PERBANDINGAN KLASIFIKASI MENGGUNAKAN METODE *BACKPROPAGATION* DAN METODE *LEARNING VECTOR QUANTIZATION*
(Studi Kasus Status Gizi Balita Kabupaten/Kota di Indonesia 2016)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Statistika

Oleh:
AMALIA RAHMAWATI
155090501111007

PROGRAM STUDI SARJANA STATISTIKA
JURUSAN STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2019

LEMBAR PENGESAHAN SKRIPSI

PERBANDINGAN KLASIFIKASI MENGGUNAKAN METODE *BACKPROPAGATION* DAN METODE *LEARNING VECTOR QUANTIZATION* (Studi Kasus Status Gizi Balita Kabupaten/Kota di Indonesia 2016)

Oleh:
AMALIA RAHMAWATI
155090501111007

Setelah dipertahankan di depan Majelis Penguji Pada Tanggal 5 Maret 2019 dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Statistika

Dosen Pembimbing

Dr. Dra. Umu Sa'adah, M.Si
NIP. 196807252002122001

Mengetahui,
Ketua Jurusan Statistika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Brawijaya

Rahma Fitriani, S.Si., M.Sc, Ph.D
NIP. 197603281999032001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama	**: Amalia Rahmawati**
NIM	**: 155090501111007**
Jurusan/Program Studi	**: Statistika/Statistika**
Menuliskan skripsi yang berjudul	**:**

PERBANDINGAN KLASIFIKASI MENGGUNAKAN METODE *BACKPROPAGATION* DAN METODE *LEARNING VECTOR QUANTIZATION* (Studi Kasus Status Gizi Balita Kabupaten/Kota di Indonesia 2016)

Dengan ini menyatakan bahwa:

1. **Isi dari skripsi yang saya buat adalah benar-benar karya saya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, Maret 2019
Yang menyatakan,

Amalia Rahmawati
155090501111007

PERBANDINGAN KLASIFIKASI MENGGUNAKAN METODE *BACKPROPAGATION* DAN METODE *LEARNING VECTOR QUANTIZATION* (Studi Kasus Status Gizi Balita Kabupaten/Kota di Indonesia 2016)

ABSTRAK

Klasifikasi merupakan proses menemukan sekumpulan model yang dapat digunakan untuk menggambarkan dan membedakan konsep atau kelas-kelas data yang kelasnya tidak diketahui sebelumnya. Salah satu metode klasifikasi yang berkembang dari kelompok *machine learning* di bidang *artificial intelligence* adalah jaringan syaraf tiruan. Penelitian ini bertujuan untuk melakukan perbandingan ketepatan hasil klasifikasi metode *backpropagation* dan metode *learning vector quantization* terhadap data status gizi balita kabupaten/kota di Indonesia pada tahun 2016. Ketepatan klasifikasi dapat ditentukan melalui *hit ratio*, semakin besar *hit ratio* menunjukkan bahwa hasil klasifikasi semakin baik. Dengan menggunakan 80% data *training* dan 20% data *testing* maka diperoleh *hit ratio* metode *backpropagation* dengan enam *neuron hidden layer* sebesar 77,88% dan metode *learning vector quantization* dengan *size codebook* 9 sebesar 80,77%. Secara keseluruhan dapat disimpulkan bahwa ketepatan klasifikasi terbaik pada data status gizi balita kabupaten/kota di Indonesia tahun 2016 diperoleh menggunakan metode *learning vector quantization* dengan *size codebook* 9.

Kata kunci: Klasifikasi, Status gizi balita, *Backpropagation, Learning vector quantization, Hit ratio*.

COMPARISON OF CLASSIFICATION USING THE BACKPROPAGATION METHOD AND THE LEARNING VECTOR QUANTIZATION METHOD (Case Study of District/City Toddler Nutrition Status in Indonesia 2016)

ABSTRACT

Classification is the process of finding a set of models that can be used to describe and define concepts or classes of data whose classes are not understood beforehand. One classification method that developed from the machine learning group in the field of artificial intelligence is artificial neural networks. This study aims to compare the accuracy of the results of the classification of the backpropagation method and the learning vector quantization method to the 2016 nutritional data of district / city toddlers in Indonesia. The classification accuracy can be determined through the hit ratio, the greater the hit ratio shows that the classification results are better. By using 80% training data and 20% testing data the hit ratio backpropagation method with siz neuron hidden layers is 77.88% and the learning vector quantization method with size codebook 9 is 80.77%. Overall it can be concluded that the accuracy of the best classification on the nutritional status data of district/city toddlers in Indonesia in 2016 was obtained using the learning vector quantization method with size codebook 9.

Keywords: Classification, Nutritional status of children under five years, Backpropagation, Learning vector quantization, Hit ratio.

KATA PENGANTAR

Segala Puji dan syukur atas kehadirat Allah SWT yang telah memberi anugerah dan rahmat kepada penulis untuk menyelesaikan skripsi ini. Semoga Allah SWT membalas kebaikan semua pihak yang telah membantu dan mempermudah hingga skripsi ini selesai. Penulis mengucapkan terima kasih kepada:

1. Dr. Dra. Umu Sa'adah, M.Si selaku dosen pembimbing yang telah berkenan memberikan tambahan ilmu serta arahan selama penyusunan skripsi.
2. Darmanto, S.Si., M.Si selaku dosen penguji satu yang telah berkenan memberikan tambahan ilmu serta arahan selama penyusunan skripsi.
3. Samingun H. S.Si., M.Cs selaku dosen penguji dua yang telah berkenan memberikan tambahan ilmu serta arahan selama penyusunan skripsi.
4. Rahma Fitriani, S.Si., M.Sc., Ph.D, selaku Ketua Jurusan Statistika, Universitas Brawijaya, Malang.
5. Kedua orang tua dan kakak yang sangat saya hormati dan cintai. Terima kasih atas segala limpahan kasih sayang, perhatian, kekuatan, doa, dan dukungan yang diberikan.
6. Fahrizal Yazid Ilham yang telah memberikan semangat, bantuan, serta motivasi selama proses penyusunan skripsi.
7. Sahabat-sahabat saya Jeje, Dana, Sasgia, Rizki, Dini, dan Icha. Terimakasih atas semangat dan kebersamaan yang tidak terlupakan.
8. Semua kerabat dan pihak yang telah membantu dan tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam skripsi ini masih banyak kekurangan dan jauh dari sempurna, sehingga saran dan kritik akan penulis harapkan demi perbaikan penyusunan karya selanjutnya.

Malang, Maret 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Klasifikasi merupakan proses menemukan sekumpulan model atau fungsi yang menggambarkan dan membedakan konsep atau kelas-kelas data, dengan tujuan agar model tersebut dapat digunakan untuk prediksi kelas dari suatu objek atau data yang label kelasnya tidak diketahui (Han, 2001). Klasifikasi dapat digunakan sebagai salah satu cara untuk mengetahui kelompok dari data yang mempunyai kemiripan pada suatu kriteria tertentu.

Status gizi merupakan keadaan tubuh manusia yang berhubungan dengan konsumsi makanan serta dipengaruhi baik oleh faktor internal maupun eksternal, seperti usia, jenis kelamin, aktivitas fisik, penyakit, serta keadaan sosial ekonomi (Wolley, dkk., 2016). Status gizi pada balita dapat dilihat melalui tiga indeks, yaitu berat badan menurut umur (BB/U), tinggi badan menurut umur (TB/U), dan berat badan menurut tinggi badan (BB/TB). Tujuan kebijakan pemerintah adalah melakukan peningkatan terhadap kesejahteraan masyarakat salah satunya melalui peningkatan kesehatan. Contoh upaya peningkatan derajat kesehatan adalah perbaikan gizi masyarakat, karena gizi yang seimbang dapat meningkatkan ketahanan tubuh. Untuk memudahkan pemerintah dalam membentuk kebijakan, maka dapat dilakukan klasifikasi tingkat status gizi di setiap kabupaten/kota di Indonesia sehingga penentuan arah kebijakan perbaikan gizi masyarakat lebih efektif, efisien, dan tepat sasaran.

Salah satu metode klasifikasi yang berkembang dari kelompok *machine learning* di bidang *artificial intelligence* adalah jaringan syaraf tiruan. Jaringan syaraf tiruan didefinisikan sebagai susunan dari elemen-elemen penghitung yang disebut *neuron* (Puspitaningrum, 2006). Jaringan syaraf tiruan mempunyai kelebihan, yaitu tidak harus memenuhi asumsi tertentu akan tetapi metode jaringan syaraf tiruan memiliki kelemahan, yaitu membutuhkan waktu pelatihan yang lebih lama untuk melakukan perhitungan dalam pembentukan model dibandingkan dengan metode klasifikasi yang lainnya.

Jaringan syaraf tiruan mempunyai tingkat kemampuan dan koordinasi yang sangat baik antar jaringan sehingga dapat digunakan untuk klasifikasi (Jhon, 2004). Metode jaringan syaraf tiruan yang sering digunakan untuk klasifikasi yaitu metode *backpropagation*, *radial basis function* (RBF), *learning vector quantization* (LVQ),

perceptron, dan lain-lain. Berdasarkan penelitian terdahulu oleh Ramli dan Goenjantoro (2013) mengenai perbandingan metode klasifikasi regresi logistic biner dengan metode *backpropagation* didapatkan kesimpulan bahwa metode *backpropagation* lebih baik digunakan karena menghasilkan ketepatan klasifikasi sebesar 80,89%. Penelitian oleh Ulfasari (2010) mengenai perbandingan performansi jaringan *learning vector quantization* (LVQ) dan *radial basis function* (RBF) untuk permasalahan klasifikasi penyakit karies gigi menghasilkan kesimpulan bahwa jaringan RBF memberikan tingkat akurasi yang lebih tinggi daripada jaringan LVQ. Selain itu penelitian oleh Ramadhani dan Anis (2015) mengenai klasifikasi penyakit kencing manis (*diabetes mellitus*) menggunakan jaringan syaraf tiruan *backpropagation*, menghasilkan kesimpulan konfigurasi parameter terbaik dengan menggunakan *epoch* maksimum 1000, *learning rate* 0,15, *neuron hidden layer* 50, dan *error* 0,0001 menghasilkan tingkat akurasi sebesar 99%.

Penelitian ini bertujuan untuk melakukan perbandingan ketepatan hasil klasifikasi yang dihasilkan oleh metode *backpropagation* dan metode *learning vector quantization* (LVQ) terhadap data status gizi balita di Indonesia pada tahun 2016. Klasifikasi diharapkan dapat memberikan kontribusi terhadap tercapainya peningkatan derajat kesehatan masyarakat dan untuk memudahkan penyusunan perencanaan dan perumusan kebijakan gizi oleh pemerintah. Metode *backpropagation* mempunyai kelebihan yaitu meminimalkan *error* pada *output* yang dihasilkan, sedangkan metode *learning vector quantization* (LVQ) mempunyai kelebihan dapat meringkas data set yang besar menjadi vektor *codebook* berukuran kecil untuk klasifikasi serta model yang dihasilkan menggunakan metode ini dapat diperbaharui. Metode yang dinyatakan terbaik untuk klasifikasi adalah metode yang memiliki *misclassified* lebih kecil. *Misclassified* dapat diketahui dari hasil ketepatan prediksi masing-masing metode yang dibandingkan dengan data aktualnya.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini antara lain:

1. Bagaimana perbandingan klasifikasi yang dihasilkan menggunakan metode *backpropagation* dan metode *learning vector quantization* (LVQ) pada data status gizi balita di Indonesia tahun 2016?

2. Bagaimana perbandingan keakuratan antara hasil klasifikasi menggunakan metode *backpropagation* dan metode *learning vector quantization* (LVQ) pada data status gizi balita di Indonesia tahun 2016?
3. Metode mana yang lebih baik digunakan untuk klasifikasi antara metode *backpropagation* dan metode *learning vector quantization* (LVQ) pada data status gizi balita di Indonesia tahun 2016?

1.3 Tujuan

Tujuan dari penelitian ini antara lain:

1. Membandingkan hasil klasifikasi menggunakan metode *backpropagation* dan metode *learning vector quantization* (LVQ) pada status gizi balita di Indonesia tahun 2016.
2. Membandingkan keakuratan hasil klasifikasi antara metode *backpropagation* dan metode *learning vector quantization* (LVQ) pada data status gizi balita di Indonesia tahun 2016.
3. Menentukan metode yang lebih baik digunakan untuk klasifikasi antara metode metode *backpropagation* dan metode *learning vector quantization* (LVQ) pada data status gizi balita di Indonesia tahun 2016.

1.4 Manfaat

Manfaat dari penelitian ini antara lain:

1. Menambah wawasan serta pengetahuan mengenai penerapan metode *backpropagation* dan metode *learning vector quantization* (LVQ) pada data status gizi balita di Indonesia tahun 2016.
2. Memperoleh informasi mengenai klasifikasi pada data status gizi balita di Indonesia tahun 2016.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini antara lain:

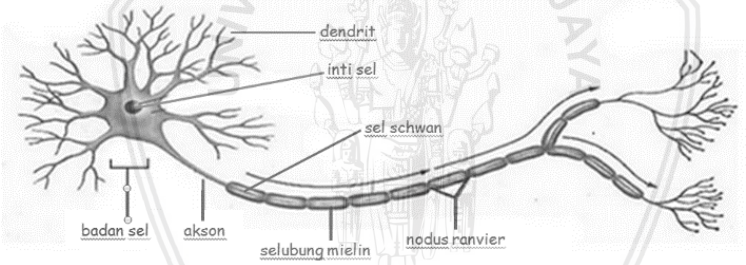
1. Data yang digunakan pada penelitian ini adalah data mengenai status gizi balita 514 kabupaten/kota di Indonesia tahun 2016.
2. Penelitian ini menggunakan persentase pembagian 80% data *training* dan 20% data *testing*.

BAB II
TINJAUAN PUSTAKA

2.1 Jaringan Syaraf Tiruan / *Neural Network*

Jaringan syaraf tiruan (JST) atau *artificial neural network* (ANN) dibuat pertama kali pada tahun 1943 oleh Waren McCulloh dan Walter Pits. Waren McCulloh dan Walter Pits menyimpulkan bahwa kombinasi beberapa *neuron* sederhana menjadi suatu sistem syaraf memiliki kemampuan untuk menyelesaikan suatu sistem yang kompleks. Bobot pada jaringan syaraf tiruan akan dimanfaatkan untuk menyelesaikan persamaan fungsi logika sederhana (Siang, 2005).

Jaringan syaraf tiruan (JST) merupakan kecerdasan buatan yang memiliki karakteristik mirip dengan jaringan syaraf biologis. JST dibangun dengan prinsip dasar perambatan sinyal-sinyal *neuron* yang dilatih terus menerus dengan algoritma tertentu sehingga dapat mengenal pola pelatihan jika diberikan data masukan baru (Kusumadewi, 2004).



Gambar 2.1.Contoh Sel Syaraf Biologis

Jaringan syaraf biologis merupakan kumpulan dari sel-sel syaraf (*neuron*). *Neuron* mempunyai tugas mengolah informasi. Komponen-komponen utama dari sebuah *neuron* dapat dikelompokkan menjadi tiga bagian (Ramli dan Goejantoro, 2013), antara lain:

1. *Dendrite*, bertugas untuk menerima informasi.
2. Badan sel (soma), bertugas sebagai tempat pengolahan informasi.
3. Akson (*neurit*), bertugas mengirimkan impuls-impuls ke sel syaraf lainnya.

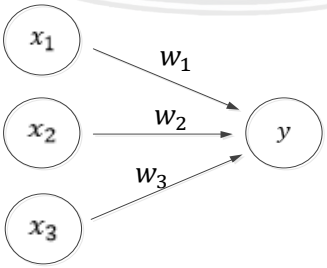
Jaringan syaraf tiruan merupakan suatu sistem pemrosesan informasi yang mampu mengenali kegiatan dengan berbasis pada pola data. Data masukan akan dipelajari oleh jaringan syaraf tiruan sehingga keputusan diberikan. Jaringan syaraf tiruan dibentuk sebagai generalisasi model matematis dari jaringan syaraf manusia dengan didasarkan pada asumsi-asumsi berikut (Hermawan, 2006):

1. Pemrosesan informasi terjadi pada elemen sederhana (*neuron*).
2. Sinyal dikirim antar *neuron-neuron* melalui penghubung yang disebut dengan sinapsis.
3. Setiap sinapsis memiliki bobot.
4. Setiap *neuron* menggunakan fungsi aktivasi (biasanya bukan fungsi linear) untuk menentukan *output*.

Hal yang ingin dicapai dengan jaringan syaraf tiruan adalah untuk mencapai keseimbangan antara kemampuan memorisasi dan generalisasi. Yang dimaksud kemampuan memorisasi adalah kemampuan jaringan syaraf tiruan untuk mengambil kembali secara sempurna sebuah pola yang telah dipelajari. Kemampuan generalisasi adalah kemampuan jaringan syaraf tiruan untuk menghasilkan respon yang bisa diterima terhadap pola-pola yang sebelumnya telah dipelajari. Jaringan syaraf tiruan ditentukan oleh tiga hal (Siang, 2005):

1. Pola hubungan antar *neuron* (arsitektur jaringan).
2. Metode untuk menentukan bobot penghubung (metode *training/learning*).
3. Fungsi aktivasi, yaitu fungsi yang digunakan untuk menentukan keluaran suatu *neuron*.

Sebagai contoh, perhatikan *neuron y* pada Gambar 2.2 di bawah ini.

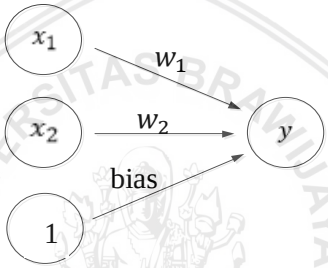


Gambar 2.2. Arsitektur Jaringan Syaraf Tiruan

Neuron y menerima *input* dari *neuron* x_1, x_2 dan x_3 dengan bobot hubungan masing-masing adalah w_1, w_2, dan w_3. Ketiga impuls *neuron* yang ada dijumlahkan (Siang, 2005).

$$y\_net = x_1 w_1 + x_2 w_2 + x_3 w_3 \quad (2.1)$$

Kadang-kadang dalam jaringan ditambahkan sebuah *neuron* masukan yang nilainya selalu sama dengan satu. *Neuron* yang demikian disebut dengan bias. Bias dapat dipandang sebagai sebuah *input* yang nilainya selalu sama dengan satu. Bias berfungsi untuk mengubah nilai *threshold* menjadi nol. Jika melibatkan bias, maka keluaran *neuron* penjumlahan terdapat pada persamaan 2.2. (Siang, 2005)



Gambar 2.3. Arsitektur Jaringan Syaraf Tiruan Dengan Bias

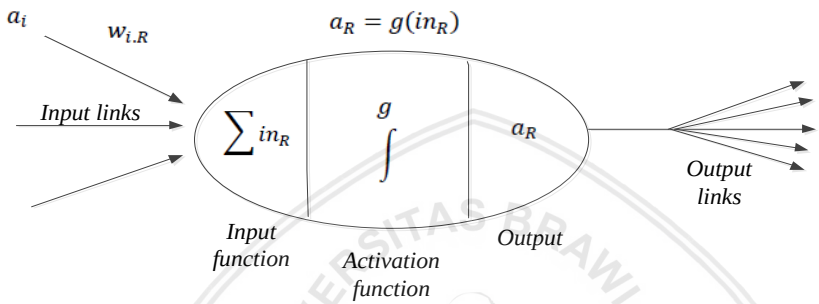
$$y\_net = b + \sum_i x_i w_i \quad (2.2)$$

2.1.1 Konsep Dasar Jaringan Syaraf Tiruan

Jaringan syaraf tiruan merupakan suatu sistem yang proses kerjanya diilhami dari aktivitas jaringan syaraf pada manusia. Jaringan ini terdiri dari sekumpulan *neuron-neuron* yang saling berinteraksi. Secara umum suatu jaringan syaraf tiruan dibentuk atas sejumlah *neuron* sebagai pengolah informasi sebagai dasar operasi untuk menjalankan fungsi dan tugasnya (Siang, 2005). Pada Gambar 2.3 sejumlah sinyal masukan a dikalikan dengan masing-masing penimbang yang bersesuaian w kemudian dilakukan penjumlahan dari seluruh hasil perkalian tersebut dan keluaran yang dihasilkan dilanjutkan ke dalam fungsi pengaktif untuk mendapatkan tingkat derajat sinyal keluaran $f(a, w)$. Dengan w merupakan bobot penghubung antar *neuron* pada jaringan syaraf tiruan, biasanya bobot

awal diinisialisasikan secara random dengan nilai antara -0,5 sampai 0,5 atau -1 sampai 1 (Fausett, 1994).

Sedangkan α atau *learning rate* merupakan nilai yang menentukan kecepatan dari proses pelatihan JST, semakin besar nilai α maka semakin cepat pula proses pelatihan. Akan tetapi jika α terlalu besar, maka algoritma menjadi tidak stabil. Untuk penyederhanaan biasanya α diberi nilai kecil yaitu 0,1 (Siang, 2005). Model tiruan *neuron* pada jaringan syaraf tiruan terdapat pada Gambar 2.4.



Gambar 2.4. Model Tiruan *Neuron* (Suhartono, 2007)

Keterangan:

a_i = Nilai aktifasi dari *neuron i*
$w_{i.R}$ = Bobot dari *neuron I* ke *neuron R*
in_R = Penjumlahan bobot dan masukan ke *neuron R*
g = Fungsi aktivasi
a_R = Nilai aktivasi dari *neuron R*

Misalkan ada *n* buah sinyal masukan dan *n* buah penimbang, fungsi keluaran dari *neuron* adalah seperti pada persamaan (2.3) di bawah ini

$$in_R = \sum_i w_{iR} * a_i \qquad (2.3)$$

Kumpulan dari *neuron* dibuat menjadi sebuah jaringan yang akan berfungsi sebagai alat komputasi. Banyaknya *neuron* dan struktur jaringan untuk setiap masalah yang akan diselesaikan adalah berbeda (Suhartono, 2007).

2.1.2 Model *Neuron*

Pada satu sel syaraf terdiri dari tiga bagian, yaitu fungsi penjumlahan (*summing function*), fungsi aktivasi (*activation function*), dan fungsi masukan (*input function*). Informasi (*input*) akan dikirim ke *neuron* dengan bobot tertentu. *Input* ini akan diproses oleh suatu fungsi yang akan menjumlahkan nilai-nilai bobot yang ada. Hasil penjumlahan kemudian akan dibandingkan dengan suatu nilai ambang (*threshold*) tertentu melalui fungsi aktivasi setiap *neuron*. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka *neuron* tersebut akan diaktifkan, jika tidak, maka *neuron* tersebut akan mengirimkan *output* melalui bobot-bobot *output*nya ke semua *neuron* yang berhubungan dengannya. Menurut Siang (2005) *neuron* terdiri dari tiga elemen pembentuk antara lain:

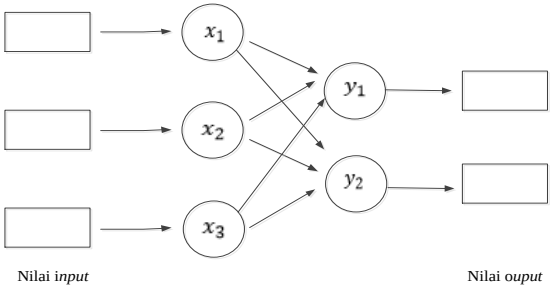
1. Himpunan *neuron-neuron* yang dihubungkan dengan jalur koneksi. Jalur-jalur tersebut memiliki bobot yang berbeda-beda. Bobot yang bernilai positif akan memperkuat sinyal dan yang bernilai negatif akan memperlemah sinyal yang dibawa. Jumlah, struktur, dan pola hubungan antar *neuron* tersebut akan menentukan arsitektur jaringan.
2. Suatu *neuron* penjumlahan yang akan menjumlahkan *input-input* sinyal yang sudah dikalikan dengan bobotnya.
3. Fungsi aktivasi yang akan menentukan apakah sinyal dari *input neuron* akan diteruskan ke *neuron* lain atau tidak.

