maka gambar pengujian tersebut dikenali.

1. Bagi data gambar menjadi dua bagian.

   Data pelatihan $A = [a_1, a_2, \ldots, a_p]$ dan data pengujian $B = [b_1, b_2, \ldots, b_p]$.

   $a_i = [a_{i1}, a_{i2}, \ldots, a_{in}]$ dan $b_i = [b_{i1}, b_{i2}, \ldots, b_{im}]$, $i = [1,2,\ldots,p]$.

   $p =$ banyaknya kelas

   $n =$ banyaknya gambar pelatihan untuk suatu kelas i

   $m =$ banyaknya gambar pengujian untuk suatu kelas i

2. Untuk masing-masing data pengujian $D=[d_1,d_2,\ldots,d_q]$, hitung jaraknya terhadap setiap data pelatihan dengan persamaan 3.6.

$$\begin{aligned} & d_h = ED(a_{ik},b_{jl}) \\ & h=[1,2,\ldots,q] \\ & i=[1,2,\ldots,p] \\ & j=[1,2,\ldots,p] \\ & k=[1,2,\ldots,n] \\ & l=[1,2,\ldots,m] \end{aligned} \tag{3.6}$$

   Simpan juga nilai indeks i dan j untuk masing masing d .
3. Cari jarak minimum dengan persamaan 3.4.
4. Jika jarak minimum yang didapatkan adalah terhadap data pelatihan dari kelas yang sama dengan data pengujian yaitu jika nilai $i=j$ untuk jarak minimum maka gambar tersebut dikenali.
5. Jika jarak minimum yang didapatkan adalah bukan terhadap data pelatihan dari kelas yang sama dengan data pengujian yaitu jika nilai $i \neq j$ untuk jarak minimum maka gambar tersebut tidak dikenali.

Proses pengenalan dengan Statistical Euclidean Distance ini diimplementasikan dalam penelitian dengan program Java. Karena tidak melibatkan angka acak, pengenalan dengan metode ini dilakukan sebanyak satu kali untuk setiap jenis vektor masukan.

3.2 *Pattern Matching* Berbasis Jarak Mahalanobis

Metode pengenalan berikutnya yang digunakan dalam penelitian ini adalah dengan menggunakan *Pattern Matching* Berbasis Jarak Mahalanobis atau MD. Penggunaan metode pengenalan ini disarankan oleh pembimbing karena tingkat pengenalan yang dihasilkan dengan dua metode lainnya masih kurang tinggi sementara vektor masukan tidak dapat dibuat lebih baik lagi.

Pada subbab ini akan dibahas mengenai definisi jarak mahalanobis serta algoritma yang digunakan untuk melakukan pengenalan dengan bantuan jarak mahalanobis.

Berbeda dengan jarak euclid, jarak mahalanobis menghitung seberapa jauhnya suatu titik terhadap sekumpulan titik lain. Perhitungan ini melibatkan distribusi dari sekumpulan titik lain tersebut sehingga hasil yang didapatkan lebih akurat daripada jarak euclid.

Untuk suatu vektor sembarang $X = [x_1, x_2, \ldots, x_n]$, jarak mahalanobisnya terhadap suatu kelas k yang memiliki vektor rata-rata $\bar{a}$ dan vektor kovarian c daapt dihitung dengan persamaan 3.7.