2.1.3 Arsitektur Jaringan Syaraf Tiruan

Jaringan syaraf tiruan memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai aplikasi. Arsitektur jaringan syaraf tiruan tersebut, antara lain (Kusumadewi, 2004):

4.1 Jaringan Lapisan Tunggal (*Single Layer Network*)

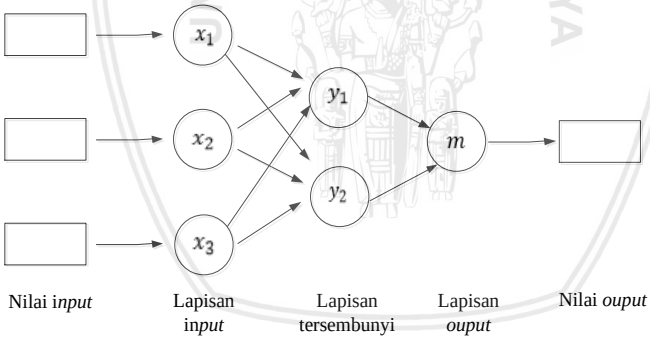
Jaringan dengan lapisan tunggal terdiri dari satu *input layer* dan satu *output layer*. Setiap *neuron* yang terdapat di dalam *input layer* selalu terhubung dengan setiap *neuron* yang terdapat pada *output layer*. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi. Contoh algoritma jaringan syaraf tiruan yang menggunakan metode ini yaitu *ADALINE*, *hopfield*, dan *perceptron*.



Gambar 2.5. Arsitektur Lapisan Tunggal (Hermawan, 2006)

4.2 Jaringan Lapisan Jamak (*Multi Layer Network*)

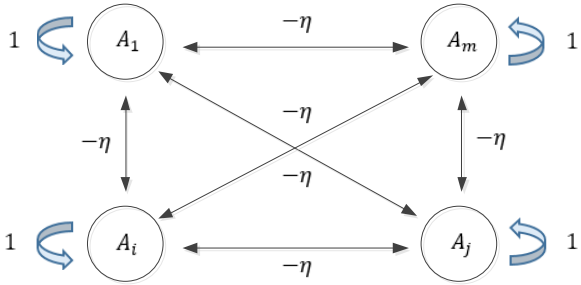
Jaringan dengan lapisan jamak memiliki tiga jenis lapisan/ *layer* sebagai ciri khas yaitu *input layer, output layer*, layer tersembunyi. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang kompleks dibandingkan jaringan dengan lapisan tunggal. Namun proses pelatihan sering membutuhkan waktu yang cenderung lama. Contoh algoritma jaringan syaraf tiruan yang menggunakan metode ini yaitu MADALINE, *backpropagation,*dan *neocognitron.*



Gambar 2.6. Arsitektur Lapisan Jamak (Hermawan, 2006)

4.3 Jaringan Lapisan Kompetitif (*Competitive Layer Network*)

Pada jaringan kompetitif, sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif. Contoh algoritma jaringan syaraf tiruan yang menggunakan metode ini yaitu *learning vector quantization* (LVQ).



Gambar 2.7. Arsitektur Lapisan Kompetitif (Hermawan, 2006)

2.1.4 Metode Pelatihan/Pembelajaran Jaringan Syaraf Tiruan

Menurut Puspitaningrum (2006), cara berlangsungnya pembelajaran atau pelatihan jaringan syaraf tiruan dapat dikelompokkan menjadi tiga, antara lain:

1. *Supervised Learning* (Pembelajaran Terawasi)

Pada metode ini, setiap pola yang diberikan kedalam jaringan syaraf tiruan telah diketahui target jaringan. Selisih antara pola *output* aktual (*output* yang dihasilkan) dengan pola *output* yang dikehendaki (*output* target) yang disebut *error* digunakan untuk mengoreksi bobot jaringan syaraf tiruan sehingga jaringan syaraf tiruan mampu menghasilkan output sedekat mungkin dengan pola target yang telah diketahui oleh jaringan syaraf tiruan.

2. *Unsupervised Learning* (Pembelajaran Tidak Terawasi)

Pada metode ini tidak memerlukan targe jaringan. Tidak dapat ditentukan hasil yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu *range* tertentu tergantung pada nilai *input* yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan *neuron-neuron* yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk klasifikasi pola.

3. *Hybrid Learning* (Pembelajaran Hibrid)

Merupakan kombinasi dari metode pembelajaran *supervised learning* dan *unsupervised learning*. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian melalui pembelajaran tak terawasi. Metode ini baik dan sesuai dalam melakukan pengenalan pola-pola pada gambar.

2.1.5 Fungsi Aktivasi Jaringan Syaraf Tiruan

Menurut Patterson (1996) dalam jaringan syaraf tiruan, fungsi aktivasi digunakan untuk menentukan keluaran suatu *neuron*. Permasalahan yang sering muncul adalah menentukan parameter atau pembobot serta mengoptimalkan banyaknya lapisan dan *neuron* suatu arsitektur syaraf tiruan.

Penggunaan fungsi aktivasi pada jaringan syaraf tiruan bergantung pada permasalahan dan teori yang mendukung. Misalnya fungsi aktivasi yang cocok untuk metode *backpropagation* adalah fungsi aktivasi *sigmoid biner* karena fungsi aktivasi ini memenuhi beberapa sifat yaitu kontinu, terdiferensial dengan mudah dan merupakan fungsi aktivasi yang tidak turun (Siang, 2005). Beberapa fungsi aktivasi yang digunakan dalam JST, antara lain:

1. Linear

Fungsi linear memiliki nilai *output* sama dengan nilai *input*nya, dirumuskan sesuai persamaan 2.4:

$$f(x) = mx + b \tag{2.4}$$

Keterangan:

$f(x)$ = Fungsi aktivasi

m, b = Konstanta

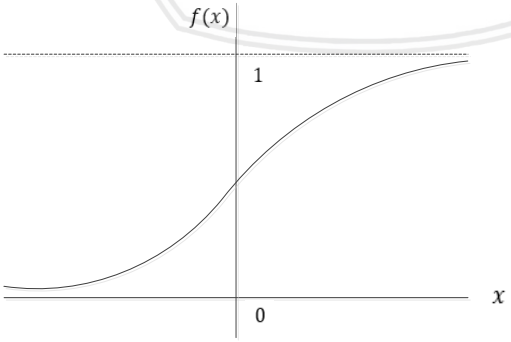
2. *Logistic Sigmoid Biner*

Fungsi *logistic sigmoid biner* sering digunakan oleh jaringan yang membutuhkan *output* pada interval 0 sampai 1 atau 0 dan 1. Persamaan fungsi *logistic sigmoid biner* terdapat pada persamaan 2.5.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Dengan turunan:

$$f'(x) = f(x)(1 - f(x)) \tag{2.6}$$

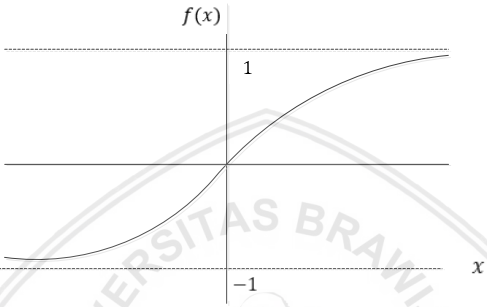


Gambar 2.8. Fungsi *Sigmoid Biner* (Fausett, 1994)

3. *Logistic Sigmoid Bipolar*

Fungsi *logistic sigmoid bipolar* hampir sama dengan fungsi *sigmoid biner*. Perbedaannya ada pada *output* yang memiliki interval -1 sampai 1. Persamaan fungsi *logistic sigmoid bipolar* dituliskan pada persamaan 2.7

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} = \frac{2}{1 + e^{-x}} = -1 \qquad (2.7)$$



Gambar 2.9. Fungsi *Sigmoid Bipolar* (Fausett, 1994)

2.1.6 Jaringan Syaraf Tiruan Untuk Klasifikasi

Salah satu kegunaan pendekatan jaringan syaraf tiruan adalah untuk mengelompokkan atau klasifikasi. Menurut Fausett (1994), dasar pengklasifikasian pada jaringan syaraf tiruan adalah menggunakan bobot. Bobot adalah nilai matematis dari koneksi antar *neuron* yang mentransfer data dari satu lapisan ke lapisan lainnya dan berfungsi untuk mengatur jaringan sehingga dapat menghasilkan *output* yang diinginkan (Hidayati, 2010).

Menurut Jhon (2004), karena mempunyai tingkat kemampuan dan koordinasi yang sangat baik antar jaringan, maka jaringan syaraf tiruan tepat untuk diterapkan untuk:

1. Klasifikasi, memilih suatu input data kedalam suatu kategori tertentu yang diterapkan.
2. Asosiasi, mengambarkan suatu objek secara keseluruhan hanya dengan sebuah bagian dari objek lain.
3. *Self organizing*, kemampuan untuk mengelola data-data input tanpa harus memiliki data sebagai target.
4. Optimasi, menemukan suatu jawaban atau solusi yang paling baik sehingga seringkali dengan meminimalisasikan suatu fungsi biaya (*optimizer*).

2.2 *Backpropagation*

Backpropagation merupakan sebuah jaringan *multilayer* yang terdiri dari satu *input layer*, satu *output layer*, dan satu atau lebih *hidden layer*. Setiap *layer* dalam jaringan *backpropagation* terhubung ke *layer* di depannya tanpa ada hubungan balik sehingga jaringan ini masuk ke dalam kategori jaringan *feedforward*. Banyaknya *neuron input* pada jaringan *backpropagation* sama dengan banyaknya fitur pada data dan banyaknya *neuron output* sesuai dengan banyaknya digit yang digunakan untuk membentuk kelas. Di sisi lain, banyaknya *neuron* pada *hidden layer* menentukan kompleksitas dan akurasi dari jaringan *backpropagation*. Penentuan banyaknya *neuron* pada *hidden layer* normalnya dilakukan melalui *trial and error* (Demuth dan Mark, 2002). Metode ini merupakan salah satu metode yang sangat baik dalam menangani masalah pengenalan pola-pola kompleks.

Di dalam jaringan *backpropagation*, setiap *neuron* yang berada di lapisan *input* berhubungan dengan setiap *neuron* yang ada di lapisan tersembunyi. Setiap *neuron* yang ada di lapisan tersembunyi terhubung dengan setiap *neuron* yang ada di lapisan *output*. Jaringan ini terdiri dari banyak lapisan (*multilayer network*). Ketika jaringan ini diberikan pola masukan sebagai pola pelatihan, maka pola tersebut menuju *neuron-neuron* lapisan tersembunyi untuk selanjutnya diteruskan pada *neuron-neuron* di lapisan keluaran. Kemudian *neuron-neuron* lapisan keluaran akan memberikan respon sebagai keluaran jaringan syaraf tiruan. Saat hasil keluaran tidak sesuai dengan yang diharapkan, maka keluaran akan disebarkan mundur (*backward*) pada lapisan tersembunyi kemudian dari lapisan tersembunyi menuju lapisan masukan (Puspitaningrum, 2006).

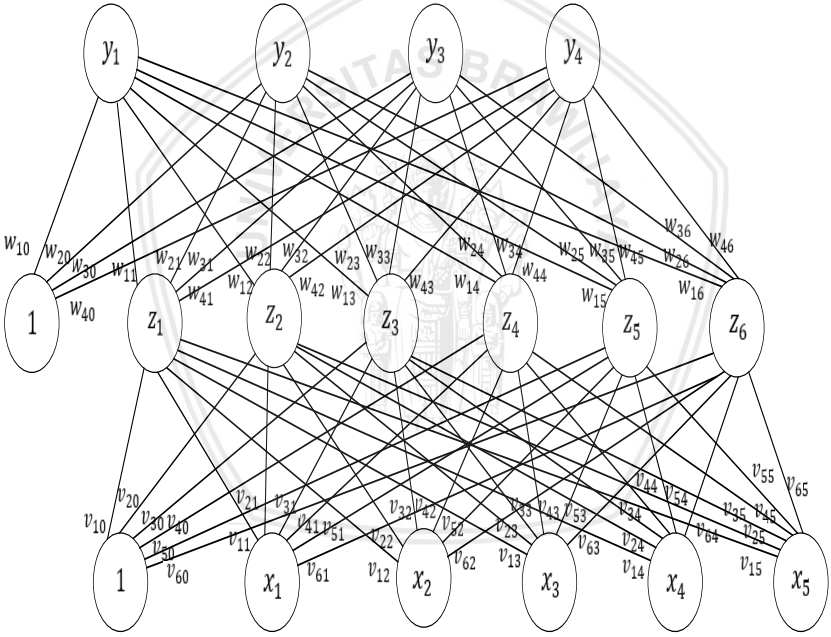
Tahap pelatihan ini merupakan langkah untuk melatih suatu jaringan syaraf tiruan, yaitu dengan cara melakukan perubahan bobot, sedangkan penyelesaian masalah akan dilakukan jika proses pelatihan tersebut telah selesai. Fase ini disebut sebagai fase pengujian (Puspitaningrum, 2006).

2.2.1 Arsitektur Jaringan *Backpropagation*

Setiap *neuron* dari *input layer* pada jaringan *backpropagation* selalu terhubung dengan setiap *neuron* yang berada pada layar tersembunyi, demikian juga setiap *neuron hidden layer* selalu terhubung dengan *neuron* pada *output layer*.

Backpropagation mempunyai beberapa *neuron* yang ada dalam satu atau lebih layar tersembunyi. Gambar 2.10 adalah arsitektur *backpropagation* dengan *n* buah masukan (ditambah sebuah bias), serta *m* buah keluaran. Nilai bias pada *input layer* bertujuan untuk mengolah data input ditambah dengan bobot v_{ji} yang masuk pada *hidden layer* dengan bantuan fungsi aktivasi.

v_{ji} merupakan bobot garis dari *input layer* x_i ke *neuron hidden layer* z_j (v_{j0} merupakan bobot garis yang menghubungkan bias di *input layer* ke *neuron hidden layer* z_j). w_{kj} merupakan bobot dari *neuron* lapisan tersembunyi z_j ke *neuron* keluaran y_k (w_{k0} merupakan bobot dari bias di lapisan tersembunyi ke *neuron* keluaran z_k) (Siang, 2005).



Gambar 2.10. Arsitektur *Backpropagation* Model 5-6-4

2.2.2 Algoritma Pelatihan Jaringan *Backpropagation*

Fausett (1994) menyatakan bahwa algoritma pelatihan jaringan *backpropagation* dapat dihasilkan menggunakan penurunan fungsi dan *error* dari jaringan. *Error* adalah perbedaan dari nilai keluaran yang dihasilkan (y_k) oleh jaringan dengan nilai target yang

ditentukan (t_k). Total *error* (E) adalah nilai yang diperoleh dengan menjumlahkan setiap *error neuron* keluaran y_k.

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - y_k)^2 \tag{2.8}$$

Dengan menggunakan bobot antara *neuron j* ke *neuron k* maka perubahan bobot (w_{kj}) diperoleh dengan penurunan persamaan 2.9.

$$\begin{aligned}\frac{\partial E}{\partial w_{kj}} &= \frac{\partial}{\partial w_{kj}} 0{,}5 \sum_{k=1}^{m}(t_k - y_k)^2 \\ &= \frac{\partial}{\partial w_{kj}} 0{,}5\big(t_k - f(y\_net_k)\big)^2 \\ &= -(t_k - y_k)\frac{\partial}{\partial w_{kj}} f(y\_net_k) \\ &= -(t_k - y_k) f'(y\_net_k)\frac{\partial}{\partial w_{kj}}(y\_net_k) \\ &= -(t_k - y_k) f'(y\_net_k) z_j \end{aligned} \tag{2.9}$$

(2.10)

dengan menggunakan persamaan 2.10 maka dapat diperoleh nilai δ_k pada persamaan 2.11.

$$\delta_k = -(t_k - y_k) f'(y\_net_k) \tag{2.11}$$

sedangkan dibawah ini merupakan perubahan bobot untuk v_{ji}

$$\begin{aligned}\frac{\partial E}{\partial v_{ji}} &= -\sum_{k=1}^{m}(t_k - y_k)\frac{\partial}{\partial v_{ji}} y_k \\ &= -\sum_{k=1}^{m}(t_k - y_k) f'(y\_net_k)\frac{\partial}{\partial v_{ji}}(y\_net_k) \\ &= -\sum_{k=1}^{m}\delta_k \frac{\partial}{\partial v_{ji}}(y\_net_k) \\ &= -\sum_{k=1}^{m}\delta_k w_{kj}\frac{\partial}{\partial v_{ji}} z_j \\ &= -\sum_{k=1}^{m}\delta_k w_{kj} f'(z\_net_j)(x_i) \end{aligned} \tag{2.12}$$

(2.13)

dengan menggunakan persamaan 2.13 maka dapat diperoleh nilai δ_j pada persamaan 2.14.

$$\delta_j = -\sum_{k=1}^{m} \delta_k w_{kj} f'(z\_net_j) \tag{2.14}$$

Kemudian perubahan bobot *neuron* tersembunyi ke *neuron* keluaran (Δw_{kj}) dapat dilihat pada persamaan 2.15.

$$\begin{aligned} \Delta w_{kj} &= -\alpha \frac{\partial E}{\partial w_{kj}} \\ &= -\alpha(-(t_k - y_k) f'(y\_net_k) z_j) \\ &= \alpha(t_k - y_k) f'(y\_net_k) z_j) \\ &= \alpha \delta_k z_j \end{aligned} \tag{2.15}$$

Dan perubahan bobot *neuron* masukan ke *neuron* tersembunyi (Δv_{ji}) dapat dilihat pada persamaan 2.16.

$$\begin{aligned} \Delta v_{ji} &= -\alpha \frac{\partial E}{\partial v_{ji}} \\ &= -\alpha(-\sum_{k=1}^{m} \delta_k w_{kj} f'(z\_net_j)(x_i)) \\ &= \alpha(\sum_{k=1}^{m} \delta_k w_{kj} f'(z\_net_j)(x_i)) \\ &= \alpha \delta_j x_i \end{aligned} \tag{2.16}$$

Sedangkan Siang (2005) menyatakan bahwa algoritma pelatihan yang lebih sederhana untuk jaringan *backpropagation* menggunakan satu *layer* tersembunyi (fungsi aktivasi *sigmoid biner*) adalah sebagai berikut.

Langkah 0
Inisialisasi bobot secara acak dengan bilangan acak kecil (sesuai *software* R).
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 *neuron* masukan menerima sinyal masukan dan sinyal tersebut diteruskan ke *neuron* lapisan tersembunyi di atasnya.

Langkah 4

Hitung keluaran di *neuron* tersembunyi $z_j, (j = 1,2, \dots, p)$

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i v_{ji} \tag{2.17}$$

Kemudian perhitungan yang sesuai dengan fungsi aktivasi yang digunakan adalah:

$$z_j = f(z\_net_j)$$

Bila yang digunakan adalah fungsi *sigmoid biner* maka bentuk fungsi tersebut adalah:

$$z_j = \frac{1}{1 + exp^{(-z\_net_j)}} \tag{2.18}$$

Kemudian mengirim sinyal tersebut ke semua *neuron* keluaran.

Langkah 5

Masing-masing *neuron* keluaran $(y_k, k = 1,2, \dots, m)$ dikalikan dengan faktor penimbang dan dijumlahkan:

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j w_{kj} \tag{2.19}$$

Menghitung kembali sesuai dengan fungsi aktivasi

$$y_k = f(y\_net_k)$$

Bila yang digunakan adalah fungsi *sigmoid biner* maka bentuk fungsi tersebut adalah:

$$y_k = \frac{1}{1 + exp^{(-y\_net_k)}} \tag{2.20}$$

Fase II: Propagasi mundur

Langkah 6

Hitung faktor δ *neuron* keluaran berdasarkan kesalahan di setiap *neuron* keluaran $(y_k, k = 1,2, \dots, m)$

$$\delta_k = (t_k - y_k) f'(y\_net_k) \tag{2.21}$$

Karena $f'(y\_net_k) = y_k$ menggunakan fungsi *sigmoid biner*, maka

$$f'(y\_net_k) = f(y\_net_k)\big(1 - f(y\_net_k)\big)$$
$$f'(y\_net_k) = y_k(1 - y_k)$$

Maka

$$\delta_k = (t_k - y_k) y_k (1 - y_k) \quad (2.22)$$

δ_k merupakan *neuron* kesalahan yang akan dipakai dalam perubahan bobot lapisan di bawahnya (langkah 7).

Hitung suku perubahan bobot w_{kj} (yang akan digunakan umtuk merubah bobot w_{kj}) dengan laju percepatan α.

$$\Delta w_{kj} = \alpha \delta_k z_j \quad ; k = 1,2,\dots,m \quad ; j = 0,1,\dots,p \quad (2.23)$$

Langkah 7

Hitung faktor δ *neuron* tersembunyi berdasarkan kesalahan setiap *neuron* tersembunyi z_j ($j = 1,2,\dots,p$)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \quad (2.24)$$

Faktor δ *neuron* tersembunyi :

$$\delta_j = \delta\_net_j f'(z\_net_j)$$

$$\delta_j = \delta\_net_j z_j (1 - z_j) \quad (2.25)$$

Hitung suku perubahan bobot v_{ji} (yang akan digunakan untuk merubah bobot v_{ji})

$$\Delta v_{ji} = \alpha \delta_j x_i \quad ; j = 1,2,\dots,p \quad ; i = 0,1,\dots,n \quad (2.26)$$

Fase III: Perubahan bobot

Langkah 8

Hitung semua perubahan bobot, masing-masing keluaran *neuron* diperbaiki bias dan penimbangnya

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \quad (2.27)$$

$$(k = 1,2,\dots,m \,; j = 0,1,\dots,p)$$

Masing-masing *neuron* tersembunyi diperbaiki bias dan penimbangnya

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \quad (2.28)$$

$$(j = 1,2,\dots,p \,; i = 0,1,\dots,n)$$

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

Langkah 9
Uji kondisi pemberhentian (akhir iterasi). Menghitung *error* yang dihasilkan menggunakan persamaan 2.8.

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - y_k)^2$$

Setiap *neuron* keluaran akan menghasilkan nilai, nilai *ouput* yang paling besar akan bernilai satu, sedangkan *ouput* lainnya akan bernilai nol (Haryati, dkk, 2016).

Keterangan:
z_j = N*euron* ke-*j* pada lapisan tersembunyi
z_net_j = Keluaran untuk *neuron* z_j
z_j = Nilai aktifasi dari *neuron* z_j
y_k = *Neuron* ke-k pada lapisan keluaran
y_net_k = Keluaran untuk *neuron* y_k
y_k = Nilai aktifasi dari *neuron* y_k
w_{kj} = Nilai penimbang sambungan dari z_j ke *neuron* y_k
Δw_{kj} = Selisih antara $\Delta w_{kj}(t)$ dengan $\Delta w_{kj}(t+1)$
v_{ji} = Nilai penimbang sambungan dari x_i ke *neuron* z_j
Δv_{ji} = Selisih antara $\Delta v_{ji}(t)$ dengan $\Delta v_{ji}(t+1)$
δ_k = Faktor pengaturan nilai penimbang sambungan pada lapisan tersembunyi
α = Konstanta laju pelatihan (*learning rate*) $0 < \alpha < 1$.

2.2.3 Aturan *Backpropagation*

Aturan pelatihan jaringan *backpropagation* terdiri dari tiga tahapan, yaitu *feedforward*, *backward propagation*, dan perubahan bobot. Setiap perubahan bobot yang terjadi dapat mengurangi *error*. Siklus perubahan bobot (*epoch*) dilakukan pada setiap set pelatihan sehingga kondisi berhenti dicapai. Apabila mencapai jumlah *epoch* yang diinginkan atau hingga sebuah nilai ambang yang ditetapkan terlampaui selama kesalahan (*error*) menurun maka pelatihan akan terus dijalankan, akan tetapi jika kesalahan meningkat maka pelatihan dapat dihentikan. Umumnya metode pelatihan dengan *backpropagation* tidak akan menghasilkan MSE yan bernilai nol, untuk itu dapat digunakan nilai *error* yang cukup kecil misalnya sebesar 0,1 (Patterson 1996).

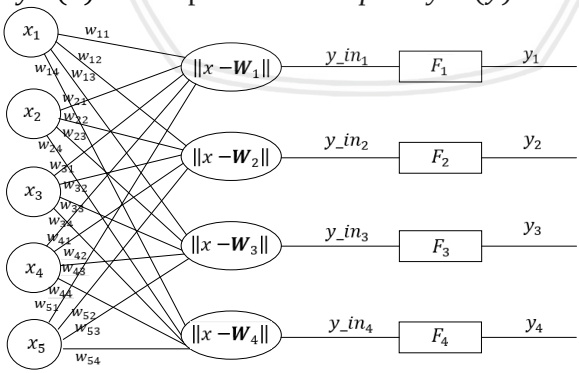
2.3 *Learning Vector Quantization* (LVQ)

Learning vector quantization (LVQ) merupakan metode klasifikasi berbasis jaringan syaraf tiruan yang mengimplementasikan konsep kompetisi. Kompetisi pada LVQ berbasiskan *winner-take-all* yang berarti hanya ada satu *neuron* pemenang dalam proses kompetisi. Dibandingkan dengan model jaringan syaraf tiruan lainnya, LVQ memiliki keunggulan dari sisi ketepatan (Kohonen, 1990). Menurut Widodo (2005) LVQ mengklasifikasikan vektor *input* dalam kelas yang sama dengan *neuron output* yang memliki vektor bobot yang paling dekat dengan vektor *input.*

2.3.1 Arsitektur Jaringan *Learning Vector Quantization* (LVQ)

Menurut Putro (2011), arsitektur LVQ sama halnya dengan *self organizing map* (SOM), LVQ juga terdiri dari dua lapisan, *input* (x) dan *output* (y). Lapisan antar keduanya dihubungkan oleh bobot tertentu yang sering disebut sebagai vektor pewakil (w). Informasi yang diberikan ke jaringan pada saat pembelajaran bukan hanya vektor data saja melainkan informasi kelas dari data juga ikut dimasukkan. Jika hasil pemrosesan jaringan memberikan hasil klasifikasi yang sama dengan informasi kelas yang diberikan di awal, maka vektor pewakil akan disesuaikan agar lebih dekat dengan vektor masukan. Sebaliknya Jika hasil klasifikasi tidak sama dengan informasi kelas yang diberikan di awal, maka vektor pewakil akan menjauh dari vektor masukan.

Arsitektur metode LVQ yang digunakan pada penelitian ini dapat dilihat pada Gambar 2.11. Dengan menggunakan lima *neuron input layer*(x) dan empat *neuron output layer* (y).



Gambar 2.11. Arsitektur LVQ

Keterangan:
$\boldsymbol{x}$ = Vektor masukan yang terdiri dari *neuron* $(x_1, x_2, \dots, x_n)$ pada lapisan *input*
$\boldsymbol{w}$ = Vektor bobot untuk lapisan *output*
$\|x - \boldsymbol{W}_1\|$= Selisih antara vektor *input* dan vektor bobot pertama
$\|x - \boldsymbol{W}_2\|$= Selisih antara vektor *input* dan vektor bobot kedua
y_in_1 = *Input* untuk lapisan kompetitif pertama
y_in_2 = *Input* untuk lapisan kompetitif kedua
F = Lapisan kompetitif
y = Nilai keluaran pada lapisan *output*

Pemrosesan yang terjadi pada setiap *neuron* adalah mencari jarak antara suatu vektor *input* ke bobot yang bersangkutan (w_1 dan w_2). w_1 adalah vektor bobot yang menghubungkan setiap *neuron* pada *input layer* ke *neuron* pertama pada *output layer*, sedangkan w_2 adalah vektor bobot yang menghubungkan setiap *neuron* pada lapisan input ke *neuron* kedua lapisan *output*. Fungsi aktivasi F_1 akan memetakan y_in_1 ke y_1=1 apabila $\|x - w_1\| < \|x - w_2\|$, dan y_1=0 jika $\|x - w_1\| > \|x - w_2\|$. Demikian pula dengan yang terjadi pada fungsi aktivasi F_2, akan memetakan y_in_2 ke y_2= 1 apabila $\|x - w_2\| < \|x - w_1\|$, dan y_2= 0 jika $\|x - w_2\| > \|x - w_1\|$.

Sebuah jaringan LVQ memiliki lapisan kompetitif pertama dan lapisan linear kedua. Pada lapisan kompetitif belajar untuk mengklasifikasikan vektor masukan. Lapisan linear mengubah kelas lapisan kompetitif ke klasifikasi target yang didefinisikan oleh pengguna. Hal ini mengacu pada kelas dipelajari oleh lapisan kompetitif sebagai subkelas dan kelas-kelas dari linear sebagai kelas target (Demuth dan Mark, 2002). Banyaknya vektor bobot/*size codebook* yang digunakan pada penelitian ini diperoleh dengan menggunakan persamaan 2.29 (Rcran, 2019) yaitu:

$$size\ codebook = \min(round(0.4 \times\ ng \times \left(ng - 1 + \left(\frac{p}{2}\right), 0\right), n) \quad (2.29)$$

Keterangan:
ng = Banyaknya kelas
p = Banyaknya faktor yang digunakan pada penelitian
n = Banyaknya data.

Menurut Putri (2012), faktor-faktor yang mempengaruhi pembelajaran pada LVQ adalah sebagai berikut:
1. Inisialisasi bobot awal

Inisialisasi bobot awal pada LVQ dapat dilakukan dengan beberapa cara, antara lain memilih salah satu bobot awal pada tiap kelas, memilih bobot awal secara acak, dan inisialisasi awal dengan nilai nol.

2. Laju pelatihan (α)
   Pada LVQ, laju pelatihan akan berpengaruh pada saat pergeseran bobot. Oleh karena itu sebaiknya nilai dari laju pelatihan tersebut jangan terlalu besar dan terlalu kecil.

Pembelajaran LVQ ke dalam lapisan kompetitif didasarkan pada satu *set input* / pasangan target.

$$\{s^{(q)}: t^{(q)}\}, \qquad q = 1,2, \dots, Q$$

Keterangan:

$s^{(q)}$ = Vektor *input*

$t^{(q)}$ = Vektor *output*

Lapisan kompetitif pada LVQ mencakup sebuah *subnet* kompetitif dan sebuah lapisan linear. Pada lapisan kompetitif, *neuron* dalam lapisan pertama belajar vektor prototipe yang memungkinkan untuk mengklasifikasikan wilayah ruang *input*. Hubungan antara vektor *input* dan salah satu bobot diukur dari jarak *Euclid* diantara vektor *input* dan bobot. Sebuah *subnet* digunakan untuk menemukan elemen terkecil pada data *input*.

$$n^{(1)} = \begin{bmatrix} \left\| x - \boldsymbol{W}_1^{(1)} \right\| \\ \left\| x - \boldsymbol{W}_2^{(1)} \right\| \\ \vdots \\ \left\| x - \boldsymbol{W}_Q^{(1)} \right\| \end{bmatrix} \tag{2.30}$$

Elemen yang sesuai dengan nilai satu berarti bahwa vektor *input* termasuk dari kelas terkait dan bagian yang lain dinilai dengan nol sehingga bukan termasuk dari kelas terkait. Suatu *subnet* sebagai sebuah nilai vektor dengan fungsi sebagai berikut:

$$\boldsymbol{a}^{(1)} = compet(n^{(1)}) \tag{2.31}$$

Lapisan kedua (lapisan linear) dari jaringan LVQ kemudian digunakan untuk menggabungkan *subclass* ke dalam sebuah *single class*. Hal ini dilakukan dengan menggunakan $\boldsymbol{W}^{(2)}$. Matriks bobot yang mempunyai elemen

$$\boldsymbol{a}^{(2)} = \boldsymbol{W}^{(2)}\boldsymbol{a}^{(1)} \tag{2.32}$$

$$w_{ij} = \begin{cases} 1, jika\ neuron\ i\ termasuk\ sebuah\ subclass\ j \\ 0, yang\ lainnya \end{cases}$$

Ketika $\boldsymbol{W}^{(2)}$. Adalah *set*, $\boldsymbol{W}$ tidak akan berubah. Selain itu, bobot $\boldsymbol{W}^{(1)}$, untuk lapisan kompetitif harus dilatih menggunakan aturan kohonen LVQ.

2.3.2 Algoritma Pelatihan Jaringan *Learning Vector Quantization* (LVQ)

Menurut Fausett (1994), algoritma pelatihan jaringan LVQ adalah sebagai berikut:

Langkah 0
Inisialisasi vektor referensi.
Langkah 1
Inisialisasi laju pelatihan $(\alpha) = 0 < \alpha < 1$
Langkah 2
Jika kondisi penghentian belum terpenuhi, lakukan langkah 3—7.
Langkah 3
Untuk setiap vektor masukan pelatihan x, kerjakan langkah 4—5.
Langkah 4
Mencari nilai $\boldsymbol{J}$ sehingga minimum $\|x - w_j\|$, kemudian sebut dengan C_j, T merupakan kategori/target yang benar untuk *vektor training*
Langkah 5
Perbaharui nilai w_j dengan ketentuan sebagai berikut:
Jika $T = C_j$ maka

$$w_j \text{ (baru)} = w_j \text{ (lama)} + \alpha\|x - w_j\| \quad (2.33)$$

Jika $T \neq C_j$ maka

$$w_j \text{ (baru)} = w_j \text{ (lama)} - \alpha\|x - w_j\| \quad (2.34)$$

Langkah 6
Kurangi laju pelatihan (α)
Langkah 7
Uji syarat berhenti, cacah iterasi atau laju pelatihan.

2.3.3 Aturan *Learning Vector Quantization* (LVQ)

Aturan pelatihan jaringan *learning vector quantization* (LVQ) dimulai dengan pengklasifikasian vektor *input* pada lapisan kompetitif. Pada setiap proses iterasi yang dilakukan, satu dari vektor *training* dipersembahkan ke jaringan sebagai *input* x dan jarak *Euclid* dari vektor *input* ke setiap vektor prototipe (membentuk kolom

matriks bobot) dihitung. *Neuron j** membuat kompetisi jika jarak *euclid* diantara *x* dan *j* prototype* vektor paling kecil. Aktivasi $a^{(1)}$ dikalikan oleh $\boldsymbol{W}^{(2)}$ di sebelah kanannya untuk mendapatkan input $\boldsymbol{n}^{(2)}$. *Output* yang dihasilkan adalah $\boldsymbol{a}^{(2)} = \boldsymbol{n}^{(2)}$, selama fungsi transfer *neuron output* adalah fungsi identitas $\boldsymbol{a}^{(2)}$ juga hanya mempunyai satu nol elemen *k**, hal ini mengidentifikasikan bahwa vektor input tersebut termasuk kelas *k**.

Berdasarkan aturan Kohonen, *neuron* dengan bobot paling mendekati vektor masukan akan diperbaiki dan dibuat lebih dekat lagi. Pertambahan dan berubahnya vektor masukan yang diberikan, mengakibatkan vektor masukan akan terbagi menjadi beberapa kelompok. Jika x telah diklasifikasikan secara benar, lalu vektor bobot $w_{.j*}^{(1)}$ pemenang dari *hidden neuron* dipindahkan mendekat ke x.

$$\Delta w_{.j*}^{(1)} = \alpha\left(x - w_{.j*}^{(1)}\right) \quad jika\ \left(a_{k*}^{2} = t_{k*} = 1\right) \tag{2.35}$$

Akan tetapi jika x salah diklasifikasikan, hal ini mengidentifikasikan bahwa *hidden neuron* yang salah memenangkan kompetisi. Dalam hal ini bobot dipindahkan menjauh dari x.

$$\Delta w_{.j*}^{(1)} = -\alpha\left(x - w_{.j*}^{(1)}\right) \quad jika\ \left(a_{k*}^{2} = t_{k*} \neq 1\right) \tag{2.36}$$

Setelah dilakukan pelatihan (*training*), akan diperoleh bobot akhir (w) yang digunakan untuk melakukan simulasi, pengujian atau klasifikasi selanjutnya (Kusumadewi, 2004).

2.4 Prosedur Ketepatan Klasifikasi

Kriteria perbandingan teknik klasifikasi didasarkan pada ketepatan klasifikasinya yang dikenal dengan istilah *hit ratio*. *Hit ratio* merupakan nilai ketepatan klasifikasi dari suatu observasi berdasarkan suatu fungsi klasifikasi tertentu (Johnson dan Wichern, 2007).

Menurut Johnson dan Wichern (2007), terjadinya kesalahan klasifikasi suatu observasi merupakan hal yang sangat mungkin terjadi. Hal ini dikarenakan terkadang terdapat beberapa observasi yang tidak berasal dari kelompok tertentu tetapi dimasukkan ke dalam kelompok tersebut.

Tabel 2.1 Tabel Klasifikasi (Johnson dan Wichern, 2007)

Kelompok	$\hat{y}$ prediksi	
	$\widehat{y_1}$	$\widehat{y_2}$
π_0	n_{00}	$n_{01} = n_0 - n_{00}$
π_1	$n_{10} = n_1 - n_{11}$	n_{11}

Keterangan:

π_0 = Kelompok pertama

π_1 = Kelompok kedua

n_{00} = Frekuensi anggota pengamatan yang terklasifikasi secara tepat pada π_0

n_{01} = Frekuensi anggota pengamatan yang terklasifikasi secara tidak tepat pada π_0

n_{11} = Frekuensi anggota pengamatan yang terklasifikasi secara tepat pada π_1

n_{10} = Frekuensi anggota pengamatan yang terklasifikasi secara tidak tepat pada π_1

n_0 = Penjumlahan dari n_{00} dan n_{01}

n_1 = Penjumlahan dari n_{11} dan n_{10}

Persamaan *hit ratio* (Johnson dan Wichern, 2007):

$$Hit\ ratio = \frac{total\ objek\ yang\ tepat\ dikasifikasikan}{total\ sampel} \times 100\%$$

$$Hit\ Ratio = \frac{n_{00} + n_{11}}{n_0 + n_1} \times 100\% \qquad (2.37)$$

2.5 Ketepatan Fungsi Klasifikasi

Menurut Dianiati (2013), sebelum melakukan klasifikasi, data dibagi menjadi dua. Bagian pertama disebut sebagai data *training* yang digunakan untuk membentuk arsitektur jaringan syaraf tiruan yang optimal, sedangkan bagian kedua disebut sebagai data *testing* untuk menguji arsitektur optimal yang diperoleh oleh data *training*. Hair dkk (2010) menjelaskan prinsip pembagian sampel yang paling popular adalah 50%-50%, tetapi kebanyakan peneliti menggunakan prinsip 60%-40% atau 80%-20% karena memang tidak ada aturan baku mengenai pembagian data tersebut.

Ketepatan klasifikasi yang telah dilakukan dapat ditentukan menggunakan nilai *hit ratio* dari setiap model jaringan syaraf yang telah dihasilkan sebelumnya. Nilai *hit ratio* dapat diperoleh menggunakan persamaan 2.37.

2.6 Statistika Deskriptif

Statistika deskriptif adalah metode-metode yang berkaitan dengan pengumpulan dan penyajian suatu data sehingga memberikan informasi yang berguna (Walpole, 1982). Menurus Sugiyono (2007) Statistik deskriptif berfungsi untuk mendeskripsikan atau memberi gambaran terhadap objek yang diteliti melalui data sampel atau populasi.

2.6.1 Rata-rata

Rata-rata atau nilai tengah merupakan suatu ukuran pusat data apabila data tersebut diurutkan dari bilangan terkecil ke bilangan terbesar (Walpole, 1982). Persamaan di bawah ini merupakan rumus untuk menghitung rata-rata dari suatu data berkelompok dan tidak berkelompok menurut Yitnosumarto (1990).

1. Data berkelompok

Pengamatan berkelompok merupakan data yang disajikan dalam bentuk tabel frekuensi. Nilai tengah dari data tersebut dapat dihitung menggunakan persamaan 2.38.

$$\bar{x} = \frac{\sum_{i=1}^{k} f_i x_i}{\sum_{i=1}^{k} f_i} \tag{2.38}$$

Keterangan:

f_i = Frekuensi pada kelas ke-i $(i = 1,2, \dots, k)$

x_i = Nilai tengah kelas ke-i $(i = 1,2, \dots, k)$

k = Banyaknya kelas

2. Data tidak berkelompok

Jika x merupakan peubah acak dengan nilai pengamatan $x_1, x_2, \dots, x_n$ nilai tengah dari data tersebut dapat dihitung menggunakan persamaan 2.39.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.39}$$

Keterangan:

x_i = Pengamatan ke-i $(i = 1,2, \dots, n)$

n = Banyaknya data

2.6.2 Ragam

Ragam merupakan ukuran penyebaran suatu pengamatan terhadap rata-rata data (Walpole, 1982). Ragam untuk sebuah contoh acak $x_1, x_2, \dots, x_n$ dapat dihitung menggunakan persamaan 2.40 di bawah ini.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \quad (2.40)$$

Keterangan:
x_i = Pengamatan ke-i $(i = 1,2,\dots,n)$
n = Banyaknya data

2.6.3 Median

Median adalah nilai pengamatan yang terletak di tengah dari suatu data yang telah diurutkan (Yitnosumarto, 1990). Untuk data tidak berkelompok dengan n adalah banyaknya data, yang dimaksud dengan data diurutkan adalah $x_1, x_2, \dots, x_n$ dan $x_1 \leq x_2 \leq x_3 \leq \cdots \leq x_n$.

Rumus untuk menghitung nilai median bergantung pada banyaknya data atau n, persamaan 2.41 merupakan rumus untuk menghitung median dengan n bilangan genap sedangkan persamaan 2.42 merupakan rumus untuk menghitung median dengan n bilangan ganjil.

$$Me = \frac{1}{2}\left(x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}\right) \quad (2.41)$$

$$Me = x_{\left(\frac{n+1}{2}\right)} \quad (2.42)$$

2.6.4 Modus

Modus merupakan nilai yang paling sering muncul atau yang mempunya frekuensi paling tinggi pada suatu pengamatan. Modus tidak selalu ada, hal ini terjadi apabila semua pengamtan mempunyai frekuensi terjadi yang sama. Untuk suatu data tertentu, jika terdapat beberapa nilai dengan frekuensi yang tinggi, maka terdapat lebih dari satu modus pada data (Walpole, 1982).

Cara mengetahui modus untuk data berkelompok terdapat pada persamaan 2.43 sedangkan modus data tidak berkelompok dapat menggunakan perhitungan manual pada data.

$$Mo = Bb + \left(\frac{a}{a+b}\right)I \quad (2.43)$$

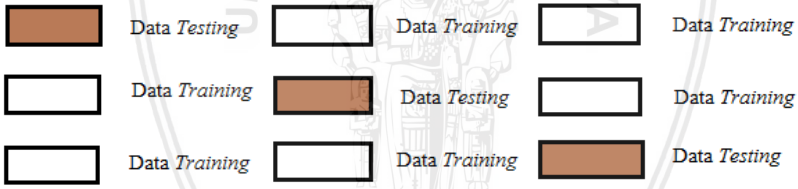
Keterangan:
Bb = Batas kelas terbawah pada kelas dengan frekuensi terbesar
a = Selisih frekuensi tertinggi dengan frekuensi kelas sebelumnya
b = Selisih frekuensi tertinggi dengan frekuensi kelas sesudahnya
I = Selang dalam kelas (lebar kelas)

2.7 *Cross Validation*

Cross validation adalah metode statistik yang mengevaluasi dan membandingkan algoritma pembelajaran dengan membagi data menjadi dua bagian. Pertama data *training* yang digunakan untuk pelatihan model dan kedua data *testing* yang digunakan untuk validasi model. Pada *Cross validation* data bergantian menjadi data pelatihan dan data validasi sehingga setiap data pelatihan memiliki peluang untuk divalidasi (Rafaeilzadeh dkk, 2009).

2.7.1 *K-Fold Cross Validation*

K-fold cross validation adalah metode validasi dengan membagi data ke dalam *k-subset*, kemudian melakukan pengulangan sebanyak *k* untuk pelatihan dan pengujian. Pada setiap pengulangan, digunakan satu *subset* sebagai data validasi dan *subset* lainnya sebagai data pelatihan. Keuntungan dari metode ini adalah setiap data, minimal akan menjadi data validasi sebanyak satu kali dan akan menjadi data pelatihan juga minimal sebanyak satu kali (Rafaeilzadeh dkk, 2009). Gambar 2.12 di bawah ini merupakan contoh *cross validation* menggunakan *k=3*.



Gambar 2.12. Contoh 3-*Fold Cross Validation*

Dalam pembelajaran pada *machine learning* maka 10-*fold cross validation* (*k*=10) merupakan pembagian *k-subset* yang paling umum untuk digunakan. Nilai *k* ini adalah pembagian yang baik karena membuat prediksi menggunakan 90% dari data, membuatnya lebih mungkin digeneralisasikan ke data lengkap (Rafaeilzadeh dkk, 2009).

2.8 Status Gizi Balita

Balita merupakan anak yang berada dalam rentang usia 1—5 tahun. Balita merupakan masa pertumbuhan tubuh dan otak yang sangat pesat dalam pencapaian keoptimalan fungsinya. Pertumbuhan dasar yang akan mempengaruhi serta menentukan perkembangan

kemampuan berbahasa, kreatifitas, kesadaran sosial, emosional, dan intelegensia (Supartini, 2004).