$$MD(k,b) = \sqrt{(b-\bar{a})c^{-1}(b-\bar{a})^T} \tag{3.7}$$

Proses pengenalan dengan jarak mahalanobis adalah sebagai berikut

1. Bagi data gambar menjadi dua bagian.
   Data pelatihan $A = [a_1, a_2, \ldots, a_p]$ dan data pengujian $B = [b_1, b_2, \ldots, b_p]$.
   $a_i = [a_{i1}, a_{i2}, \ldots, a_{in}]$ dan $b_i = [b_{i1}, b_{i2}, \ldots, b_{im}]$, $i = [1, 2, \ldots, p]$.
   $p =$ banyaknya kelas
   $n =$ banyaknya gambar pelatihan untuk suatu kelas i
   $m =$ banyaknya gambar pengujian untuk suatu kelas i
2. Dari data pelatihan.
   a. Buat vektor rata-rata $\bar{A} = [\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_p]$ untuk masing-masing kelas dengan persamaan 3.8.
      $$\bar{a}_i = \frac{a_{i1} + a_{i2} + \ldots + a_{in}}{n} \tag{3.8}$$
   b. Buat vektor kovarian $C = [c_1, c_2, \ldots, c_p]$ untuk masing-masing kelas.
3. Untuk setiap data pengujian $D = [d_1, d_2, \ldots, d_p]$, hitung jarak mahalanobis terhadap setiap kelas seperti pada persamaan 3.9.

$$d_i = MD(k_i, b_{jk}), i = [1,2,\ldots,p], j = [1,2,\ldots,p], k = [1,2,\ldots,m] \quad (3.9)$$

4. Cari jarak minimum dan indeks dari jarak minimum tersebut dengan persamaan 3.4 dan 3.5
5. Jika jarak minimum yang didapatkan adalah terhadap kelas yang sama dengan data pengujian yaitu untuk data b_{jk} , nilai $x = j$ maka data b_{jk} tersebut dikenali.
6. Jika jarak minimum yang didapatkan adalah bukan terhadap kelas yang sama dengan data pengujian yaitu untuk data b_{jk} , nilai $x \neq j$ maka data b_{jk} tersebut tidak dikenali.

Proses pengenalan dengan jarak mahalanobis ini diimplementasikan dalam penelitian dengan program Matlab. Karena tidak melibatkan angka acak, pengenalan dengan metode ini dilakukan sebanyak satu kali untuk setiap jenis vektor masukan.

3.3 Jaringan Syaraf Tiruan Berbasis Propagasi Balik

Jaringan Syaraf Tiruan Berbasis Propagasi Balik atau biasa disebut *Backpropagation Neural Network* (BPNN) digunakan sebagai salah satu metode pengenalan iris. BPNN digunakan dalam penelitian atas saran dari pembimbing dengan pertimbangan bahwa BPNN sudah sering digunakan pada penelitian sebelumnya. Parameter yang mempengaruhinya sudah diketahui sehingga untuk penelitian ini dapat dipakai parameter yang optimal dari penelitian sebelumnya.

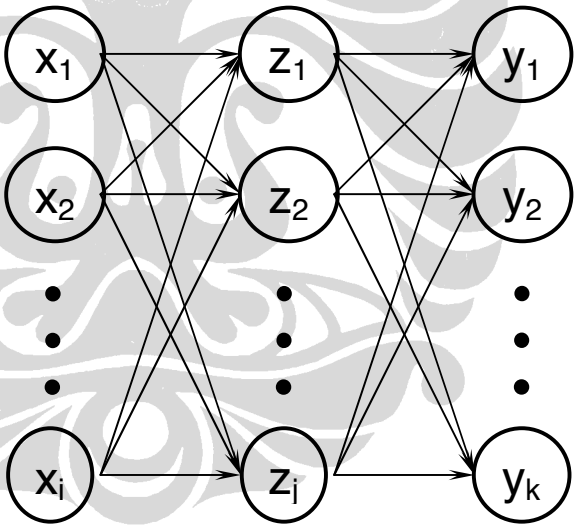
Pada subbab ini akan dijelaskan mengenai BPNN; struktur, cara kerja, serta algoritma yang digunakan. Dalam penelitian ini BPNN diimplementasikan seluruhnya dengan program Java.

3.3.1 Struktur dan Cara Kerja

BPNN merupakan sekumpulan neuron yang terstruktur menjadi tiga lapisan. Lapisan tersebut adalah lapisan masukan *X*, lapisan keluaran *Y* dan lapisan tersembunyi *Z*. Lapisan *Y* dan *Z* masing-masing memiliki bobot *V* dan *W* yang digunakan untuk menghitung masukan untuk kedua lapisan tersebut.