Status gizi merupakan keadaan status pada tubuh manusia yang berhubungan dengan konsumsi makanan serta dipengaruhi oleh berbagai faktor internal maupun eksternal seperti usia, jenis kelamin, aktivitas fisik, penyakit, serta keadaan sosial ekonomi (Wolley dkk, 2016). Kebutuhan gizi pada masa balita membutuhkan lebih banyak nutrisi, karena masa balita adalah periode keemasan yaitu periode kehidupan yang sangat penting bagi perkembangan fisik dan mental. Pada masa ini pula balita banyak melakukan dan menemukan hal-hal baru. Dalam hal ini nutrisi yang baik memegang peran penting (Hasdianah dan Peristyowati, 2014).

Status gizi balita dapat dinilai melalui tiga indeks, yaitu berat badan menurut umur (BB/U), tinggi badan menurut umur (TB/U), dan berat badan menurut tinggi badan (BB/TB). Ketiga nilai tersebut dibandingkan dengan simpangan baku pertumbuhan oleh WHO. Batasan untuk kategori status gizi balita menurut indeks BB/U, TB/U, dan BB/TB menurut WHO dapat dilihat pada Tabel 2.2.

Tabel 2.2. Pengertian Kategori Status Gizi Balita (Kepmenkes No. 1995/MENKES/SK/XII/2010)

Indikator	Status Gizi	*Z_Score*
BB/U	Gizi Buruk	$Z\_Score < -3$
	Gizi Kurang	$-3 < Z\_Score < -2$
	Gizi Baik	$-2 < Z\_Score < 2$
	Gizi Lebih	$Z\_Score > 2$
TB/U	Sangat Pendek	$Z\_Score < -3$
	Pendek	$Z\_Score < -3$
	Normal	$Z\_Score \geq 2$
BB/TB	Sangat Kurus	$Z\_Score < -3$
	Kurus	$-3 < Z\_Score < -2$
	Normal	$-2 < Z\_Score < 2$
	Gemuk	$Z\_Score > 2$

Menurut Supariasa dan Nyoman (2002) nilai simpangan baku atau SD dapat dihitung melalui bebarapa kategori, antara lain:

1) 1 SD (1 *Z_score* kurang lebih sama dengan 11% dari median BB/U).
2) 1 SD (1 *Z_score* kira-kira 10% dari median BB/TB).

3) 1 SD (1 *Z_score* kira-kira 5% dari median TB/U).

Terdapat banyak faktor yang dapat berpengarauh terhadap status gizi balita, faktor–faktor yang digunakan pada penenelitian ini antara lain:

1. Gizi Kurang

Gizi kurang adalah keadaan kurang zat gizi tingkat sedang yang disebabkan oleh rendahnya asupan energi dan protein dalam waktu cukup lama yang ditandai dengan berat badan menurut umur (BB/U) berada pada $-3 < Z\_Score < -2$ tabel baku WHO-NCHS. Gizi kurang banyak terjadi pada anak usia kurang dari lima tahun.Gizi buruk adalah kondisi gizi kurang hingga tingkat yang berat dan di sebabkan oleh rendahnya konsumsi energi dan protein dari makanan sehari-hari dan terjadi dalam waktu yang cukup lama (Khaidirmuhaj, 2009).

2. Balita Gemuk

Kelebihan berat badan (obesitas) adalah kelebihan lemak tubuh yang terakumulasi sedemikian rupa sehingga menimbulkan dampak merugikan bagi kesehatan yang dapat menurunkan harapan hidup atau meningkatkan masalah kesehatan (WHO, 2010). Obesitas disebabkan oleh kombinasi antara asupan energi makanan yang berlebihan serta kurangnya aktivitas fisik. Obesitas adalah penyebab kematian yang dapat dicegah paling utama di dunia, dengan prevalensi pada orang dewasa dan anak yang semakin meningkat, sehingga obesitas sebagai salah satu masalah kesehatan masyarakat.

3. Inisiasi Menyusu Dini (IMD)

Inisiasi menyusu dini (IMD) adalah proses membiarkan bayi dengan nalurinya sendiri dapat menyusu segera dalam satu jam pertama setelah bayi lahir. Persamaan dengan kontak kulit antara bayi dengan kulit ibunya, bayi dibiarkan setidaknya selama satu jam di dada ibu, sampai bayi menyusu sendiri (Depkes, 2008).

4. Asi Ekslusif

Menurut Peraturan Pemerintah Nomor 33 Tahun 2012 pada ayat 1 diterangkan air susu ibu eksklusif yang selanjutnya disebut ASI eksklusif adalah ASI yang diberikan kepada bayi sejak dilahirkan selama enam bulan, tanpa menambahkan atau mengganti dengan makanan atau minuman lain.

5. Garam Beriodium

Garam beriodium adalah garam yang telah diperkaya dengan yodium, yang dibutuhkan tubuh untuk membuat hormon yang mengatur pertumbuhan dan perkembangan kecerdasan (Depkes,

2009). Kekurangan yodium akan mengalami gangguan fisik maupun mental mulai dari yang ringan maupun berat. Gangguan pertumbuhan fisik antara lain mencakup penyakit gondok, badan kerdil, gangguan motorik seperti kesulitan berdiri ataupun berjalan normal, bisu, tuli, atau mata juling sedangkan gangguan mental termasuk berkurangnya kecerdasan (Supariasa dan Nyoman, 2002)

BAB III
METODE PENELITIAN

3.1. Sumber Data

Data yang digunakan adalah data sekunder yang diperoleh dari Pemantauan Status Gizi (PSG) oleh Kementerian Kesehatan Republik Indonesia pada tahun 2016. Banyaknya data yang digunakan sebanyak 514 kabupaten/kota di Indonesia. Variabel respon pada penelitian ini adalah:

1. Masalah Gizi Balita

Status gizi merupakan keadaan status pada tubuh manusia yang berhubungan dengan konsumsi makanan serta dipengaruhi oleh berbagai faktor internal maupun eksternal seperti usia, jenis kelamin, aktivitas fisik, penyakit, dan keadaan sosial ekonomi (Wolley dkk, 2016). Tabel 3.2 merupakan kategori masalah gizi balita yang akan digunakan pada penelitian sedangkan Tabel 3.1 menjelaskan istilah yang digunakan pada Tabel 3.2.

Tabel 3.1. Istilah Gizi (WHO, 1997)

Istilah	Pengertian
Gizi Kurang	Gabungan gizi buruk dan gizi kurang
Pendek	Gabungan sangat pendek dan pendek
Kurus	Gabungan sangat kurus dan kurus

Tabel 3.2. Kategori Masalah Gizi (WHO, 1997)

Masalah Gizi	Pravelensi Pendek	Prevelensi Kurus
Baik	Kurang dari 20%	Kurang dari 5%
Akut	Kurang dari 20%	5% atau lebih
Kronis	20% atau lebih	Kurang dari 5%
Akut + Kronis	20% atau lebih	5% atau lebih

Sesuai dengan standar WHO (1997), maka suatu wilayah dikatakan mempunyai kategori baik jika pravelensi balita pendek kurang dari 20% dan prevalensi balita kurus kurang dari 5% sedangkan wilayah dikatakan mempunyai kategori masalah gizi akut jika prevelensi balita pendek kurang dari 20% dan prevelensi balita kurus kurang dari 5%. Berdasarkan teori tersebut, maka kategori peubah respon dapat dibagai menjadi empat kelompok antara lain:

a. 1 = Masalah gizi balita akut + kronis
b. 2 = Masalah gizi balita kronis
c. 3 = Masalah gizi balita akut
d. 4 = Masalah gizi balita baik

Sedangkan variabel prediktor yang digunakan pada penelitian ini antara lain:

1. Persentase balita dengan gizi kurang (X_1)
2. Persentase bayi gemuk (X_2)
3. Persentase bayi mendapat IMD < satu jam (X_3)
4. Persentase bayi yang mendapatkan ASI ekslusif (X_4)
5. Persentase bayi yang mengkonsumsi garam beriodium (X_5)

3.2. Metode Penelitian

Tahapan analisis yang dilakukan untuk menyelesaikan permasalahan dalam penelitian ini adalah sebagai berikut:

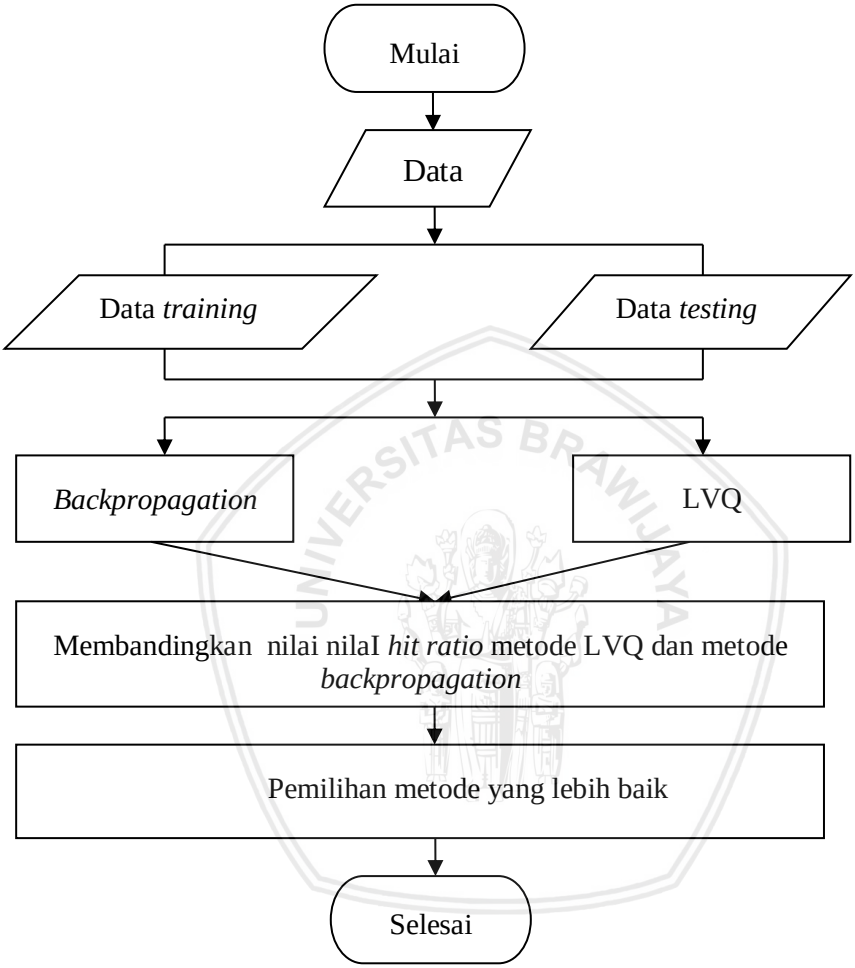
1. Membagi data menjadi dua bagian, yaitu 80% data untuk pelatihan (*training*) dan 20% data *testing* untuk validasi.
2. Melakukan analisis jaringan syaraf tiruan dengan metode pelatihan *backpropagation*. Di bawah ini merupakan tahapan analisis metode *backpropagation*
   a. Menentukan target *error* penelitian, maksimum *epoch, learning rate*, dan menginisialisasi bobot-bobot awal secara acak (*random*).
   b. Membuat arsitektur jaringan *backpropagation.*
   c. Menentukan fungsi aktivasi yang akan digunakan dalam jaringan *feed forward.*
   d. Melakukan analisis metode *backpropagation* pada *output layer*.
   e. Melakukan analisis metode *backpropagation* pada *hidden layer*.
   f. Pembaharuan bobot jaringan *backpropagation*.
   g. Menguji kondisi berhenti melalui nilai *error* yang dihasilkan atau maksimum *epoch* yang telah dilakukan. Jika salah satu kondisi berhenti terpenuhi maka iterasi dihentikan dan analisis telah selesai, tapi jika kondisi berhenti belum terpenuhi, maka ulangi langkah d—g.
3. Melakukan analisis jaringan syaraf tiruan dengan metode pelatihan metode *learning vector quantization* (LVQ). Di

bawah ini merupakan tahapan analisis metode *learning vector quantization* (LVQ):

a. Membentuk matriks *input* pada data *training* serta membentuk vektor atau faktor klasifikasi untuk data *training*, kolom matriks bobot menyatakan banyaknya komponen dalam sebuah vektor sedangkan baris menyatakan banyaknya maksimum kelompok yang akan dibentuk (Siang, 2005).
b. Inisialisasi bobot (*w*) dari jaringan LVQ.
c. Menentukan laju latihan (α).
d. Memperbaharui bobot pada lapisan kompetitif sehingga didapatkan bobot optimum.
e. Membentuk arsitektur optimal.
f. Melakukan klasifikasi ulang pada data *testing* berdasarkan arsitektur terbaik metode LVQ yang dibentuk dari data *training*.

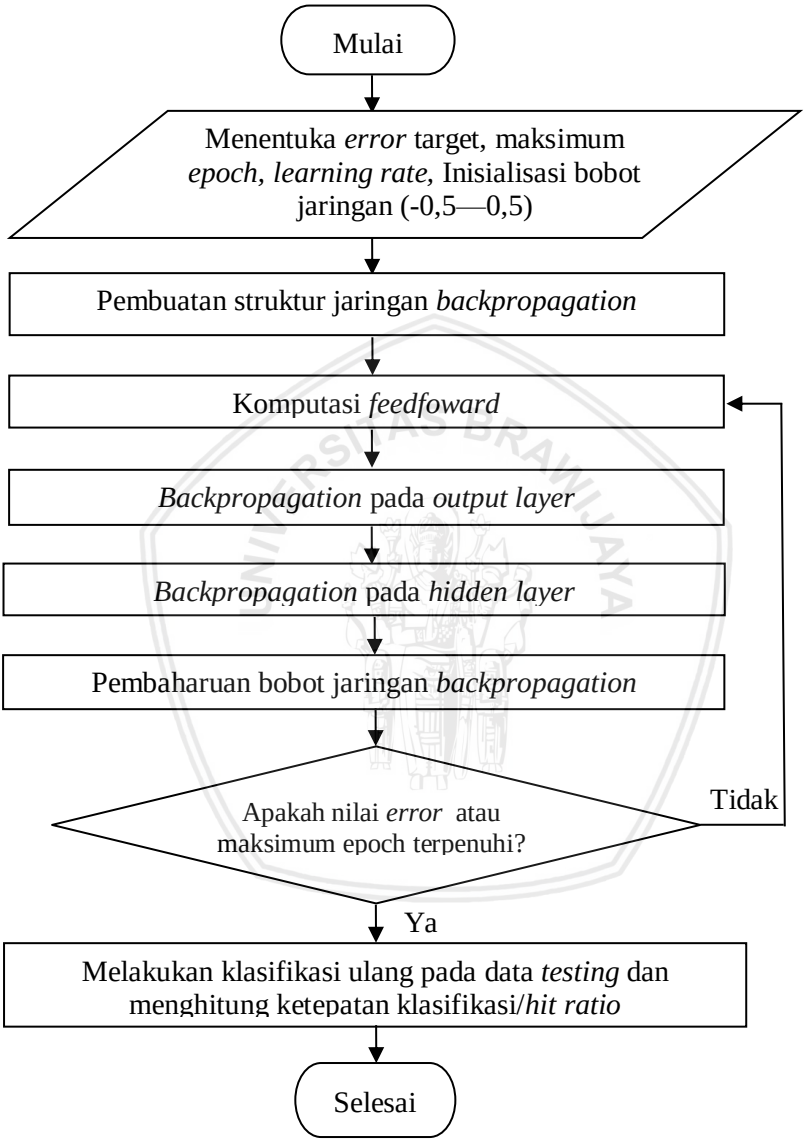
4. Mengevaluasi kebaikan model *backpropagation* dan *learning vector quantization* (LVQ) dengan menggunakan *hit ratio* ketepatan klasifikasi antara kedua metode tersebut.

Flowchart di bawah ini merupakan langkah-langkah metode penelitian.



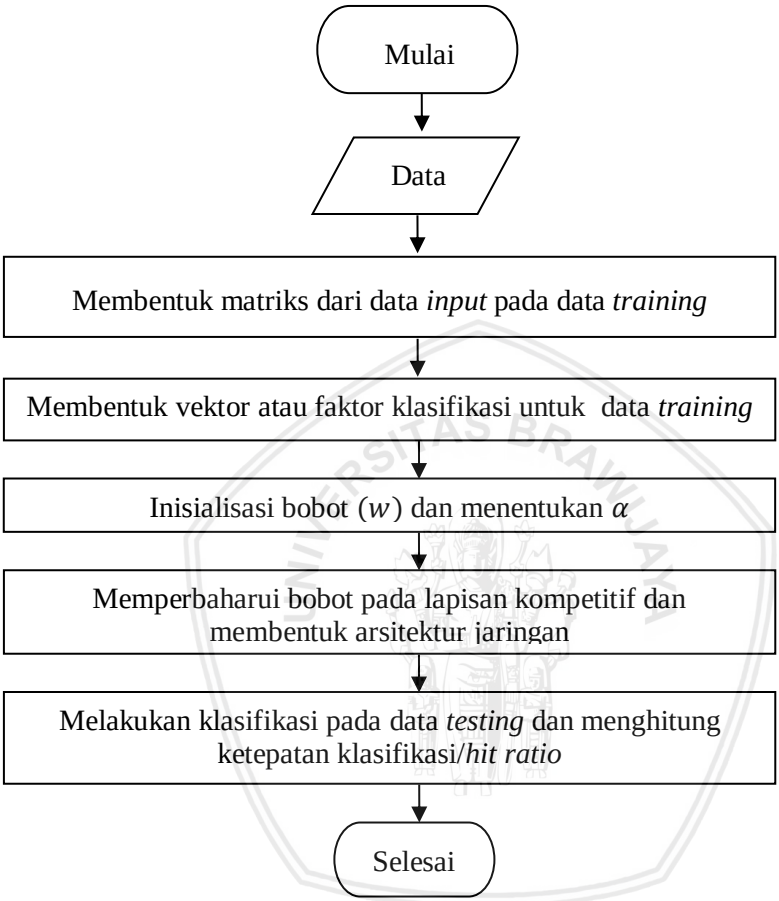
Gambar 3.1. *Flowchart* Metode Penelitian

Flowchart di bawah ini merupakan langkah-langkah metode *backpropagation.*



Gambar 3.2. *Flowchart* Metode *Backpropagation*

Flowchart di bawah ini merupakan langkah-langkah metode LVQ.



Gambar 3.3. *Flowchart* Metode LVQ

BAB IV
HASIL DAN PEMBAHASAN

4.1 Statistika Deskriptif

Analisis statistika deskriptif digunakan sebagai tambahan informasi mengenai keadaan faktor-faktor yang berpengaruh terhadap status gizi balita di Indonesia pada tahun 2016. Hasil analisis statistika deskriptif dapat digunakan sebagai gambaran umum mengenai keadaan status gizi balita di Indonesia. Hasil analisis statistika deskriptif terdapat pada Tabel 4.1 di bawah ini.

Tabel 4.1. Analisis Statistika Deskriptif

	Gizi Kurang	Balita Gemuk	IMD	Asi Ekslusif	Konsumsi Garam
Rata-rata	17,7%	4,8%	42,7%	33,3%	90,9%
Ragam	59,4%	10,2%	245,3%	435,9%	266,32%
Median	17,5%	4,1%	43,9%	29,1%	97,5%
Modus	14,2%	3,2%	41,8% 42% 50%	6,2% 16,7% 18,9% 25%	100%

Berdasarkan Tabel 4.1 dapat diperoleh informasi bahwa pada tahun 2016 sebagian besar kabupaten/kota mempunyai balita kurang gizi sebesar 17,7%, balita gemuk 4,8%, balita yang mendapat IMD 42,7%, balita yang mendapatkan asi ekslusif 33,3%, dan balita yang mengkonsumsi garam beryodium 90,9%.

Tingkat keragaman antar kabupaten/kota di Indonesia pada tahun 2016 pada variabel gizi kurang adalah sebesar 59,4%, balita gemuk 10,2%, balita yang mendapat IMD 245,3%, balita yang mendapat asi ekslusif sebesar 435,9%, dan balita yang mengkonsumsi garam beryodium sebesar 266,32%.

Sekitar 257 kabupaten/kota di Indonesia pada tahun 2016 mempunyai balita gizi kurang di bawah 17,5%, balita gemuk di bawah 4,1%, balita yang mendapat IMD di bawah 43,9%, balita yang mendapat asi ekslusif di bawah 29,1%, dan balita yang mengkonsumsi garam beryodium di bawah 97,5% sedangkan 257 kota/kabupaten yang lainnya mempunyai persentase diatas nilai-nilai tersebut.

Persentase gizi kurang yang paling banyak terjadi pada tahun 2016 adalah sebesar 14,2%, persentase balita gemuk terbanyak adalah 3,2%, persentase balita yang mendapat IMD terbanyak adalah 41,8%, 42%, dan 50%, persentase balita yang mendapatkan asi ekslusif terbanyak adalah 6,2%, 16,7%, 18,9%, dan 25%, dan persentase balita yang mengkonsumsi garam beryodium terbanyak adalah 100%.

4.2 Hasil Klasifikasi Metode *Backpropagation*

Klasifikasi dengan menggunakan metode *backpropagation* bergantung pada target *error*, maksimum epoch, serta *learning rate* yang digunakan. Pada penelitian ini target *error* yang digunakan untuk metode *backpropagation* sebesar 0,1 (Patterson, 1996), maksimum *epoch* sebanyak 1000 kali (Ramadhani dan Anis, 2009), bobot awal antara -0,5 sampai dengan 0,5 (Fausett, 1994), dan digunakan *learning rate* sebesar 0,01 (*default* R).

Klasifikasi metode *backpropagation* pada data *learning* menginisialisasikan target menjadi sebuah data biner. Dengan target kelas bernilai satu dan kelas lain bernilai nol. Setiap *neuron* keluaran akan menghasilkan nilai, nilai *ouput* yang paling besar akan bernilai satu, sedangkan *ouput* lainnya akan bernilai nol (Haryati, dkk, 2016). Pada penelitian ini digunakan empat target kelas klasifikasi, sehingga inisialisasi pada data *learning* dapat dilihat pada Tabel 4.2.

Tabel 4.2. Inisialisasi Data *Training* Berdasarkan Target

Target Kelas	Inisalisasi Data Biner
1	1 0 0 0
2	0 1 0 0
3	0 0 1 0
4	0 0 0 1

4.2.1. Penentuan *Neuron Hidden Layer Backropagation*

Arsitektur jaringan *backpropagation* disusun oleh banyaknya *neuron* pada *input layer*, *neuron hidden layer*, dan *neuron output layer*. Banyaknya *neuron input layer* sesuai dengan variabel prediktor yaitu lima *neuron* sedangkan banyaknya *neuron output layer*, sesuai dengan target *output* yang digunakan yaitu empat *neuron*.

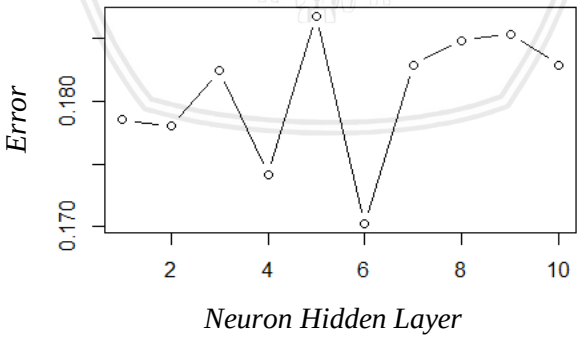
Banyaknya *neuron* pada *hidden layer* dapat ditentukan dengan menggunakan metode pengujian 10-*fold cross validation*. Nilai akurasi didapatkan dari perhitungan rata-rata *error* data *testing* setiap percobaan *neuron hidden layer*. Pada pengujian ini menggunakan satu

sampai dengan sepuluh *neuron hidden layer*. Tabel 4.3 merupakan nilai *error* yang diperoleh dengan metode 10-*fold cross validation* untuk setiap penggunaan *neuron hidden layer* pada data *training*

Tabel 4.3. Nilai *Error* Metode 10-*Fold Cross Validation*

Neuron Hidden Layer	*Error*
1	0,1785
2	0,1780
3	0,1824
4	0,1742
5	0,1868
6	0,1702
7	0,1829
8	0,1849
9	0,1854
10	0,1829

Berdasarkan Tabel 4.3, dapat diperoleh informasi bahwa dengan menggunakan enam *neuron* arsitektur jaringan yang dihasilkan akan mempunyai nilai *error* yang lebih kecil jika dibandingkan dengan arsitektur yang dibentuk oleh *neuron* lain. Gambar 4.1 merupakan nilai *error* metode 10-*fold cross validation* yang digambarkan dalam bentuk plot.



Gambar 4.1. Nilai *Error* Metode 10-*Fold Cross Validation*

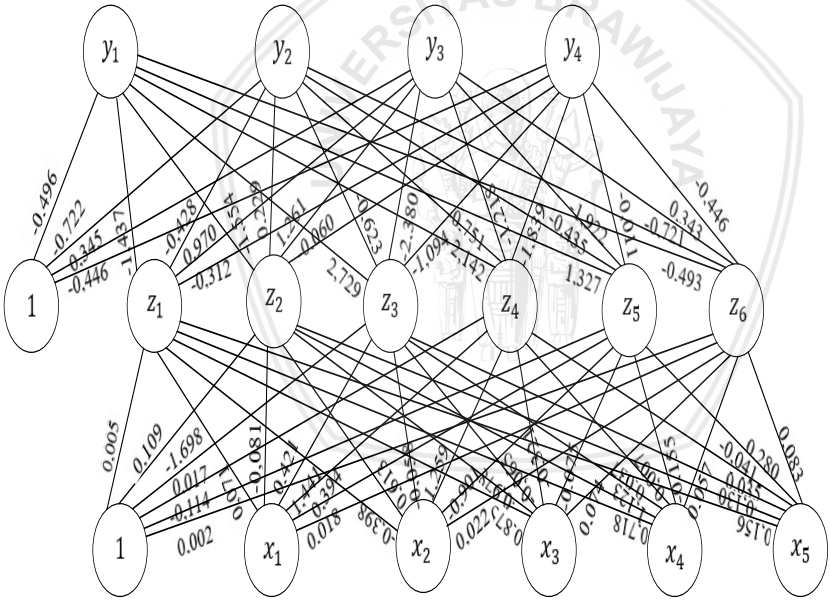
Pada Gambar 4.1 terlihat bahwa nilai *error* terkecil dihasilkan dengan menggunakan enam *neuron hidden layer*. Berdasarkan

pengujian menggunakan metode 10-*fold cross validation*, diperoleh kesimpulan bahwa jaringan dengan enam *neuron hidden layer* menghasilkan nilai *error* terkecil yaitu sebesar 0,1702.

Arsitektur jaringan *backpropagation* yang terbentuk adalah 5-6-4, yaitu dengan menggunakan lima *neuron* pada *input layer*, enam *neuron* pada *hidden layer*, dan empat *neuron* pada *output layer*.

4.2.2. Bobot Jaringan *Backpropagation*

Setelah perhitungan data *training*, maka akan diperoleh bobot optimum jaringan. Bobot optimum inilah yang akan digunakan untuk klasifikasi pada data *testing*. Gambar 4.2 merupakan bobot akhir jaringan *backpropagation* enam *neuron hidden layer* yang ditampilkan dalam bentuk arsitektur jaringan dengan menggunakan 80% data *training* dan 20% data *testing*. Nilai bobot secara lebih jelas dapat dilihat pada Lampiran 6.



Gambar 4.2. Arsitektur *Backpropagation* Dengan Bobot

4.2.3. Ketepatan Klasifikas Metode *Bacpropagation*

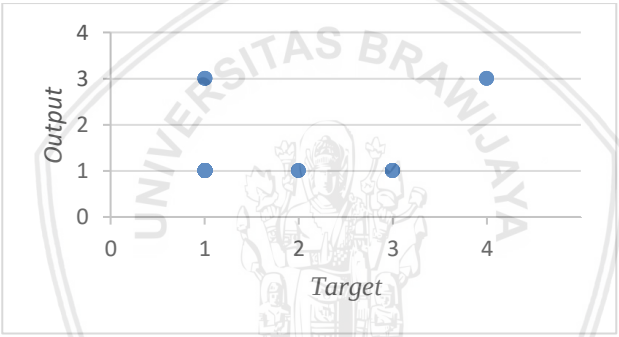
Ketepatan klasifikasi dapat ditentukan melalui nilai *hit ratio* pada data *testing*. Tabel 4.4 merupakan hasil klasifikasi pada data

testing sedangkan Tabel 4.5 merupakan hasil klasifikasi pada data *training*.

Tabel 4.4. Tabel Kontingensi Data *Testing* Metode *Backpropagation*

		Kelompok				Jumlah
		1	2	3	4	
Kelompok	1	81	0	4	0	85
	2	4	0	0	0	4
	3	12	0	0	0	12
	4	0	0	3	0	3

Ketepatan klasifikasi pada data *testing* dapat digambarkan dalam bentuk plot seperti pada Gambar 4.2 di bawah ini

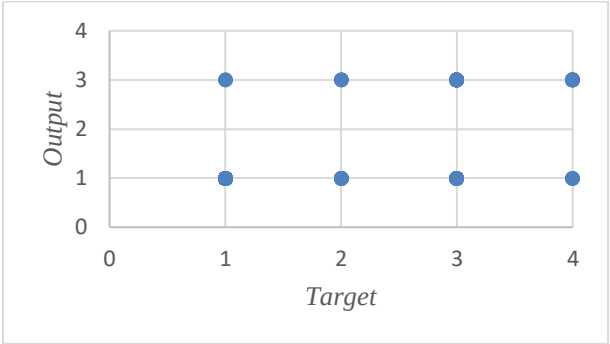


Gambar 4.3. *Target* dan *Output* Kelas Data *Testing* Metode *Backpropagation*

Tabel 4.5. Tabel Kontingensi Data *Training* Metode *Backpropagation*

		Kelompok				Jumlah
		1	2	3	4	
Kelompok	1	335	0	2	0	337
	2	11	0	3	0	14
	3	22	0	25	0	47
	4	3	0	9	0	12

Sedangkan ketepatan klasifikasi pada data *training* dapat digambarkan dalam bentuk plot seperti pada Gambar 4.4 di bawah ini



Gambar 4.4. *Target* dan *Output* Kelas Data *Training* Metode *Backpropagation*

Dengan menggunakan rumus pada persamaan 2.37 dapat diperoleh nilai *hit ratio* pada data *testing* dan data *training* seperti pada Tabel 4.6.

Tabel 4.6. Nilai *Hit Ratio* Metode *Backpropagation*

Neuron Hidden Layer	*Hit Ratio*	
	Data *training*	Data *testing*
6	87,8%	77,88%

Berdasarkan Tabel 4.6 dapat diperoleh informasi bahwa ketepatan klasifikasi metode *backpropagation* dengan enam *neuron hidden layer* menggunakan 80% data *training* dan 20% data *testing* adalah sebesar 77,88%. Ketepatan klasifikasi ditentukan melalui *hit ratio* dari data *testing* karena data *training* hanya digunakan untuk membentuk arsitektur jaringan sedangkan kebaikan dari arsitektur diuji dengan data *testing*.

4.3 Hasil Klasifikasi Metode LVQ

Learning vector quantization (LVQ) merupakan metode pembelajaran pada lapisan kompetitif yang terawasi, suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor *input* (Kusumadewi, 2004). Penelitian ini menggunakan persentase 80% data *training* dan 20% data *testing* sedangkan nilai α pada penelitian ini menggunkan *default* dari *software* R, yaitu 0,3.

4.2.1. Penentuan *Size Codebook* LVQ

Learning vector quantization (LVQ) merupakan jaringan syaraf tiruan yang mengimplementasikan konsep kompetisi. Penentuan banyaknya vektor bobot/*size codebook* pada jaringan LVQ dapat diperoleh dengan menggunakan persamaan 2.29 yaitu

$$size\ codebook = \min(round(0.4 \times ng \times \left(ng - 1 + \left(\frac{p}{2}\right), 0\right), n)$$

Dengan ng merupakan banyaknya target *output*, p merupakan banyaknya variabel prediktor serta n merupakan banyaknya data *training*, maka vektor bobot pada jaringan LVQ dapat diperoleh dengan perhitungan seperti di bawah ini

$$size\ codebook = \min(round(0.4 \times 4 \times \left(4 - 1 + \left(\frac{5}{2}\right), 0\right), 410)$$

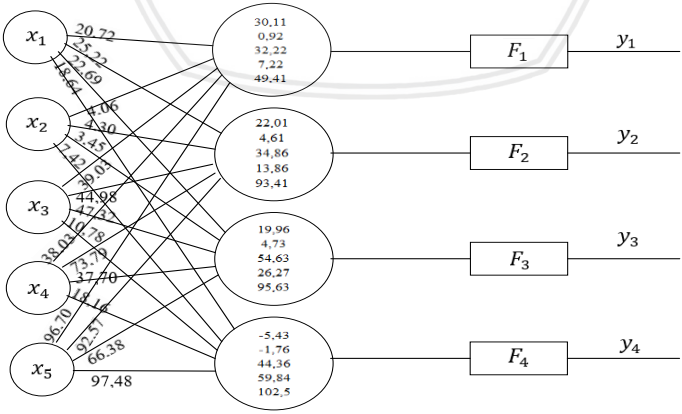
$$size\ codebook = \min(round(9{,}410))$$

$$size\ codebook = 9$$

Berrdasarkan perhitungan di atas, dapat diketahui bahwa banyaknya /*size codebook* yang sesuai ini adalah 9.

4.2.2. Bobot Jaringan LVQ

Setelah perhitungan data *training*, maka akan diperoleh bobot optimum jaringan. Bobot optimum inilah yang akan digunakan untuk klasifikasi pada data *testing*. Gambar 4.5 merupakan bobot akhir jaringan LVQ *size codebook* 9 yang ditampilkan dalam bentuk arsitektur jaringan dengan menggunakan 80% data *training* dan 20% data *testing*. Nilai bobot secara lebih jelas dapat dilihat pada Lampiran 7.



Gambar 4.5. Arsitektur Metode LVQ Dengan Bobot

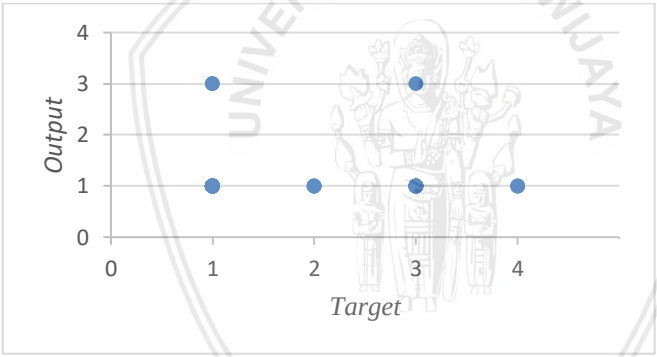
4.2.3. Ketepatan Klasifikasi Metode LVQ

Ketepatan klasifikasi dapat ditentukan melalui nilai *hit ratio* pada data *testing*. Tabel 4.7 merupakan hasil klasifikasi pada data *testing* sedangkan Tabel 4.8 merupakan hasil klasifikasi pada data *training* dengan menggunakan metode LVQ pada *size codebook* 9.

Tabel 4.7. Tabel Kontingensi Data *Testing* Metode LVQ

		Kelompok				Jumlah
		1	2	3	4	
Kelompok	1	83	0	2	0	85
	2	4	0	0	0	4
	3	11	0	1	0	12
	4	3	0	0	0	3

Ketepatan klasifikasi pada data *testing* dapat digambarkan dalam bentuk plot seperti pada Gambar 4.6 di bawah ini

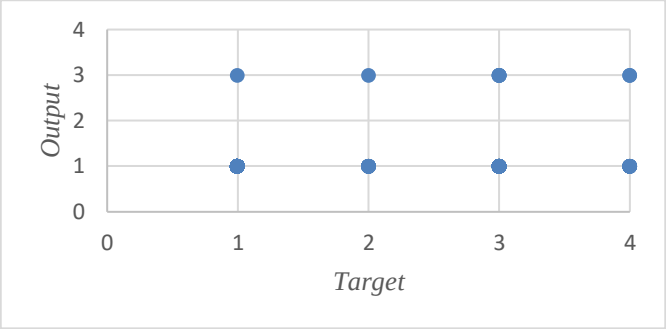


Gambar 4.6. *Target* dan *Output* Kelas Data *Testing* Metode LVQ

Tabel 4.8. Tabel Kontingensi Data *Training* Metode LVQ

		Kelompok				Jumlah
		1	2	3	4	
Kelompok	1	336	0	1	0	337
	2	13	0	1	0	14
	3	42	0	5	0	47
	4	10	0	2	0	12

Sedangkan ketepatan klasifikasi pada data *training* dapat digambarkan dalam bentuk plot seperti pada Gambar 4.7 di bawah ini



Gambar 4.7. *Target* dan *Output* Kelas Data *Training* Metode LVQ

Dengan menggunakan rumus pada persamaan 2.37 dapat diperoleh nilai *hit ratio* pada data *testing* dan data *training* seperti pada Tabel 4.9.

Tabel 4.9. Nilai *Hit Ratio* Metode LVQ

Size codebook	*Hit Ratio*	
	Data *training*	Data *testing*
9	83,17%	80,77%

Berdasarkan Tabel 4.9 dapat diperoleh informasi bahwa ketepatan klasifikasi metode LVQ dengan *size codebook* 9 menggunakan 80% data *training* dan 20% data *testing* adalah sebesar 80,77%. Ketepatan klasifikasi ditentukan melalui *hit ratio* dari data *testing* karena data *training* hanya digunakan untuk membentuk arsitektur jaringan sedangkan kebaikan dari arsitektur diuji dengan data *testing*.

4.4 Perbandingan Ketepatan Klasifikasi

Setelah melakukan perhitungan ketepatan klasifikasi metode *backpropagation* dan LVQ maka akan dilakukan perbandingan ketepatan klasifikasi kedua metode tersebut menggunakan nilai *hit ratio*. Tabel 4.10 merupakan perbandingan ketepatan klasifikasi metode *bacpropagation* dan metode LVQ dengan menggunakan 80% data *training* dan 20% data *testing*.

Tabel 4.10. Perbandingan Ketepatan Klasifikasi

Metode	*Neuron Hidden Layer*/*Size codebook*	*Hit Ratio*	
		Data *training*	Data *testing*
Backpropagation	6	87,8%	77,88%
LVQ	9	83,17%	80,77%

Bedasarkan hasil perbandingan nilai *hit ratio* pada Tabel 4.10, dapat diketahui bahwa ketepatan klasifikasi terbaik sebesar 80,77% diperoleh menggunakan metode LVQ dengan *size codebook* 9. Klasifikasi terbaik pada data status gizi balita di Indonesia tahun 2016 adalah dengan menggunakan metode LVQ, akan tetapi untuk kasus data yang lain mungkin akan berbeda.

Metode terbaik untuk klasifikasi bergantung pada data yang digunakan, inisialisasi awal pada bobot, dan fungsi aktivasi pada jaringan. Metode LVQ menggunakan fungsi aktivasi berupa jarak *Euclid* antara vektor bobot dan vektor *input* untuk menentukan jarak terdekat yang akan digunakan untuk klasifikasi, sedangkan metode *backpropagation* menggunakan fungsi aktivasi sigmoid biner untuk menentukan keluaran dari suatu *neuron*. Walaupun jaringan *backpropagation* mempunyai arsitektur yang lebih kompleks jika dibandingkan dengan metode LVQ, tetapi fungsi aktivasi jaringan LVQ yaitu jarak *Euclid* membuat jaringan tersebut menghasilkan klasifikasi secara lebih tepat pada data ini.

BAB V
PENUTUP

5.1 Kesimpulan

1. Berdasarkan hasil penelitian pada data status gizi balita kabupaten/kota di Indonesia tahun 2016, dapat diperoleh hasil bahwa anggota pengamatan yang tepat diklasifikasikan dengan menggunakan metode *backpropagation* enam *neuron hidden layer* adalah sebanyak 81 pengamatan dari data *testing* sedangkan anggota pengamatan yang tepat diklasifikasikan dengan menggunakan metode *Learning Vector Quantization* (LVQ) *size codebook* 9 adalah sebanyak 84 pengamatan dari data *testing*.
2. Ketepatan klasifikasi (*hit ratio)* yang diperoleh pada data status gizi balita kabupaten/kota di Indonesia tahun 2016 menggunakan metode *backpropagation* enam *neuron hidden layer* sebesar 77,88% sedangkan ketepatan klasifikasi metode *learning vector quantization* (LVQ) dengan *size codebook* 9 sebesar 80,77%.
3. Hasil penelitian secara keseluruhan menunjukkan bahwa metode *learning vector quantization* (LVQ) dengan *size codebook* 9 lebih baik digunakan untuk klasifikasi pada data status gizi balita kabupaten/kota di Indonesia tahun 2016 dibandingankan dengan metode *backpropagation* enam *neuron hidden layer*.

5.2 Saran

Setelah dilakukan penelitian disarankan agar pemerintah sebaiknya meningkatkan bantuan kesehatan bagi ibu dan balita yang disesuaikan berdasarkan kelompok status gizi balita setiap kabupaten/kota.

DAFTAR PUSTAKA

Demuth, H. dan Mark, B. (2000). *Neural Network Toolbox For Use With MATLAB User's Guie Version 4*. The Math Works, Inc.

Departemen Kesehatan RI. (2009). *Pedoman Pemantuan Status Gizi (PSG) dan Keluarga Sadar Gizi (KADARZI)*. Jakarta: Depkes.

Dianiati, A.N. (2013). Analisis Diskriminan Linier Robust Pada Pengklasifikasian Berat Bayi Baru Lahir. http://statistik.studentjournal.ub.ac.id/index.php/statistik/article/download/30/31. (Diakses Oktober 2018).

Direktorat Bina Gizi Masyarakat Depkes. (2008). *Pesan-pesan Tentang Inisiasi Menyusu Dini (IMD) dan Air Susu Ibu (ASI) Eksklusif Untuk Tenaga Kesehatan & Keluarga*. Jakarta: Depkes RI.

_____.(2017). *Pemantauan Status Gizi (PSG) dan Penjelasannya Tahun 2016*. Jakarta: Depkes RI.

Fausett, L. (1994). *Fundamentals of Neural Networks*. New Jersey: Prentice Hall.

Hair, J. F., Black, W. C., Babin, B. J., dan Anderson, R. E. (2010). *Multivariate Data Analysis: A Global Perspective (7th ed.).* New Jersey: Prentice Hall International.

Han, J. dan Kamber M. (2001). *Data Minning Concepts & techniques*. USA: Academic Press.

Haryati, D. F., Abdillah, G., dan Hadiana, A. I. (2016). Klasifikasi Jenis Batubara Menggunakan Jaringan Syaraf Tiruan Dengan Algoritma Backpropagation. *Jurnal SENTIKA 2016. ISSN: 2089-9815*.

Hasdianah, H. S. S. dan Peristyowati Y. (2014). *Gizi Pemanfaatan Gizi,Diet Dan Obesitas*. Yogyakarta: Nuha Medika.

Hermawan, A. (2006). *Jaringan Saraf Tiruan Teori dan Aplikasi*. Yogyakarta: Andi.

Hidayati, N. dan Warsito, B. (2010). Pediksi Terjangkitnya Penyakit Jantung Dengan Metode Learning Vector Quantization. *Jurnal Media Statistika Vol. 3(1): 21-22.*

Jhon, A. B. (2004). *Self Organizing Maps: Fundamentals*. www.cs.bham.ac.uk/~jxb/NN/l16/ . (Diakses Oktober 2018).

Johnson, R. A. dan Wichern. D. W. (2007). *Applied Multivariate Statistical Analysis. Sixth Edition.* New Jersey: Prentice Hall International. Inc.

Kementerian Kesehatan RI. (2011). *Keputusan Menteri Kesehatan Republik Indonesia Nomor 1995/MENKES/SK/XII/2010 Tentang Standar Antropometri Penilaian Status Gizi Anak*. Jakarta: Direktorat Bina Gizi Kementerian Kesehatan RI.

Khaidirmuhaj, (2009). *Klasifikasi Status Gizi*. http://khaidirmuhaj.blogspot.com/2009/02/gizi-.htm. (Diakses Oktober 2018).

Kohonen, T. (1990). Improved Versions of Learning Vector Quantization. *International Jont Conference on Neural Networks 1*. San Diego. CA.

Kusumadewi, S. (2004). *Membangun Jaringan Syaraf Tiruan Menggunakan Matlab dan Excel Link*. Yogyakarta: Graha Ilmu.

Patterson, D.W. (1996). *Artifical Neural Network Theory and Application*. Singapore: Prentice Hall.

Payam, R.., Tang, L., dan Liu, H. (2008). *Cross Validation*. USA: Arizona State University.

Prasetyo, E. (2012). *Data Mining Konsep dan Aplikasi Menggunakan Matlab*. Yogyakarta: Andi Offset.

Puspitaningrum, D. (2006). *Pengantar Jaringan Saraf Tiruan*. Yogyakarta: Andi.

Putri, N. R. (2012). *Learning Vector Quantization Dengan Logika Fuzzy untuk Pengenalan Wajah Berspektrum Cahaya Tampak Dengan Variasi Cahaya*. Skripsi . Universitas Indonesia.

Putro, D.S. (2011). *Pengembangan Jaringan Syaraf Tiruan dengan Metode SOM dan LVQ Fuzzy*. Tesis. Depok: Universitas Indonesia.

Refaeilzadeh, P., Tang, L., dan Liu, H. (2009). *Cross-Validation*. Springer: Encyclopedia of Database Systems.

Ramadhani, S. dan Anis U. Klasifikasi Penyakit Kencing Manis (Diabetes Melitus) Menggunakan Jaringan Syaraf Tiruan *Backpropagation. Jurnal Teknika, Volume 1 No.2 Tahun 2009. ISSN 2085-0859.*

Ramli, Y. D., dan Goejantoro R. (2013). Perbandingan Metode Klasifikasi Regresi Logistik Dengan Jaringan Saraf Tiruan (Studi Kasus: Pemilihan Jurusan Bahasa dan IPS pada SMAN 2 Samarinda Tahun Ajaran 2011/2012). *Jurnal Eksponensial Volume 4, Nomor 1, Mei 2013. ISSN 2085-7829*

Rcran. (2019). *Package Class*. https://cran.r-project.org/web/packages/class/class.pdf. (Diakses Janauari 2019).

Republik Indonesia. (1992). *Undang-Undang Nomor 23 tahun 1992 tentang kesehatan*. Jakarta: Sekertariat Negara.