BPNN menerima suatu vektor masukan x yang merupakan suatu representasi vektor satu dimensi dari objek yang akan dikenali. Besarnya vektor masukan berbeda-beda sesuai dengan ukuran objek yang akan dikenali. Secara umum semakin besar ukuran vektor masukan semakin buruk kinerja BPNN.

Vektor masukan akan diproses di dalam BPNN untuk menghasilkan vektor keluaran *y*. Vektor keluaran adalah suatu vektor satu dimensi yang berisi nilai-nilai pengenalan untuk masing-masing kelas dari objek yang sedang dikenali. Pada proses pelatihan, vektor keluaran dibandingkan dengan vektor target *t* untuk menghitung nilai kesalahan yang akan digunakan untuk megoreksi bobot *V* dan *W*. Pada proses pengujian, vektor keluaran akan langsung diperiksa untuk menentukan apakan vektor masukan dikenali atau tidak.



Gambar 3.1 – Struktur Backpropagation Neural Network

Lapisan masukan memiliki neuron sebanyak besarnya vektor masukan. Ukuran vektor masukan yang masih mentah cukup besar (sekitar tiga ribu). Oleh karena itu vektor masukan mentah diproses dengan PCA terlebih dahulu untuk mereduksi ukurannya menjadi lima puluh.

Lapisan tersembunyi memiliki neuron yang jumlahnya bisa ditentukan dengan bebas. Jumlah neuron pada lapisan tersembunyi yang efektif adalah sekitar setengah dari jumlah neuron pada lapisan masukan. Oleh karena itu jumlah neuron pada lapisan tersembunyi adalah tiga puluh.

Lapisan keluaran memiliki neuron sebanyak jumlah objek yang akan dikenali. Pada tahap pelatihan setiap neuron di lapisan ini akan mengeluarkan nilai yang akan digunakan untuk menghitung nilai kesalahan. Nilai kesalahan digunakan untuk mengubah nilai bobot *V* dan *W*. Pada tahap pengujian setiap neuron di lapisan ini mengeluarkan nilai yang akan dievaluasi untuk menentukan apakah masukan suatu objek dikenali atau tidak.

BPNN bekerja dalam dua tahap utama, yaitu pelatihan dan pengujian. Pada tahap pelatihan, BPNN diberi masukan data pelatihan. Data pelatihan akan diproses untuk menghasilkan vektor keluaran. Vektor keluaran lalu dibandingkan dengan target untuk mendapatkan nilai koreksi bobot dan bias untuk lapisan tersembunyi dan lapisan keluaran. Di akhir tahap pelatihan, nilai bobot dan bias disimpan untuk digunakan dalam tahap pengujian.

Pelatihan dilakukan dalam satuan epoch. Satu epoch adalah proses pelatihan untuk seluruh data pelatihan. Pelatihan berakhir ketika nilai kesalahan dalam satu epoch sudah melewati batas bawahnya (BPNN sudah siap untuk mengenali data pengujian). Untuk mengantisipasi waktu pelatihan yang lama epoch memiliki batas atas. Ketika jumlah epoch sudah melebihi batas tersebut pelatihan dihentikan.

Pada tahap pengujian, BPNN diberi masukan data pengujian. Data pengujian akan diproses untuk menghasilkan vektor keluaran. Berbeda dengan tahap pelatihan, pada tahap pengujian vektor keluaran akan langsung dievaluasi untuk menentukan apakah vektor masukan dikenali atau tidak dikenali. Pada tahap pengujian tidak ada proses koreksi bobot dan bias.

Agar suatu BPNN dapat bekerja dengan maksimal, ada beberapa faktor yang mempengaruhinya. Foktor-faktor tersebut adalah laju pembelajaran, momentum, fungsi aktivasi, inisialisasi bobot awal, dan fungsi kesalahan epoch.

3.3.2 Laju pembelajaran

Laju pembelajaran atau biasa dilambangkan dengan α merupakan salah satu parameter yang mengendalikan proses penyesuaian bobot. Laju pembelajaran memiliki nilai optimal yang berbeda-beda sesuai dengan objek yang sedang dikenali.