_____. (2012). *Peraturan Pemerintah Republik Indonesia Nomor 33 Tahun 2012 Tentang Pemberian Air Susu Ibu Eksklusif*. Jakarta: Sekretariat Negara.

Rustiadi E., Saefulhakim S., dan Panuju D.R. (2011). *Perencanaan dan Pengembangan Wilayah*. Jakarta: Yayasan Pustaka Obor Rakyat.

Siang, J. J. (2005). *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan MATLAB.* Yogyakarta: Andi.

Sugiyono. (2007). *Metode Penelitian Kuantitatif Kualitatif dan R&D.* Bandung: Alfabeta.

Suhartono. (2007). *Feed Forward Neural Network Untuk Pemodelan* Runtun Waktu. Disertasi. Yogyakarta: Fakultas Mipa UGM

Supariasa dan Nyoman, I.D. (2002). *Penilaian Status Gizi*. Jakarta: EGC.

Supartini. (2004). *Buku Ajar Konsep Dasar Keperawatan Anak*. Jakarta: EGC

Ulfasari, R. (2010). *Perbandingan Performansi Jaringan Learning Vector Quantization (LVQ) Dan Radial Basis Function (RBF) Untuk Permasalahan Klasifikasi Penyakit Karies Gigi*. TESIS. Jurusan Matematika FMIPA ITS Surabaya.

Walpole, R.E. (1982). *Pengantar Statistika Edisi ke-3*. Jakarta: PT Gramedia.

Widodo, T.N. (2005). *Sistem Neuro Fuzzy*. Yogyakarta: Graha Ilmu.

WHO, (1997).

_____, (1948).

_____, (2000). *Obesity: Preventing and Managing the Global Epidemic*.Geneva: WHO Technical ReportSeries.

Wolley, N.G.A., Gunawan, S., dan Warouw, S. M. (2016). Perubahan Status Gizi Pada Anak Dengan Leukimia Limfoblastik Akut Selama Pengobatan. *Jurnal e-Clinic (eCl), 4(1)*.

Yitnosumarto, S. (1990) *Dasar-dasar Statistika*. Jakarta: PT Raja Grafindo Persada.

LAMPIRAN

Lampiran 1. Data Status Gizi Balita di Indonesia Tahun 2016

No.	Gizi Kurang (%)	Gemuk (%)	IMD (%)	Asi Ekslusif (%)	Konsumsi Garam (%)	Kategori
1.	22,2	2,7	55,1	36	100	Akut+Kronis
2.	7,7	3,9	71,4	12,6	98,1	Akut+Kronis
3.	20,8	4,2	66,7	36,2	95,4	Akut+Kronis
4.	6,3	6,7	65,2	18,9	64,3	Akut+Kronis
5.	25,2	2,8	63,6	12,2	32,2	Akut+Kronis
…	…	…	…	…	…	…
…	…	…	…	…	…	…
421.	7,8	4,8	13,8	13,9	88,6	Akut+Kronis
422.	19,3	7,1	29,7	15,8	95,4	Akut+Kronis
423.	9,6	1	59,2	62	93,5	Kronis
424.	14,5	3,5	39,3	55,8	99,6	Kronis
425.	11,2	3,2	16,7	33,6	90,7	Kronis
…	…	…	…	…	…	…
…	…	…	…	…	…	…
439.	14,2	2,6	36,9	43,4	95,3	Kronis
440.	13	7	51	32,1	98,1	Kronis
441.	9,5	5,7	38,3	37,9	99	Akut
442.	4,6	3,4	50	41,9	99,4	Akut
443.	10,1	5,1	38,7	56	99,7	Akut
…	…	…	…	…	…	…
…	…	…	…	…	…	…
498.	9,3	5,1	52,4	41,8	99,8	Akut
499.	14,8	2,3	60,1	17,1	86,2	Akut
500.	2,2	4,1	22,7	3,9	99	Baik
501.	1,3	4,8	57,9	39	100	Baik
502.	2,8	4	18,7	6,2	98	Baik
…	…	…	…	…	…	…
…	…	…	…	…	…	…
513.	0,3	3	24,4	8	100	Baik
514.	4	6	40,6	17,9	100	Baik

Lampiran 2. Perhitungan Manual Metode *Backpropagation*

Di bawah ini merupakan contoh klasifikasi menggunakan metode *backpropagation* secara manual dengan dua *neuron hidden layer*, dua *neuron input layer*, dan dua *neuron ouput layer*. Nilai $\alpha = 0{,}5$ serta maksimum *epoch* adalah dua kali. Nilai pada x_1 dan x_2 adalah sebagai berikut, data pertama dan kedua akan digunakan sebagai *training* sedangkan data ketiga dan keempat digunakan sebagai *testing*.

x_1	x_2	Kategori
0,25	0,4	Ya
0,2	0.05	Ya
0,5	0,15	Tidak
0,25	0,3	Ya

Dengan nilai target *output* adalah

y_1	y_2
1	0

Langkah 0
Inisialisasi semua bobot dengan bilangan acak kecil.

	z_1	z_2
x_1	0,15	0,1
x_2	-0,02	0,1
1	0,35	-0,25

	y_1	y_2
z_1	-0,1	-0,03
z_2	0,8	0,2
1	-0,03	0,01

x_1	x_2	Kategori
0,25	0,4	Ya

Langkah 1
Jika kondisi berhenti belum tercapai, lakukan langkah 2—9.

Lampiran 2. Lanjutan

Langkah 2
Untuk setiap pasang data pelatihan, lakukan langkah 3—8.
Langkah 3
Tiap *neuron* masukan menerima sinyal dan meneruskannya ke *neuron* tersembunyi di atasnya.

Langkah 4
Hitung keluaran *neuron* tersembunyi (z_j)

$$z\_net_j = v_{j0} + \sum_{i=1}^{2} x_i v_{ji}$$

$$z\_net_1 = v_{10} + x_1(v_{11}) + x_2(v_{12})$$
$$= 0{,}35 + 0{,}25(0{,}15) + 0{,}4(-0{,}02) = 0{,}3795$$
$$z\_net_2 = v_{20} + x_1(v_{21}) + x_2(v_{22})$$
$$= -0{,}25 + 0{,}25(0{,}11) + 0{,}4(0{,}1) = -0{,}185$$

Menghitung fungsi aktivasi dari *neuron* tersembunyi (z_j)

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

$$z_1 = \frac{1}{1 + e^{-0{,}3795}} = 0{,}59375 \;\; ; \;\; z_2 = \frac{1}{1 + e^{0{,}185}} = 0{,}45388$$

Langkah 5
Hitung keluaran *neuron* keluaran (y_k)

$$y\_net_k = w_{k0} + \sum_{j=1}^{2} z_j w_{kj}$$

$$y\_net_1 = w_{10} + z_1(w_{11}) + z_2(w_{12})$$
$$= -0{,}03 + 0{,}59375(-0{,}1) + 0{,}45388(0{,}8) = 0{,}27373$$
$$y\_net_2 = w_{20} + z_1(w_{21}) + z_2(w_{22})$$
$$= 0{,}01 + 0{,}59375(-0{,}03) + 0{,}45388(0{,}2) = 0{,}08296$$

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

$$y_1 = \frac{1}{1 + e^{-0{,}27373}} = 0{,}56801 \;\; ; \;\; y_2 = \frac{1}{1 + e^{-0{,}08296}} = 0{,}52073$$

Langkah 6
Hitung faktor δ di *neuron* keluaran (y_k)

Lampiran 2. Lanjutan

$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_k - y_k) y_k (1 - y_k)$
$\delta_1 = (t_1 - y_1) y_1 (1 - y_1)$
$= (1 - 0{,}56801) 0{,}56801 (1 - 0{,}56801) = 0{,}106$
$\delta_2 = (t_2 - y_2) y_2 (1 - y_2)$
$= (0 - 0{,}52073) 0{,}52073 (0 - 0{,}52073) = -0{,}12996$

Suku perubahan bobot w_{kj} (dengan $\alpha = 0{,}5$)

$$\Delta w_{kj} = \alpha \delta_k z_j = \alpha \delta z_j \ ; \ k = 1{,}2 \ ; \ j = 0{,}1{,}2$$

$\Delta w_{10} = 0{,}5(0{,}106)(1) = 0{,}053$
$\Delta w_{11} = 0{,}5(0{,}106)(0{,}59375) = 0{,}03147$
$\Delta w_{12} = 0{,}5(0{,}106)(0{,}45388) = 0{,}02406$

$\Delta w_{20} = 0{,}5(-0{,}12996)(1) = -0{,}06498$
$\Delta w_{21} = 0{,}5(-0{,}12996)(0{,}59375) = -0{,}03858$
$\Delta w_{22} = 0{,}5(-0{,}12996)(0{,}45388) = -0{,}02949$

Langkah 7
Hitung penjumlahan kesalahan dari unit tersembunyi

$$\delta\_net_j = \sum_{k=1}^{2} \delta_k w_{kj}$$

$\delta\_net_1 = \delta_1(w_{11}) + \delta_2(w_{21})$
$= \big(0{,}106(-0{,}1)\big) + \big(-0{,}12996(-0{,}03)\big) = -0{,}0067$
$\delta\_net_2 = \delta_1(w_{12}) + \delta_2(w_{22})$
$= \big(0{,}106(0{,}8)\big) + \big(-0{,}12996(0{,}2)\big) = 0{,}0588$
Hitung kesalahan di unit tersembunyi (δ_j)

$$\delta_j = \delta\_net_j f'(z\_net_j) = \delta\_net_j z_j (1 - z_j)$$

$\delta_1 = \delta\_net_1 z_1 (1 - z_1)$
$= (-0{,}0067)(0{,}5937525)(1 - 0{,}5937525) = -0{,}01616$
$\delta_2 = \delta\_net_2 z_2 (1 - z_2)$
$= (0{,}0588)(0{,}45388)(1 - 0{,}45388) = 0{,}01458$

Lampiran 2. Lanjutan

Suku perubahan bobot ke unit tersembunyi $\Delta v_{ji} = \alpha \delta_j x_i \; ; j = 1,2 \; ; i = 0,1,2$

	z_1	z_2
x_1	$\Delta v_{11} = (0,5)(-0,01616)(0,25) = -0,000202$	$\Delta v_{21} = (0,5)(0,01458)(0,25) = 0,001822$
x_2	$\Delta v_{12} = (0,5)(-0,01616)(0,4) = -0,000323$	$\Delta v_{22} = (0,5)(0,01458)(0,4) = 0,002915$
1	$\Delta v_{10} = (0,5)(-0,01616)(1) = -0,000808$	$\Delta v_{20} = (0,5)(0,01458)(1) = 0,007288$

Langkah 8
Hitung semua perubahan bobot

Perubahan bobot unit keluaran:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \quad (k = 1,2 \; ; j = 0,1,2)$$

$$w_{10}(baru) = w_{10}(lama) + \Delta w_{10} = -0.03 + (0,053) = 0,023$$

$$w_{11}(baru) = w_{11}(lama) + \Delta w_{11} = -0.1 + (0,031469) = -0,06853$$

$$w_{12}(baru) = w_{12}(lama) + \Delta w_{12} = 0,8 + (0,02406) = 0,8241$$

$$w_{20}(baru) = w_{20}(lama) + \Delta w_{20} = 0,01 + (-0,06498) = -0,05498$$

$$w_{21}(baru) = w_{21}(lama) + \Delta w_{21} = -0,03 + (-0,03858) = -0,06858$$

$$w_{22}(baru) = w_{22}(lama) + \Delta w_{22} = 0,2 + (-0,02949) = 0,1705$$

Lampiran 2. Lanjutan

Perubahan bobot unit tersembunyi:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \quad (j = 1,2 \ ; i = 0,1,2)$$

	z_1	z_2
x_1	$v_{11}(baru)$ $= 0,15 - 0,0002$ $= 0,149798$	$v_{21}(baru)$ $= 0,1 + 0,00182$ $= 0,10182$
x_2	$v_{12}(baru)$ $= -0,02 - 0,0003$ $= -0,02032$	$v_{22}(baru)$ $= 0,1 + 0,00291$ $= 0,10291$
1	$v_{10}(baru)$ $= 0,35 - 0,00081$ $= 0,34919$	$v_{23}(baru)$ $= -0,25 + 0,00728$ $= -0,2427$

Langkah 9

Menghitung nilai *error*

$$y_1 = 0,568008 \qquad y_2 = 0,520729$$

$$MSE = \frac{(1 - 0,568008)^2 + (0 - 0,520729)^2}{2}$$

$$MSE = 0,22889$$

Kondisi pemberhentian dapat ditentukan melalui banyaknya *epoch* yang telah dilakukan atau nilai *error* yang telah tercapai. Karena nilai *error* yang dihasilkan lebih besar dari *error* target, dan *epoch* yang dilakukan masih satu kali, maka kondisi berhenti belum tercapai sehingga iterasi dilakukan kembali dengan menggunakan bobot yang diperoleh iterasi terakhir.

Perubahan bobot yang diperoleh adalah sebagai berikut

x_1	x_2	Kategori
0,2	0,05	Ya

	z_1	z_2
x_1	0,14979	0,10182
x_2	-0,02032	0,10291
1	0,349192	-0,24271

Lampiran 2. Lanjutan

	y_1	y_2
z_1	-0,06853	-0,06858
z_2	0,8241	0,1705
1	0,023	-0,0549

Langkah 2
Untuk setiap pasang data pelatihan, lakukan langkah 3—8.
Langkah 3
Tiap *neuron* masukan menerima sinyal dan meneruskannya ke *neuron* tersembunyi di atasnya.
Langkah 4
Hitung keluaran *neuron* tersembunyi (z_j)

$$z\_net_j = v_{j0} + \sum_{i=1}^{2} x_i v_{ji}$$

$$z\_net_1 = v_{10} + x_1(v_{11}) + x_2(v_{12})$$
$$= 0{,}3492 + 0{,}2(0{,}14979) + 0{,}05(-0{,}02032) = 0{,}3781$$
$$z\_net_2 = v_{20} + x_1(v_{21}) + x_2(v_{22})$$
$$= -0{,}2427 + 0{,}2(0{,}10182) + 0{,}05(0{,}10291) = -0{,}2172$$

Menghitung fungsi aktivasi dari *neuron* tersembunyi (z_j)

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

$$z_1 = \frac{1}{1 + e^{-0{,}3781}} = 0{,}59342 \;;\; z_2 = \frac{1}{1 + e^{0{,}2172}} = 0{,}44591$$

Langkah 5
Hitung keluaran *neuron* keluaran (y_k)

$$y\_net_k = w_{k0} + \sum_{j=1}^{2} z_j w_{kj}$$

$$y\_net_1 = w_{10} + z_1(w_{11}) + z_2(w_{12})$$
$$= 0{,}023 + 0{,}59342(-0{,}06853) + 0{,}44591(0{,}824)$$
$$= 0{,}3498$$
$$y\_net_2 = w_{20} + z_1(w_{21}) + z_2(w_{22})$$
$$= -0{,}0549 + 0{,}59342(-0{,}06858) + 0{,}44591(0{,}1705)$$
$$= -0{,}01965$$

Lampiran 2. Lanjutan

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

$$y_1 = \frac{1}{1+e^{-0,3498}} = 0,586566 \;\; ; \;\; y_2 = \frac{1}{1+e^{0,01965}} = 0,495089$$

Langkah 6

Hitung faktor δ di *neuron* keluaran (y_k)

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

$\delta_1 = (t_1 - y_1) y_1 (1 - y_1)$

$= (1 - 0,586566) 0,586566 (1 - 0,586566) = 0,10026$

$\delta_2 = (t_2 - y_2) y_2 (1 - y_2)$

$= (0 - 0,495089) 0,495089 (0 - 0,495089) = -0,12376$

Suku perubahan bobot w_{kj} (dengan $\alpha = 0,5$)

$$\Delta w_{kj} = \alpha \delta_k z_j = \alpha \delta z_j \;\; ; \;\; k = 1,2 \;\; ; \;\; j = 0,1,2$$

$\Delta w_{10} = 0,5(0,10026)(1) = 0,05013$

$\Delta w_{11} = 0,5(0,10026)(0,59342) = 0,02975$

$\Delta w_{12} = 0,5(0,10026)(0,44591) = 0,022254$

$\Delta w_{20} = 0,5(-0,12376)(1) = -0,06188$

$\Delta w_{21} = 0,5(-0,12376)(0,59342) = -0,03672$

$\Delta w_{22} = 0,5(-0,12376)(0,44591) = -0,02759$

Langkah 7

Hitung penjumlahan kesalahan dari unit tersembunyi

$$\delta\_net_j = \sum_{k=1}^{2} \delta_k w_{kj}$$

$\delta\_net_1 = \delta_1(w_{11}) + \delta_2(w_{21})$

$= \big(0,10026(-0,06853)\big) + \big(-0,12376(-0,00686)\big)$

$= -0,001617$

$\delta\_net_2 = \delta_1(w_{12}) + \delta_2(w_{22})$

$= \big(0,10026(0,8241)\big) + \big(-0,12376(0,1705)\big)$

$= 0,061518$

Hitung kesalahan di unit tersembunyi (δ_j)

$$\delta_j = \delta\_net_j f'(z\_net_j) = \delta\_net_j z_j (1 - z_j)$$

$\delta_1 = \delta\_net_1 z_1 (1 - z_1)$

$= (-0,001617)(0,5934233)(1 - 0,5934233) = 0,00039$

Lampiran 2. Lanjutan

$\delta_2 = \delta\_net_2 z_2(1 - z_2)$
$= (0{,}061518)(0{,}44591)(1 - 0{,}44591) = 0{,}015199$

Suku perubahan bobot ke unit tersembunyi $\Delta v_{ji} = \alpha \delta_j x_i \; ; j = 1{,}2 \; ; i = 0{,}1{,}2$

	z_1	z_2
x_1	$\Delta v_{11} = (0{,}5)(0{,}00039)(0{,}2) = 0{,}000039$	$\Delta v_{21} = (0{,}5)(0{,}015199)(0{,}2) = 0{,}00152$
x_2	$\Delta v_{12} = (0{,}5)(0{,}00039)(0{,}05) = 0{,}00000975$	$\Delta v_{22} = (0{,}5)(0{,}015199)(0{,}05) = 0{,}00038$
1	$\Delta v_{10} = (0{,}5)(0{,}00039)(1) = 0{,}000195$	$\Delta v_{20} = (0{,}5)(0{,}015199)(1) = 0{,}007599$

Langkah 8
Hitung semua perubahan bobot

Perubahan bobot unit keluaran:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \quad (k = 1{,}2 \; ; j = 0{,}1{,}2)$$

$w_{10}(baru) = w_{10}(lama) + \Delta w_{10}$
$= 0.023 + (0{,}05013) = 0{,}07313$
$w_{11}(baru) = w_{11}(lama) + \Delta w_{11}$
$= -0{,}06853 + (0{,}029748) = -0{,}039878$
$w_{12}(baru) = w_{12}(lama) + \Delta w_{12}$
$= 0{,}8241 + (0{,}022354) = 0{,}846409$

$w_{20}(baru) = w_{20}(lama) + \Delta w_{20}$
$= -0{,}0549 + (-0{,}06188) = -0{,}11686$
$w_{21}(baru) = w_{21}(lama) + \Delta w_{21}$
$= -0{,}06858 + (-0{,}03672) = -0{,}1053$
$w_{22}(baru) = w_{22}(lama) + \Delta w_{22}$
$= 0{,}1705 + (-0{,}02759) = 0{,}1429$

Lampiran 2. Lanjutan

Perubahan bobot unit tersembunyi:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \quad (j = 1,2 \ ; i = 0,1,2)$$

	z_1	z_2
x_1	$v_{11}(baru)$ $= 0{,}14979 + 0{,}000039$ $= 0{,}149837$	$v_{21}(baru)$ $= 0{,}1018 + 0{,}00152$ $= 0{,}10334$
x_2	$v_{12}(baru)$ $= -0{,}0032 + 0{,}00000975$ $= -0{,}02031$	$v_{22}(baru)$ $= 0{,}1029 + 0{,}00038$ $= 0{,}103295$
1	$v_{10}(baru)$ $= 0{,}349192 + 0{,}000195$ $= 0{,}3494$	$v_{23}(baru)$ $= -0{,}2427 + 0{,}007599$ $= -0{,}2351$

Langkah 9

Menghitung nilai *error*

$$y_1 = 0{,}586566 \qquad y_2 = 0{,}495089$$

$$MSE = \frac{(1 - 0{,}586566)^2 + (0 - 0{,}495089)^2}{2}$$

$$MSE = 0{,}208$$

Kondisi pemberhentian dapat ditentukan melalui banyaknya *epoch* yang telah dilakukan atau nilai *error* yang telah tercapai. Nilai *error* yang dihasilkan lebih besar dari *error* target akan tetapi *epoch* yang dilakukan telah mencapai batas maksimum, sehingga kondisi berhenti tercapai dan iterasi dihentikan.

Klasifikasi pada data *testing* dapat dihitung menggunakan bobot akhir yang telah diperoleh, antara lain:

	z_1	z_2
x_1	0,149837	0,103342
x_2	-0,02031	0,103295
1	0,349387	-0,23511

	y_1	y_2
z_1	-0,03878	-0,105303
z_2	0,846409	0,142914
1	0,07313	-0,11686

Lampiran 2. Lanjutan

Untuk data ketiga, yaitu $x_1 = 0{,}5$ dan $x_2 = 0{,}15$
Hitung keluaran *neuron* tersembunyi (z_j)

$$z\_net_j = v_{j0} + \sum_{i=1}^{2} x_i v_{ji}$$

$$\begin{aligned} z\_net_1 &= v_{10} + x_1(v_{11}) + x_2(v_{12}) \\ &= 0{,}349387 + 0{,}5(0{,}149837) + 0{,}15(-0{,}02031) \\ &= 0{,}4213 \end{aligned}$$

$$\begin{aligned} z\_net_2 &= v_{20} + x_1(v_{21}) + x_2(v_{22}) \\ &= -0{,}23511 + 0{,}25(0{,}103342) + 0{,}4(0{,}103295) \\ &= -0{,}167946 \end{aligned}$$

Menghitung fungsi aktivasi dari *neuron* tersembunyi (z_j)

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

$$z_1 = \frac{1}{1 + e^{-0{,}4213}} = 0{,}6038 \;;\; z_2 = \frac{1}{1 + e^{0{,}167946}} = 0{,}45811$$

Hitung keluaran *neuron* keluaran (y_k)

$$y\_net_k = w_{k0} + \sum_{j=1}^{2} z_j w_{kj}$$

$$\begin{aligned} y\_net_1 &= w_{10} + z_1(w_{11}) + z_2(w_{12}) \\ &= 0{,}07313 + 0{,}6038(-0{,}03878) + 0{,}45811(-0{,}1053) \\ &= 0{,}437464 \end{aligned}$$

$$\begin{aligned} y\_net_2 &= w_{20} + z_1(w_{21}) + z_2(w_{22}) \\ &= -0{,}0549 + 0{,}6038(-0{,}1053) + 0{,}45811(0{,}14914) \\ &= -0{,}11497 \end{aligned}$$

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

$$y_1 = \frac{1}{1 + e^{-0{,}437464}} = 0{,}607655 \;;\; y_2 = \frac{1}{1 + e^{0{,}11497}} = 0{,}471289$$

Karena nilai y_1 lebih besar jika dibandingkan dengan nilai y_2 maka data ketiga tidaj tepat diklasifikasikan ke dalam kelas kedua.