Laju pembelajaran yang terlalu kecil menyebabkan konvergensi jaringan menjadi lambat sehingga proses pelatihan berjalan dengan lama. Laju pembelajaran yang terlalu besar membuat BPNN menjadi tidak stabil karena laju pembelajaran yang terlalu besar dapat menyebabkan penyesuaian bobot yang lebih besar daripada seharusnya.

Pada penelitian ini digunakan nilai laju pembelajaran sebesar 0.2. Nilai ini merupakan nilai yang cukup optimal jika digunakan pada pengenalan wajah. Untuk pengenalan iris nilai ini mungkin berbeda namun penulis tidak mengubah nilai ini selama penelitian karena penelitian ini bukan bertujuan untuk mencari parameter BPNN yang optimal untuk pengenalan iris.

3.3.3 Momentum

Momentum yang biasa dilambangkan dengan μ adalah parameter yang digunakan untuk mempercepat proses pelatihan BPNN. Tanpa momentum penyesuaian bobot pada saat t akan berjalan secara independen, tidak terpengaruh oleh penyesuaian bobot pada saat $t-1$. Dengan momentum penyesuaian bobot pada saat t dipengaruhi juga oleh penyesuaian bobot pada saat $t-1$.

Pada penelitian ini digunakan nilai momentum sebesar 0.4. Nilai ini merupakan nilai yang cukup optimal jika digunakan pada pengenalan wajah. Untuk pengenalan iris nilai ini mungkin berbeda namun penulis tidak mengubah nilai ini selama penelitian

karena penelitian ini bukan bertujuan untuk mencari parameter BPNN yang optimal untuk pengenalan iris.

3.3.4 Fungsi Aktivasi

Untuk menghasilkan keluaran dari setiap lapisan digunakan fungsi aktivasi. Lapisan masukan memiliki fungsi identitas $f(x) = x$ sebagai fungsi aktivasinya. Lapisan lainnya bisa saja menggunakan fungsi aktivasi yang sama namun agar kinerja BPNN menjadi lebih baik fungsi aktivasi tersebut harus memenuhi syarat berikut:

- Bersifat kontinyu dan dapat diturunkan untuk semua masukan. Agar komputasi menjadi lebih efisien, turunan fungsi aktivasi juga harus bisa dinyatakan oleh fungsi aktivasi itu sendiri.
- Nilai keluaran yang diharapkan biasanya adalah antara 1 sampai 0. Oleh karena itu fungsi aktivasi harus memiliki *range* antara 1 sampai 0.

Fungsi yang memenuhi kedua syarat tersebut adalah fungsi sigmoid biner dan bipolar. Dalam penelitian ini digunakan fungsi sigmoid biner untuk fungsi aktivasi.

Fungsi sigmoid biner dapat dilihat pada persamaan 3.10. Turunan fungsi sigmoid biner dapat dilihat pada persamaan 3.11

$$f(x) = \frac{1}{1+\exp(-x)} \tag{3.10}$$

$$f'(x) = f(x)\left|1 - f(x)\right| \tag{3.11}$$

3.3.5 Inisialisasi bobot awal

Ketika akan mulai dilatih, BPNN memiliki bobot yang tidak bernilai sama sekali. Hal ini tidak baik karena nilai awal bobot menentukan kecepatan konvergensi BPNN dan apakah BPNN akan mencapai nilai minimum global atau minimum lokal kesalahan.

Untuk itulah diperlukan inisialisasi bobot awal. Pada umumnya bobot diinisialisasi dengan nilai acak. Tetapi dengan metode tersebut hasilnya tidak terlalu bagus maka

pada penelitian ini digunakan metode lain yang merupakan modifikasi dari metode tersebut yaitu inisialisasi bobot Nguyen-Widrow.

Langkah-langkah inisialisasi bobot Nguyen-Widrow adalah sebagai berikut

1. Untuk bias dan bobot V dan W lakukan inisialisasi acak dengan jangkauan -0.5 sampai 0.5.
2. Untuk bias dan bobot V lakukan
   c. Hitung faktor skala beta dengan persamaan 3.12.

$$\beta = 0.7(P)^{\frac{1}{N}} \tag{3.12}$$

   β adalah faktor skala yang akan digunakan untuk inisialisasi bobot.

   P adalah besarnya lapisan tersembunyi

   N adalah besarnya lapisan masukan

   d. Inisialisasi nilai bobot V dengan bilangan acak antara -0.5 sampai 0.5.

   e. Hitung norma v_j

$$\|v_j\| = \sum_i v_j \tag{3.13}$$

   f. Inisialisasi ulang v_{ij}

$$v_{ij} = \frac{\beta v_{ij}}{\|v_j\|} \tag{3.14}$$

   g. Inisialisasi bias v_{0j} dengan bilangan acak antara $-\beta$ dan β

3.3.6 Fungsi Kesalahan Propagasi

Fungsi kesalahan propagasi digunakan untuk menghitung nilai kesalahan sistem secara keseluruhan. Nilai kesalahan didapatkan dengan membandingkan vektor keluaran dan vektor target untuk kelas yang bersesuaian pada tahap pelatihan. Fungsi ini dihitung untuk setiap vektor masukan dalam satu epoch, setelah itu semuanya dijumlahkan untuk menghitung nilai kesalahan pada satu epoch. Jika BPNN berjalan dengan baik, nilai kesalahan tersebut akan konvergen mengecil.

Ketika nilai kesalahan telah mengecil sampai batas bawah yang ditentukan tahap pelatihan berakhir. Nilai kesalahan yang kecil berarti BPNN sudah siap untuk mengenali objek dari data pengujian. Akan tetapi nilai kesalahan yang kecil bukan berarti tingkat pengenalan juga akan tinggi.

Ada beberapa fungsi yang dapat digunakan sebagai fungsi kesalahan propagasi. Diantaranya adalah fungsi kesalahan kuadratik, *cross entropy*, dan metrik. Dalam penelitian ini digunakan fungsi *cross entropy*. Fungsi tersebut dipilih karena perhitungannya yang paling akurat dan juga dapat mempercepat konvergensi nilai kesalahan.

Fungsi *cross entropy* didefinisikan pada persamaan 3.15.

$$\sum_k [(-t_k)\ln(y_k)] - [(1-t_k)\ln(1-y_k)] \qquad (3.15)$$

Fungsi ini digunakan untuk menghitung nilai kesalahan untuk satu vektor masukan. *k* adalah banyaknya kelas pada BPNN. Hasil dari vektor masukan ini akan dijumlahkan lagi dengan hasil dari vektor masukan lainnya untuk menghasilkan nilai kesalahan epoch.

3.3.7 Algoritma pelatihan

1. Inisialisasi bobot *V* dan *W*, alpha, beta, epoch maksimal dan batasan tingkat kesalahan minimal.
2. Selama batasan tingkat kesalahan minimal dan epoch maksimal belum tercapai lakukan langkah 3 dan 4
3. Tahap *feed forward*
   a. Setiap neuron pada lapisan masukan ($X_i, i = 1,2,\ldots,n$) menerima vektor masukan x_i lalu diteruskan kepada lapisan tersembunyi *Z*.
   b. Setiap neuron pada lapisan tersembunyi ($Z_j, j = 1,2,\ldots,p$).
      - Menghitung nilai masukan berdasarkan bobot *V* dengan persamaan 3.16.

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (3.16)$$

- Menghitung nilai aktivasinya masing-masing dengan persamaan 3.17.

$$z_j = f(z\_in_j) \quad (3.17)$$

- Meneruskan nilai aktivasi kepada lapisan keluaran *Y*.

c. Setiap neuron pada lapisan keluaran ($Y_k, k = 1,2,\ldots,m$).

- Menghitung nilai masukan berdasarkan bobot *W* dengan persamaan 3.18.

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad (3.18)$$

- Menghitung nilai aktivasinya masing-masing dengan persamaan 3.19.

$$y_k = f(y\_in_k) \quad (3.19)$$

4. Tahap *backpropagation*

a. Setiap neuron pada lapisan keluaran ($Y_k, k = 1,2,\ldots,m$)

- Menerima vektor target *t* yang bersesuaian dengan vektor masukan *x*.
- Menghitung nilai kesalahan dengan persamaan 3.20.