Untuk data pertama, yaitu $x_1 = 0{,}25$ dan $x_2 = 0{,}3$
Hitung keluaran *neuron* tersembunyi (z_j)

Lampiran 2. Lanjutan

$$z\_net_j = v_{j0} + \sum_{i=1}^{2} x_i v_{ji}$$

$z\_net_1 = v_{10} + x_1(v_{11}) + x_2(v_{12})$
$= 0{,}349387 + 0{,}25(0{,}149837) + 0{,}3(-0{,}02031)$
$= 0{,}3808$
$= -0{,}23511 + 0{,}25(0{,}103342) + 0{,}3(0{,}103295)$
$= -0{,}1783$

Menghitung fungsi aktivasi dari *neuron* tersembunyi (z_j)

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

$$z_1 = \frac{1}{1 + e^{-0{,}3808}} = 0{,}594 \;;\; z_2 = \frac{1}{1 + e^{0{,}1783}} = 0{,}456$$

Hitung keluaran *neuron* keluaran (y_k)

$$y\_net_k = w_{k0} + \sum_{j=1}^{2} z_j w_{kj}$$

$y\_net_1 = w_{10} + z_1(w_{11}) + z_2(w_{12})$
$= 0{,}07313 + 0{,}594(-0{,}03878) + 0{,}456(-0{,}1053)$
$= 0{,}4357$
$y\_net_2 = w_{20} + z_1(w_{21}) + z_2(w_{22})$
$= -0{,}0549 + 0{,}594(-0{,}1053) + 0{,}456(0{,}14914)$
$= -0{,}1$

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

$$y_1 = \frac{1}{1 + e^{-0{,}4357}} = 0{,}607227 \;;\; y_2 = \frac{1}{1 + e^{0{,}1}} = 0{,}4749$$

Karena nilai y_1 lebih besar jika dibandingkan dengan nilai y_2 maka data keempat tepat diklasifikasikan ke dalam kelas pertama.

Lampiran 3. Perhitungan Manual Metode LVQ

Perhitungan data *training*

$$\{s^{(q)}; t^{(q)}\}, \qquad q = 1,2, \dots, Q$$

Di mana

$s^{(q)} =$ *training vector*

$t^{(q)} =$ *target output vector*

Dengan menggunakan dua kelas sebagai *target output*, maka vektor $s^{(q)}$ dan vertor $t^{(q)}$ yang terbentuk untuk setiap kelompok adalah sebagai berikut:

Kelas pertama

$$s^{(1)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \qquad t^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$s^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad t^{(2)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Kelas dua

$$s^{(3)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad t^{(3)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$s^{(4)} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad t^{(4)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Langkah 1

Menentukan bobot yang akan menghubungkan *neuron* satu dan dua ke *neuron output* satu dan menghubungkan *neuron* dua dan tiga ke *neuron output* dua.

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Langkah 2

Inisialisasi setiap kolom $\boldsymbol{W}^{(1)}$ secara acak. Misalnya:

$$w_{.1}^{(1)} = \begin{bmatrix} -0{,}543 \\ 0{,}840 \end{bmatrix} \qquad w_{.2}^{(1)} = \begin{bmatrix} -0{,}969 \\ -0{,}249 \end{bmatrix}$$

$$w_{.3}^{(1)} = \begin{bmatrix} 0{,}997 \\ 0{,}094 \end{bmatrix} \qquad w_{.4}^{(1)} = \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix}$$

Langkah 3

Secara acak memilih vektor *training* untuk perhitungan pada jaringan. Misalnya mengambil $s^{(3)}$

Lampiran 3. Lanjutan

$$s^{(3)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad t^{(3)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Langkah 4

Menghitung jarak *Euclid* dari vektor *input* ke vektor bobot dengan rumus

$$n^{(1)} = \begin{bmatrix} \left\| s^{(3)} - \boldsymbol{W}_1^{(1)} \right\| \\ \left\| s^{(3)} - \boldsymbol{W}_2^{(1)} \right\| \\ \left\| s^{(3)} - \boldsymbol{W}_3^{(1)} \right\| \\ \left\| s^{(3)} - \boldsymbol{W}_4^{(1)} \right\| \end{bmatrix}$$

$$\left\| s^{(3)} - \boldsymbol{W}_1^{(1)} \right\| = \sqrt{(1+0{,}543)^2 + (-1-0{,}840)^2} = 2{,}40$$
$$\left\| s^{(3)} - \boldsymbol{W}_2^{(1)} \right\| = \sqrt{(1+0{,}969)^2 + (-1+0{,}249)^2} = 2{,}11$$
$$\left\| s^{(3)} - \boldsymbol{W}_3^{(1)} \right\| = \sqrt{(1-0{,}997)^2 + (-1-0{,}094)^2} = 1{,}09$$
$$\left\| s^{(3)} - \boldsymbol{W}_4^{(1)} \right\| = \sqrt{(1-0{,}456)^2 + (-1-0{,}954)^2} = 2{,}04$$

$$n^{(1)} = \begin{bmatrix} 2{,}40 \\ 2{,}11 \\ 1{,}09 \\ 2{,}04 \end{bmatrix}$$

Langkah 5

Mancari *neuron* pemenang dalam kasus ini jelas bahwa *neuron hidden* yang ketiga mempunyai jarak vektor bobot yang dekat dengan vektor *input* $s^{(3)}$, dan didapatkan *neuron* pemenang seperti berikut:

$$a^{(1)} = compet\left(n^{(1)}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Lampiran 3. Lanjutan

Karena $a^{(2)}$ sama seperti $t^{(3)}$, dapat disimpulkan bahwa $s^{(3)}$ diklasifikasikan tepat pada kelas kedua.
Sehingga vektor bobot *hidden* $w_{.3}^{(1)}$ dipindahkan mendekati $s^{(3)}$:

$$\Delta w_{.3}^{(1)} = \alpha(s^{(3)} - w_{.3}^{(1)})$$
$$\Delta w_{.3}^{(1)} = \begin{bmatrix} 0{,}997 \\ 0{,}094 \end{bmatrix} + 0{,}5\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0{,}997 \\ 0{,}094 \end{bmatrix}\right) = \begin{bmatrix} 0{,}998 \\ -0{,}453 \end{bmatrix}$$

Iterasi pertama untuk $s^{(3)}$ telah dilakukan lalu dilanjutkan iterasi berikutnya. Dengan cara yang sama seperti di atas didapatkan jarak *Euclid*, $a^{(1)}, a^{(2)}$, dan bobot baru sebagai berikut:

Jaka *Euclid*

Iterasi	1	2	3	4	5	6	7	8
$\|s^{(3)} - \boldsymbol{W}_1^{(1)}\|$	2,401	2,401	2,401	2,401	2,401	2,401	2,401	2,401
$\|s^{(3)} - \boldsymbol{W}_2^{(1)}\|$	2,107	2,107	2,107	2,107	2,107	2,107	2,107	2,107
$\|s^{(3)} - \boldsymbol{W}_3^{(1)}\|$	1,094	0,547	0,274	0,137	0,068	0,034	0,017	0,009
$\|s^{(3)} - \boldsymbol{W}_4^{(1)}\|$	2,028	2,028	2,028	2,028	2,028	2,028	2,028	2,028

Di bawah ini merupakan tabel vektor pemenang setiap iterasi

	1	2	3	4	5	6	7	8
$a^{(1)}$	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0

Kemudian menghitung nilai vektor $a^{(2)}$ dengan rumus

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)}$$

Diperoleh hasil seperti pada tabel di bawah ini

	1	2	3	4	5	6	7	8
$a^{(2)}$	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1

Lampiran 3. Lanjutan

Karena nilai vektor $a^{(2)}$ pada setiap iterasi sama seperti $t^{(3)}$, dapat disimpulkan bahwa $s^{(3)}$ diklasifikasikan tepat pada kelas kedua pada setiap iterasi dan vektor bobot *hidden* $w_{.3}^{(1)}$ dipindahkan mendekati $s^{(3)}$.

$$\Delta w_{.3}^{(1)} = \alpha(s^{(3)} - w_{.3}^{(1)})$$

Diperoleh bobot baru untuk $w_{.3}^{(1)}$ seperti pada tabel di bawah ini

$w_{.3}^{(1)}$	1	2	3	4	5	6	7	8
	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	-0,45	-0,73	-0,86	-0,93	-0,97	-0,98	-0,99	-1,00

Dengan iterasi pada data *training* sebanyak 8 maka diperoleh kesimpulkan bahwa $w_{.3}^{(1)}$ dipindahkan mendekati $s^{(3)}$.

Perhitungan dilanjutkan ke data *training* berikutnya.
Langkah 2
Inisialisasi setiap kolom $\boldsymbol{W}^{(1)}$ secara acak.

$$w_{.1}^{(1)} = \begin{bmatrix} -0{,}543 \\ 0{,}840 \end{bmatrix} \qquad w_{.2}^{(1)} = \begin{bmatrix} -0{,}969 \\ -0{,}249 \end{bmatrix}$$

$$w_{.3}^{(1)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad w_{.4}^{(1)} = \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix}$$

Langkah 3
Secara acak memilih vektor *training* untuk perhitungan pada jaringan. Misalnya mengambil $s^{(1)}$

$$s^{(1)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \qquad t^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Langkah 4
Menghitung jarak *Euclid* dari vektor *input* ke vektor bobot dengan rumus

Lampiran 3. Lanjutan

$$n^{(1)} = \begin{bmatrix} \left\| s^{(1)} - \boldsymbol{W}_1^{(1)} \right\| \\ \left\| s^{(1)} - \boldsymbol{W}_2^{(1)} \right\| \\ \left\| s^{(1)} - \boldsymbol{W}_3^{(1)} \right\| \\ \left\| s^{(1)} - \boldsymbol{W}_4^{(1)} \right\| \end{bmatrix}$$

$$\left\| s^{(1)} - \boldsymbol{W}_1^{(1)} \right\| = \sqrt{(-1+0{,}543)^2 + (-1-0{,}840)^2} = 1{,}89$$
$$\left\| s^{(1)} - \boldsymbol{W}_2^{(1)} \right\| = \sqrt{(-1+0{,}969)^2 + (-1+0{,}249)^2} = 0{,}75$$
$$\left\| s^{(1)} - \boldsymbol{W}_3^{(1)} \right\| = \sqrt{(-1-1)^2 + (-1+1)^2} = 2$$
$$\left\| s^{(1)} - \boldsymbol{W}_4^{(1)} \right\| = \sqrt{(-1-0{,}456)^2 + (-1-0{,}954)^2} = 2{,}44$$

$$n^{(1)} = \begin{bmatrix} 1{,}89 \\ 0{,}75 \\ 2 \\ 2{,}44 \end{bmatrix}$$

Langkah 5

Mancari *neuron* pemenang dalam kasus ini jelas bahwa *neuron hidden* yang kedua mempunyai jarak vektor bobot yang dekat dengan vektor *input* $s^{(1)}$, dan didapatkan *neuron* pemenang seperti berikut:

$$a^{(1)} = compet\left(n^{(1)}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Karena $a^{(2)}$ sama seperti $t^{(1)}$, dapat disimpulkan bahwa $s^{(1)}$ diklasifikasikan tepat pada kelas pertama.

Sehingga vektor bobot *hidden* $w_{.2}^{(1)}$ dipindahkan mendekati $s^{(1)}$:

$$\Delta w_{.2}^{(1)} = \alpha(s^{(1)} - w_{.2}^{(1)})$$

Lampiran 3. Lanjutan

$$\Delta w_{.2}^{(1)} = \begin{bmatrix} -0{,}969 \\ -0{,}249 \end{bmatrix} + 0{,}5\left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} - \begin{bmatrix} -0{,}969 \\ -0{,}249 \end{bmatrix}\right) = \begin{bmatrix} -0{,}98 \\ -0{,}62 \end{bmatrix}$$

Iterasi pertama untuk $s^{(1)}$ telah dilakukan lalu dilanjutkan iterasi berikutnya. Dengan cara yang sama seperti di atas didapatkan jarak *Euclid*, $a^{(1)}, a^{(2)}$, dan bobot baru sebagai berikut:

Jarak *Euclid*

Iterasi	1	2	3	4	5	6	7	8
$\|s^{(1)} - \boldsymbol{W}_1^{(1)}\|$	1,896	1,896	1,896	1,896	1,896	1,896	1,896	1,896
$\|s^{(1)} - \boldsymbol{W}_2^{(1)}\|$	0,752	0,376	0,188	0,094	0,047	0,023	0,012	0,006
$\|s^{(1)} - \boldsymbol{W}_3^{(1)}\|$	2	2	2	2	2	2	2	2
$\|s^{(1)} - \boldsymbol{W}_4^{(1)}\|$	2,437	2,437	2,437	2,437	2,437	2,437	2,437	2,437

Di bawah ini merupakan tabel vektor pemenang setiap iterasi

	1	2	3	4	5	6	7	8
$a^{(1)}$	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

Kemudian menghitung nilai vektor $a^{(2)}$ dengan rumus

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)}$$

Diperoleh hasil seperti pada tabel di bawah ini

	1	2	3	4	5	6	7	8
$a^{(2)}$	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0

Karena nilai vektor $a^{(2)}$ pada setiap iterasi sama seperti $t^{(1)}$, dapat disimpulkan bahwa $s^{(1)}$ diklasifikasikan tepat pada kelas pertama pada setiap iterasi dan vektor bobot *hidden* $w_{.2}^{(1)}$ dipindahkan mendekati $s^{(1)}$.

$$\Delta w_{.2}^{(1)} = \alpha(s^{(1)} - w_{.2}^{(1)})$$

Lampiran 3. Lanjutan

Diperoleh bobot baru untuk $w_{.2}^{(1)}$ seperti pada tabel di bawah ini

$w_{.2}^{(1)}$	1	2	3	4	5	6	7	8
	-0,98	-0,99	-1,00	-1,00	-1,00	-1,00	-1,00	-1,00
	-0,62	-0,81	-0,91	-0,95	-0,98	-0,99	-0,99	-1,00

Dengan iterasi pada data *training* sebanyak 8 maka diperoleh kesimpulkan bahwa $w_{.2}^{(1)}$ dipindahkan mendekati $s^{(1)}$.

Perhitungan dilanjutkan ke data *training* berikutnya.
Langkah 2
Inisialisasi setiap kolom $\boldsymbol{W}^{(1)}$ secara acak.

$$w_{.1}^{(1)} = \begin{bmatrix} -0{,}543 \\ 0{,}840 \end{bmatrix} \qquad w_{.2}^{(1)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$w_{.3}^{(1)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad w_{.4}^{(1)} = \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix}$$

Langkah 3
Secara acak memilih vektor *training* untuk perhitungan pada jaringan. Misalnya mengambil $s^{(2)}$

$$s^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad t^{(2)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Langkah 4
Menghitung jarak *Euclid* dari vektor *input* ke vektor bobot dengan rumus

$$n^{(1)} = \begin{bmatrix} \left\| s^{(2)} - \boldsymbol{W}_1^{(1)} \right\| \\ \left\| s^{(2)} - \boldsymbol{W}_2^{(1)} \right\| \\ \left\| s^{(2)} - \boldsymbol{W}_3^{(1)} \right\| \\ \left\| s^{(2)} - \boldsymbol{W}_4^{(1)} \right\| \end{bmatrix}$$

$$\left\| s^{(2)} - \boldsymbol{W}_1^{(1)} \right\| = \sqrt{(1+0{,}543)^2 + (1-0{,}84)^2} = 1{,}55$$

$$\left\| s^{(2)} - \boldsymbol{W}_2^{(1)} \right\| = \sqrt{(1+1)^2 + (1+1)^2} = 2{,}828$$

$$\left\| s^{(2)} - \boldsymbol{W}_3^{(1)} \right\| = \sqrt{(1-1)^2 + (1+1)^2} = 2$$

Lampiran 3. Lanjutan

$$\left\| s^{(2)} - \boldsymbol{W}_4^{(1)} \right\| = \sqrt{(1-0{,}456)^2 + (1-0{,}954)^2} = 0{,}55$$

$$n^{(1)} = \begin{bmatrix} 1{,}55 \\ 2{,}828 \\ 2 \\ 0{,}55 \end{bmatrix}$$

Langkah 5
Mancari *neuron* pemenang dalam kasus ini jelas bahwa *neuron hidden* yang keempat mempunyai jarak vektor bobot yang dekat dengan vektor *input* $s^{(2)}$, dan didapatkan *neuron* pemenang seperti berikut:

$$a^{(1)} = compet\left(n^{(1)}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Karena $a^{(2)}$ berbeda dengan$t^{(2)}$,sehingga dapat disimpulkan bahwa $s^{(2)}$ diklasifikasikan salah pada kelas pertama.
Sehingga vektor bobot *hidden* $w_{.4}^{(1)}$ dipindahkan menjauhi $s^{(2)}$:

$$\Delta w_{.4}^{(1)} = -\alpha(s^{(2)} - w_{.4}^{(1)})$$

$$\Delta w_{.4}^{(1)} = \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix} - 0{,}5\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix}\right) = \begin{bmatrix} 0{,}18 \\ 0{,}93 \end{bmatrix}$$

Iterasi pertama untuk $s^{(1)}$ telah dilakukan lalu dilanjutkan iterasi berikutnya. Dengan cara yang sama seperti di atas didapatkan jarak *Euclid*, $a^{(1)}, a^{(2)}$, dan bobot baru sebagai berikut:

Lampiran 3. Lanjutan

Jarak *Euclid*

Iterasi	1	2	3	4	5	6	7	8
$\|\|s^{(1)} - \boldsymbol{W}_1^{(1)}\|\|$	1,551	1,551	1,551	1,551	0,776	0,388	0,194	0,097
$\|\|s^{(1)} - \boldsymbol{W}_2^{(1)}\|\|$	2,828	2,828	2,828	2,828	2,828	2,828	2,828	2,828
$\|\|s^{(1)} - \boldsymbol{W}_3^{(1)}\|\|$	2	2	2	2	2	2	2	2
$\|\|s^{(1)} - \boldsymbol{W}_4^{(1)}\|\|$	0,546	0,819	1,228	1,843	1,843	1,843	1,843	1,843

Iterasi	9	10	11	12
$\|\|s^{(1)} - \boldsymbol{W}_1^{(1)}\|\|$	0,048	0,024	0,012	1,551
$\|\|s^{(1)} - \boldsymbol{W}_2^{(1)}\|\|$	2,828	2,828	2,828	2,828
$\|\|s^{(1)} - \boldsymbol{W}_3^{(1)}\|\|$	2	2	2	2
$\|\|s^{(1)} - \boldsymbol{W}_4^{(1)}\|\|$	1,843	1,843	1,843	1,843

Di bawah ini merupakan tabel vektor pemenang setiap iterasi

	1	2	3	4	5	6	7	8	9	10	11	12
$a^{(1)}$	0	0	0	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	0	0	0	0	0	0	0	0	0

Kemudian menghitung nilai vektor $a^{(2)}$ dengan rumus

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)}$$

Diperoleh hasil seperti pada tabel di bawah ini

	1	2	3	4	5	6	7	8	9	10	11	12
$a^{(2)}$	0	0	0	1	1	1	1	1	1	1	1	1
	1	1	1	0	0	0	0	0	0	0	0	0

Karena nilai vektot $a^{(2)}$ pada iterasi pertama sampai ketiga berbeda dengan $t^{(2)}$, dapat disimpulkan bahwa $s^{(2)}$ pada iterasi tersebut diklasifikasikan tidak tepat pada kelas pertama pada setiap iterasi dan vektor bobot *hidden* $w_{.4}^{(1)}$ dipindahkan menjauhi $s^{(2)}$.

$$\Delta w_{.4}^{(1)} = -\alpha(s^{(2)} - w_{.4}^{(1)})$$

Lampiran 3. Lanjutan

Sedangkan $a^{(2)}$ pada iterasi keempat sampai iterasi keduabelas sama seperti $t^{(2)}$, hal ini berarti bahwa $s^{(2)}$ pada iterasi tersebut diklasifikasikan dengan benar menurut kelas pertama sehingga pada iterasi tersebut juga vektor bobot hidden $w_{.1}^{(1)}$ dipindahkan mendekati $s^{(2)}$

$$\Delta w_{.1}^{(1)} = -\alpha(s^{(2)} - w_{.1}^{(1)})$$

Diperoleh bobot baru untuk $w_{.1}^{(1)}$ dan $w_{.4}^{(1)}$ seperti pada tabel di bawah ini

$w_{.4}^{(1)}$	1	2	3
	0,18	-0,22	-0,84
	0,93	0,90	0,84

$w_{.4}^{(1)}$	4	5	6	7	8	9	10	11	12
	0,23	0,61	0,81	0,90	0,95	0,98	0,99	0,99	1
	0,92	0,96	0,98	0,99	1	1	1	1	1

Dengan iterasi pada data *training* sebanyak 12 maka diperoleh kesimpulkan bahwa $w_{.1}^{(1)}$ dipindahkan mendekati $s^{(2)}$.

Perhitungan dilanjutkan ke data *training* berikutnya.
Langkah 2
Inisialisasi setiap kolom $\boldsymbol{W}^{(1)}$ secara acak.

$$w_{.1}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad w_{.2}^{(1)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$w_{.3}^{(1)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad w_{.4}^{(1)} = \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix}$$

Langkah 3
Secara acak memilih vektor *training* untuk perhitungan pada jaringan. Karena pada perhitungan ini hanya data *training* keempat yang belum dimasukkan ke dalam perhitungan pada jaringan, maka $s^{(4)}$ dipilih

$$s^{(4)} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad t^{(4)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Lampiran 3. Lanjutan

Langkah 4

Menghitung jarak *Euclid* dari vektor *input* ke vektor bobot dengan rumus

$$n^{(1)} = \begin{bmatrix} \left\| s^{(4)} - \boldsymbol{W}_1^{(1)} \right\| \\ \left\| s^{(4)} - \boldsymbol{W}_2^{(1)} \right\| \\ \left\| s^{(4)} - \boldsymbol{W}_3^{(1)} \right\| \\ \left\| s^{(4)} - \boldsymbol{W}_4^{(1)} \right\| \end{bmatrix}$$

$$\left\| s^{(4)} - \boldsymbol{W}_1^{(1)} \right\| = \sqrt{(-1-1)^2 + (1-1)^2} = 2$$
$$\left\| s^{(4)} - \boldsymbol{W}_2^{(1)} \right\| = \sqrt{(-1+1)^2 + (1+1)^2} = 1{,}99$$
$$\left\| s^{(4)} - \boldsymbol{W}_3^{(1)} \right\| = \sqrt{(-1-1)^2 + (1+1)^2} = 2{,}83$$
$$\left\| s^{(4)} - \boldsymbol{W}_4^{(1)} \right\| = \sqrt{(-1-0{,}456)^2 + (1-0{,}954)^2} = 1{,}46$$

$$n^{(1)} = \begin{bmatrix} 2 \\ 1{,}99 \\ 2{,}83 \\ 1{,}46 \end{bmatrix}$$

Langkah 5

Mancari *neuron* pemenang dalam kasus ini jelas bahwa *neuron hidden* yang keempat mempunyai jarak vektor bobot yang dekat dengan vektor *input* $s^{(4)}$, dan didapatkan *neuron* pemenang seperti berikut:

$$a^{(1)} = compet\left(n^{(1)}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Karena $a^{(2)}$ sama seperti $t^{(4)}$, dapat disimpulkan bahwa $s^{(4)}$ diklasifikasikan tepat pada kelas kedua.

Sehingga vektor bobot *hidden* $w_{.4}^{(1)}$ dipindahkan mendekati $s^{(4)}$:

$$\Delta w_{.4}^{(1)} = \alpha(s^{(4)} - w_{.4}^{(1)})$$

Lampiran 3. Lanjutan

$$\Delta w_{.4}^{(1)} = \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix} + 0{,}5\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0{,}456 \\ 0{,}954 \end{bmatrix}\right) = \begin{bmatrix} -0{,}27 \\ 0{,}98 \end{bmatrix}$$

Iterasi pertama untuk $s^{(4)}$ telah dilakukan lalu dilanjutkan iterasi berikutnya. Dengan cara yang sama seperti di atas didapatkan jarak *Euclid*, $a^{(1)}, a^{(2)}$, dan bobot baru sebagai berikut:

Jarak *Euclid*

Iterasi	1	2	3	4	5	6	7	8	9
$\|s^{(4)} - W_1^{(1)}\|$	2	2	2	2	2	2	2	2	2
$\|s^{(4)} - W_2^{(1)}\|$	1,996	1,996	1,996	1,996	1,996	1,996	1,996	1,996	1,996
$\|s^{(4)} - W_3^{(1)}\|$	2,828	2,828	2,828	2,828	2,828	2,828	2,828	2,828	2,828
$\|s^{(4)} - W_4^{(1)}\|$	1,457	0,728	0,364	0,182	0,091	0,046	0,023	0,011	0,006

Di bawah ini merupakan tabel vektor pemenang setiap iterasi

	1	2	3	4	5	6	7	8	9
$a^{(1)}$	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1

Kemudian menghitung nilai vektor $a^{(2)}$ dengan rumus

$$a^{(2)} = \boldsymbol{W}^{(2)} a^{(1)}$$

Diperoleh hasil seperti pada tabel di bawah ini

	1	2	3	4	5	6	7	8	9
$a^{(2)}$	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1

Karena nilai vektor $a^{(2)}$ pada setiap iterasi sama seperti $t^{(4)}$, dapat disimpulkan bahwa $s^{(4)}$ diklasifikasikan tepat pada kelas kedua pada setiap iterasi dan vektor bobot *hidden* $w_{.4}^{(1)}$ dipindahkan mendekati $s^{(4)}$.

$$\Delta w_{.4}^{(1)} = \alpha(s^{(1)} - w_{.4}^{(1)})$$

Lampiran 3. Lanjutan

Diperoleh bobot baru untuk $w_{.2}^{(1)}$ seperti pada tabel di bawah ini

$w_{.4}^{(1)}$	1	2	3	4	5	6	7	8	9
	-0,27	-0,64	-0,82	-0,91	-0,95	-0,98	-0,99	-0,99	-1
	0,98	0,99	0,99	1	1	1	1	1	1

Dengan iterasi pada data *training* sebanyak 9 maka diperoleh kesimpulkan bahwa $w_{.4}^{(1)}$ dipindahkan mendekati $s^{(4)}$.

Lampiran 4. Statistika Deskiptif *Software* Minitab

Welcome to Minitab, press F1 for help.

Descriptive Statistics: Gizi_Kurang, Gemuk, IMD, Asi_Ekslusif, Konsumsi_Garam

Variable	N	N*	Mean	Variance	Median	Mode	N for Mode
Gizi_Kurang	514	0	17.709	59.407	17.5	14.2	7
Gemuk	514	0	4.793	10.191	4.1	3.2	19
IMD	514	0	42.669	245.343	43.9	41.8, 42 50	6
Asi_Ekslusif	514	0	33.301	435.927	29.1	6.2, 16.7 18.9, 25	5
Konsumsi _Garam	514	0	90.864	266.320	97.5	100	56

The data contain at least five mode values. Only the smallest four are shown.

Lampiran 5. 10-*Fold Cross Validation*

```
library(nnet)
library(e1071)
gizi<-read.csv(file.choose(),header=TRUE)
tmodel = tune.nnet(Kategori ~ ., data=gizi, size = 1:10)
summary(tmodel)
```

Parameter tuning of ‘nnet’:

- sampling method: 10-fold cross validation

- best parameters:
 size
 6

- best performance: 0.1702439

- Detailed performance results:

	size	error	dispersion
1	1	0.1785366	0.06439517
2	2	0.1780488	0.05809454
3	3	0.1824390	0.05904029
4	4	0.1741463	0.06225965
5	5	0.1868293	0.05813093
6	6	0.1702439	0.05174974
7	7	0.1829268	0.05586733
8	8	0.1848780	0.05424416
9	9	0.1853659	0.05547552
10	10	0.1829268	0.05467140

```
plot(tmodel)
tmodel$best.model
a 5-6-4 network with 64 weights
inputs: Gizi_Kurang Gemuk IMD Asi_Ekslusif Konsumsi_Garam
output(s): Kategori
options were - softmax modelling
```

Lampiran 6. *Backpropagation* Enam *Neuron Hidden Layer*

```
gizi_80<-read.csv(file.choose(),header=TRUE)
head(gizi_80)
  Gizi_Kurang   Gemuk   IMD   Asi_Ekslusif   Konsumsi_Garam
1       22.2       2.7      55.1          36.0           100.0
2        7.7       3.9      71.4          12.6            98.1
3       20.8       4.2      66.7          36.2            95.4
4        6.3       6.7      65.2          18.9            64.3
5       25.2       2.8      63.6          12.2            32.2
6       13.1       2.0      51.3          26.9            98.3
library(nnet)
clas.ind<-function(cl)
 {
    n<-length(cl)
    cl<-as.factor(cl)
    x<-matrix(0, n, length(levels(cl)))
    x[(1:n)+n*(unclass(cl)-1)]<-1
    dimnames(x)<-list(names(cl),levels(cl))
    x
 }
targets<-clas.ind(c(rep("a",422),rep("b",18),rep("c",59), rep("d",15))
)
head(targets)
      a b c d
[1,] 1 0 0 0
[2,] 1 0 0 0
[3,] 1 0 0 0
[4,] 1 0 0 0
[5,] 1 0 0 0
[6,] 1 0 0 0
tail(targets)
         a b c d
[509,] 0 0 0 1
[510,] 0 0 0 1
[511,] 0 0 0 1
[512,] 0 0 0 1
[513,] 0 0 0 1
[514,] 0 0 0 1
```

Lampiran 6. Lanjutan

```
set.seed(500)
samp<-c(sample(1:422,337),sample(423:440,14), sample(441:499,47
),sample(500:514,12))
training<-gizi_80[samp,]
head(training)
```

	Gizi_Kurang	Gemuk	IMD	Asi_Ekslusif	Konsumsi_Garam
352	13.3	3.2	47.5	31.0	98.0
306	26.4	5.8	46.0	29.0	99.7
410	21.2	2.8	31.7	9.7	97.8
196	14.2	4.6	42.4	33.1	95.1
340	10.0	4.8	48.7	28.2	95.2
86	15.4	3.0	50.0	31.9	99.4

```
testing<-gizi_80[-samp,]
head(testing)
```

	Gizi_Kurang	Gemuk	IMD	Asi_Ekslusif	Konsumsi_Garam
7	14.0	4.1	71.5	17.1	99.4
9	20.3	3.9	55.9	11.1	3.1
19	8.3	8.8	42.2	22.7	99.3
24	15.7	14.0	21.4	13.3	98.3
25	11.4	13.1	5.9	10.7	99.3
28	20.2	15.4	40.1	11.5	85.8

```
gizi6<-nnet(gizi_80[samp,],targets[samp,],size=6,rang=0.5,decay=0.
1,maxit=1000)
# weights:  64
initial  value 440.059205
iter  10 value 141.550681
iter  20 value 104.559735
iter  30 value 100.975868
iter  40 value 98.003064
iter  50 value 95.689612
iter  60 value 91.970926
iter  70 value 89.101359
iter  80 value 87.702358
iter  90 value 87.066221
iter 100 value 85.949119
iter 110 value 85.470046
iter 120 value 85.462353
```

Lampiran 6. Lanjutan

```
final  value 85.462347
converged
head(gizi6)
$n
[1] 5 6 4

$nunits
[1] 16

$nconn
 [1]  0  0  0  0  0  0  0  6 12 18 24 30 36 43 50 57 64

$conn
 [1]  0  1  2  3  4  5  0  1  2  3  4  5  0  1  2  3  4  5  0  1  2
[22]  3  4  5  0  1  2  3  4  5  0  1  2  3  4  5  0  6  7  8  9 10
[43] 11  0  6  7  8  9 10 11  0  6  7  8  9 10 11  0  6  7  8  9 10
[64] 11

$nsunits

[1] 16

$decay
[1] 0.1

str(gizi6)
List of 15
 $ n           : num [1:3] 5 6 4
 $ nunits      : int 16
 $ nconn       : num [1:17] 0 0 0 0 0 0 0 6 12 18 ...
 $ conn        : num [1:64] 0 1 2 3 4 5 0 1 2 3 ...
 $ nsunits     : int 16
 $ decay       : num 0.1
 $ entropy     : logi FALSE
 $ softmax     : logi FALSE
 $ censored    : logi FALSE
 $ value       : num 85.5
 $ wts         : num [1:64] 0.0601 1.3822 -0.6342 0.2859 -0.0589 ...
```

Lampiran 6. Lanjutan

```
 $ convergence  : int 0
 $ fitted.values: num [1:410, 1:4] 0.903 0.983 0.983 0.9 0.654 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:410] "352" "306" "410" "196" ...
  .. ..$ : chr [1:4] "a" "b" "c" "d"
 $ residuals    : num [1:410, 1:4] 0.0969 0.0165 0.0165 0.0998 0.345
9 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:410] "352" "306" "410" "196" ...
  .. ..$ : chr [1:4] "a" "b" "c" "d"
 $ call         : language nnet.default(x = gizi_80[samp, ], y = targets[s
amp, ], size = 6,      rang = 0.5, decay = 0.1, maxit = 1000)
 - attr(*, "class")= chr "nnet"
wts.in6<-gizi6$wts
wts.in6
 [1]  0.060136826  1.382172260 -0.634207320  0.285945285 -0.058
934325
 [6] -0.261715167  0.065472070  0.602802974 -1.235498087 -0.066
303681
[11]  0.283279701  0.088293115 -0.090713956 -1.235976028  0.694
636976
[16]  0.257797388  0.320245377 -0.016064337 -0.230076924  0.468
189073
[21] -0.762546583 -0.081598086 -0.160195537  0.238504886  0.063
470222
[26]  0.876194018  1.849763491  0.103953194  0.009915610 -0.173
790050
[31]  0.183626166 -1.486624849 -1.644324602  0.584097953  0.005
903385
[36] -0.078917737 -1.522878050  1.564274814  1.126744294 -1.717
106305
[41]  1.043039510  1.876536298 -2.072894920 -1.236185247 -0.494
462026
[46] -0.482691086  0.225267337 -0.776898363 -0.580950530  0.218
777424
[51]  0.922392838 -1.155411341 -1.678883055  1.488521746 -1.383
037869
```

Lampiran 6. Lanjutan

```
[56] -0.789287706  1.550441423 -0.685835782 -1.316733793 -0.175
221962
[61]  0.220814976 -0.558234574 -1.526045162  0.857628694
conn6<-gizi6$conn
conn6
 [1]  0  1  2  3  4  5  0  1  2  3  4  5  0  1  2  3  4  5  0  1  2
[22]  3  4  5  0  1  2  3  4  5  0  1  2  3  4  5  0  6  7  8  9 10
[43] 11  0  6  7  8  9 10 11  0  6  7  8  9 10 11  0  6  7  8  9 10
[64] 11
fitvalue<-gizi6$fitted.values
fitvalue
```

	a	b	c	d
352	0.90313015	0.03537212	0.07616503	0.01924182
306	0.98349638	0.02735384	0.01655997	0.01389755
410	0.98349750	0.02735414	0.01655839	0.01389850
196	0.90017747	0.03589771	0.07798812	0.01999102
340	0.65413313	0.05319139	0.21203434	0.06024080
86	0.91526584	0.03402043	0.06880101	0.01734338
214	0.91732719	0.03387316	0.06741563	0.01720170
385	0.98270503	0.02767223	0.01717728	0.01428534
344	0.98308488	0.02745154	0.01690650	0.01396185
294	0.70584546	0.05353262	0.18450454	0.05801999
117	0.91451573	0.03403195	0.06944171	0.01727677
367	0.97946478	0.02814212	0.02001539	0.01429604
314	0.98347615	0.02735832	0.01657736	0.01390000
68	0.98339290	0.02737604	0.01664993	0.01390875
299	0.59806880	0.05678105	0.24335954	0.07182195
104	0.69136728	0.05460347	0.19148115	0.06153200
207	0.84029101	0.03537002	0.09740087	0.03360143
251	0.91466571	0.03402341	0.06933898	0.01726893
246	0.91463548	0.03402663	0.06936293	0.01727030
103	0.98346592	0.02736045	0.01658648	0.01390091
32	0.97643035	0.03059400	0.02159643	0.01874659
325	0.98079058	0.02789858	0.01888965	0.01417263
15	0.98348307	0.02736571	0.01657614	0.01390190
216	0.98349189	0.02735481	0.01656387	0.01389804
107	0.84635606	0.04039326	0.10874394	0.02800479
324	0.98203507	0.02765678	0.01782386	0.01405051

Lampiran 6. Lanjutan

```
276         0.98160500 0.02774192 0.01819324 0.01409350
354         0.98349695 0.02735440 0.01655855 0.01389884
16          0.91434224 0.03404147 0.06953913 0.01730063
52          0.87176363 0.03907372 0.09433863 0.02510517
27          0.69498393 0.05470263 0.19070895 0.06134407
394         0.94921864 0.03898613 0.03811388 0.03603621
241         0.91457843 0.03402848 0.06940047 0.01727229
39          0.78352409 0.05149808 0.13539421 0.05807582
147         0.86226381 0.04089025 0.09847731 0.02909194
327         0.98349934 0.02735321 0.01655738 0.01389723
319         0.98149807 0.02776282 0.01828490 0.01410406
391         0.98041264 0.02885996 0.01876925 0.01606055
357         0.86166499 0.03551373 0.09530195 0.02475051
18          0.98349887 0.02735343 0.01655774 0.01389753
75          0.91437923 0.03405212 0.06952362 0.01730476
161         0.97476518 0.02890852 0.02393416 0.01468271
317         0.98349934 0.02735321 0.01655738 0.01389723
184         0.85342815 0.03611900 0.10414809 0.02272239
323         0.98349876 0.02735334 0.01655788 0.01389731
223         0.98349330 0.02735451 0.01656264 0.01389788
217         0.98347087 0.02735933 0.01658215 0.01390032
176         0.97841342 0.02832549 0.02090112 0.01438827
90          0.59024162 0.06212683 0.22184960 0.08590544
89          0.91681741 0.04559505 0.05366754 0.05438906
 [ reached getOption("max.print") -- omitted 160 rows ]
test.cl6<-function(true,pred)
 {
    true<-max.col(true)
    cres<-max.col(pred)
    table(true,cres)
 }
 test.cl6(targets[-samp,], predict(gizi6, gizi_80[-samp,]))
      cres
true  1  3
   1 81  4
   2  4  0
   3 12  0
   4  0  3
```

Lampiran 6. Lanjutan

```
train.cl6<-function(true,pred)
 {
    true<-max.col(true)
    cres<-max.col(pred)
    table(true,cres)
 }
train.cl6(targets[samp,], predict(gizi6, gizi_80[samp,]))
       cres
true   1   3
   1 335  2
   2  11   3
   3  22  25
   4   3   9
```

Lampiran 7. *Function* LVQ1

```
library(class)
trace(lvq1)
lvq1
function (x, cl, codebk, niter = 100 * nrow(codebk$x), alpha = 0.03)
{
    x <- as.matrix(x)
    if (any(is.na(x)) || any(is.na(cl)))
        stop("no missing values are allowed")
    n <- nrow(x)
    p <- ncol(x)
    nc <- dim(codebk$x)[1L]
    if (length(cl) != n)
        stop("'x' and 'cl' have different lengths")
    iters <- sample(n, niter, TRUE)
    z <- .C(VR_lvq1, as.double(alpha), as.integer(n), as.integer(p),
        as.double(x), as.integer(unclass(cl)), as.integer(nc),
        xc = as.double(codebk$x), as.integer(codebk$cl), as.integer(nite
r),
        as.integer(iters - 1L))
    xc <- matrix(z$xc, nc, p)
    dimnames(xc) <- dimnames(codebk$x)
    list(x = xc, cl = codebk$cl)
}
<bytecode: 0x0000000007147950>
<environment: namespace:class>
```

Lampiran 8. *Function* OLVQ1

```
library(class)
trace(olvq1)
olvq1
function (x, cl, codebk, niter = 40 * nrow(codebk$x), alpha = 0.3)
{
    x <- as.matrix(x)
    if (any(is.na(x)) || any(is.na(cl)))
        stop("no missing values are allowed")
    n <- nrow(x)
    p <- ncol(x)
    nc <- dim(codebk$x)[1L]
    if (length(cl) != n)
        stop("'x' and 'cl' have different lengths")
    iters <- sample(n, niter, TRUE)
    z <- .C(VR_olvq, as.double(alpha), as.integer(n), as.integer(p),
        as.double(x), as.integer(unclass(cl)), as.integer(nc),
        xc = as.double(codebk$x), as.integer(codebk$cl), as.integer(nite
r),
        as.integer(iters - 1L))
    xc <- matrix(z$xc, nc, p)
    dimnames(xc) <- dimnames(codebk$x)
    list(x = xc, cl = codebk$cl)
}
<bytecode: 0x00000000071ad968>
<environment: namespace:class>
```

Lampiran 9. *Function* LVQINIT

```
library(class)
trace(lvqinit)
lvqinit
function (x, cl, size, prior, k = 5)
{
    x <- as.matrix(x)
    n <- nrow(x)
    p <- ncol(x)
    if (length(cl) != n)
        stop("'x' and 'cl' have different lengths")
    g <- as.factor(cl)
    if (any(is.na(x)) || any(is.na(g)))
        stop("no missing values are allowed")
    counts <- tapply(rep(1, length(g)), g, sum)
    prop <- counts/n
    np <- length(prop)
    if (missing(prior))
        prior <- prop
    else if (any(prior < 0) || round(sum(prior), 5) != 1)
        stop("invalid 'prior'")
    if (length(prior) != np)
        stop("'prior' is of incorrect length")
    if (missing(size))
        size <- min(round(0.4 * np * (np - 1 + p/2), 0), n)
    inside <- knn.cv(x, cl, k) == cl
    selected <- numeric(0)
    for (i in 1L:np) {
        set <- seq_along(g)[unclass(g) == i & inside]
        if (length(set) > 1L)
            set <- sample(set, min(length(set), round(size *
                prior[i])))
        selected <- c(selected, set)
    }
    list(x = x[selected, , drop = FALSE], cl = cl[selected])
}
<bytecode: 0x0000000007ff4b88>
<environment: namespace:class>
```

Lampiran 10. LVQ *Size Codebook* 9

```
library(class)
trace(lvq1)
lvq1
trace(olvq1)
olvq1
trace(lvqinit)
lvqinit
train_80=read.csv(file.choose(),header=FALSE)
train_80=as.matrix(train_80)
cl<- factor(c(rep("1",337), rep("2",14), rep("3",47), rep("4",12)))
cd<-lvqinit(train_80, cl, 9)
trace: lvqinit(train_80, cl, 9)
cd
$x
      V1  V2   V3   V4    V5
[1,] 20.5 4.3 33.1 31.3  99.3
[2,] 16.3 9.9 42.0 42.7  91.9
[3,] 19.6 7.4 38.9 27.6  98.8
[4,] 24.0 3.9 11.5 15.3  92.6
[5,] 25.6 0.6 14.8 17.8  93.0
[6,] 19.9 2.6 32.7 15.4  97.0
[7,] 19.2 5.3 44.4 10.1  98.7
[8,]  2.0 1.7 48.9 62.9 100.0

$cl
[1] 1 1 1 1 1 1 1 3
Levels: 1 2 3 4

lvqtest(cd, train_80)
[1]   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3
3
[36] 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1
1
[71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1
1
[106] 1 1 1 1 3 1 1 3 1 3 1 1 3 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1
 1
```

Lampiran 10. Lanjutan

```
[141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[176] 1 3 3 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 3 1 1
 1
[211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[246] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[316] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 3 1 1 1 1 1 1 1 1 1
 1
[351] 1 3 1 3 3 1 1 1 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1
 1
[386] 3 1 1 1 1 1 3 3 3 1 3 1 1 1 1 1 1 3 1 3 1 1 1 1 1
Levels: 1 2 3 4
cd0<-olvq1(train_80, cl, cd)
trace: olvq1(train_80, cl, cd)
cd0
$x
         V1          V2          V3          V4          V5
[1,] 21.112545  4.0407126 41.21405 36.685935  97.14413
[2,] 22.842627  4.1045799 49.21839 67.305530  93.84360
[3,] 22.839540  3.4839509 48.63124 34.876231  72.28047
[4,] 16.519629  8.2173224 10.86371 16.894519  97.74594
[5,] 31.879881  0.1185915 31.54453  5.498723  53.27998
[6,] 23.279841  4.9376361 35.07997 13.167801  96.65919
[7,] 17.922673  5.2642688 52.69584 23.141052  96.45805
[8,] -4.679696 -1.4489500 44.85973 59.508274 102.34942
$cl
[1] 1 1 1 1 1 1 1 3
Levels: 1 2 3 4
lvqtest(cd0, train_80)
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1
1
[71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1
1
```

Lampiran 10. Lanjutan

```
[106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[246] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[316] 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[351] 1 3 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1
[386] 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 3 1 3 1 1 1 1 1
Levels: 1 2 3 4
cd1<-lvq1(train_80, cl, cd0)
trace: lvq1(train_80, cl, cd0)
cd1
$x
         V1         V2       V3        V4        V5
[1,] 20.723364  4.0575934 39.03390 38.029122  96.69890
[2,] 25.222821  4.2965367 44.98342 73.788548  92.66873
[3,] 22.693427  3.4451732 47.32035 37.701128  66.38385
[4,] 18.635559  7.4194770 10.78066 18.161797  97.48227
[5,] 30.110534  0.9158375 32.22074  7.224778  49.41232
[6,] 22.008040  4.6061178 34.86411 13.864257  93.41314
[7,] 19.963320  4.7264102 54.63300 26.267730  95.63446
[8,] -5.428779 -1.7634182 44.35844 59.838353 102.53254
$cl
[1] 1 1 1 1 1 1 1 3
Levels: 1 2 3 4
lvqtest(cd1, train_80)
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
```

Lampiran 10. Lanjutan

[71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1
1
[106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
[141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
[176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
[211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
[246] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
[281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
[316] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1
1
[351] 1 3 1 1 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1
1
[386] 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 3 1 3 1 1 1 1 1
Levels: 1 2 3 4
test_20=read.csv(file.choose(),header=FALSE)
test_20=as.matrix(test_20)
lvqtest(cd1,test_20)
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 3 1 1 1 1 1
[36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
[71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1
Levels: 1 2 3 4