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (3.20)$$

- Menghitung nilai koreksi bobot *W* untuk lapisan keluaran *Y* dengan persamaan 3.21.

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (3.21)$$

- Menghitung nilai koreksi bias untuk lapisan keluaran *Y* dengan persamaan 3.22.

$$\Delta w_{0k} = \alpha \delta_k \quad (3.22)$$

- Meneruskan nilai kesalahan kepada lapisan tersembunyi *Z*.

b. Setiap neuron pada lapisan tersembunyi ($Z_j, j = 1,2,\ldots,p$).

- Menghitung nilai kesalahan dengan persamaan 3.23.

$$\delta_j = (\sum_{k=1}^{m} \delta_k w_{jk}) f'(z\_in_j) \quad (3.23)$$

- Menghitung nilai koreksi bobot V untuk lapisan tersembunyi dengan persamaan 3.24.

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (3.24)$$

- Menghitung nilai koreksi bias untuk lapisan tersembunyi dengan persamaan 3.25.

$$\Delta v_{0j} = \alpha \delta_j \quad (3.25)$$

c. Setiap neuron pada lapisan keluaran ($Y_k, k = 1,2,\ldots,m$) mengubah nilai bobot dan biasnya ($j = 1,2,\ldots,p$) dengan persamaan 3.26 dan 3.27.

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (3.26)$$

$$w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k} \quad (3.27)$$

d. Setiap neuron pada lapisan tersembunyi ($Z_j, j = 1,2,\ldots,p$) mengubah nilai bobot dan biasnya ($i = 1,2,\ldots,n$) dengan persaman 3.28 dan 3.29.

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (3.28)$$

$$v_{0j}(baru) = v_{0j}(lama) + \Delta v_{0j} \quad (3.29)$$

5. Jika batasan tingkat kesalahan minimal dan epoch maksimal belum tercapai ulangi langkah 2, 3 dan 4.

Nilai bobot *V* dan *W* disimpan untuk digunakan dalam proses pengujian.

3.1.8 Algoritma Pengujian

1. Bobot *V* dan *W* telah terinisialisasi dengan hasil yang didapat setelah pelatihan.
2. Setiap neuron pada lapisan masukan ($X_i, i = 1,2,\ldots,n$) menerima vektor masukan x_i lalu diteruskan kepada lapisan tersembunyi *Z*.
3. Setiap neuron pada lapisan tersembunyi ($Z_j, j = 1,2,\ldots,p$).
   - Menghitung nilai masukan berdasarkan bobot *V* dengan persamaan 3.16.

- Menghitung nilai aktivasinya masing-masing dengan persamaan 3.17.
- Meneruskan nilai aktivasi kepada lapisan keluaran *Y*.

4. Setiap neuron pada lapisan keluaran ($Y_k, k = 1,2,\ldots,m$).
   - Menghitung nilai masukan berdasarkan bobot *W* dengan persamaan 3.18.
   - Menghitung nilai aktivasinya masing-masing dengan persamaan 3.19. Nilai ini merupakan nilai keluaran jaringan secara keseluruhan.

Hasil keluaran setelah pengujian diperiksa untuk menentukan apakah suatu masukan dikenali atau tidak. Hasil keluaran ($Y_k, k = 1,2,\ldots,m$) berupa nilai-nilai pengenalan untuk semua *m* kelas. Kriteria suatu masukan dikenali adalah jika nilai pengenalan pada kelas masukan tersebut bernilai lebih dari 0.75 dan untuk kelas lainnya tersebut bernilai kurang dari 0.25. Jika suatu masukan tidak memenuhi kriteria tersebut pada pengujian berarti masukan tersebut tidak dikenali.

Proses inisialisasi nguyen-widrow yang digunakan dalam BPNN melibatkan angka acak. Oleh karena itu pengenalan dengan BPNN dilakukan sebanyak lima kali untuk setiap jenis vektor masukan.