Dr. Hendra Jaya, M.T
Drs. Sabran, M.Pd.
Dr. Muh. Ma'ruf Idris, S.T, M.T
Dr. Yasser A. Djawad, ST., M.Sc
A. Ilham, A.Md.
Ansari Saleh Ahmar, S.Si., M.Sc

KECERDASAN BUATAN

Penulis:
Dr. Hendra Jaya, S.Pd., M.T., dkk.

Diterbitkan Oleh : Fakultas MIPA Universitas Negeri Makassar
Cetakan : Pertama, Januari 2018
Lay Out : M. Ilham
Desain Sampul : M. Ilham

Hak Cipta dilindungi Undang-undang,
Dilarang memperbanyak isi buku ini sebagian atau seluruhnya dalam bentuk dan cara apapun juga, baik secara Mekanis maupun Elektronis, termasuk fotocopy, rekaman, dan lain-lain tanpa izin tertulis dari penerbit

KECERDASAN BUATAN

Penulis : Dr. Hendra Jaya, S.Pd., M.T.,
Drs. Sabran, M.Pd.
Dr. Muh. Ma'ruf Idris, S.T., M.T.
Yasser A. Djawad, S.T., M.Sc., Ph.D
A. Ilham, S.Pd.
Ansari Saleh Ahmar, S.Si., M.Sc.

- Makassar

Fakultas MIPA Universitas Negeri Makassar
V- 315;23-15,5 cm

ISBN 978-602-99837-9-1

KATA PENGANTAR

Puji syukur kita panjatkan kehadirat Allah SWT, semoga rahmat dan keselamatan dilimpahkan kepada nabi Muhammad SAW, para sahabat dan seluruh umatnya. dengan limpahan kemudahan dan keluasan pikiran yang diberikan sehingga penulis dapat menyelesaikan buku pegangan mata kuliah Kecerdasan Buatan (AI).

Melalui buku ini, kami berharap akan menambah referensi yang berarti bagi Dosen pengampuh mata kuliah ini, khususnya di lingkungan Jurusan Pendidikan Teknik Elektronika Fakultas Teknik Universitas Negeri Makassar. Bagi mahasiswa, buku ini diharapkan dapat menjadi landasan pemikiran dan pengetahuan dalam bidang kecerdasan buatan (AI). Harapannya, setelah memahami buku pegangan ini dengan baik, mahasiswa dapat memahami pentingnya mempelajari kecerdasan buatan (AI) seiring berkembangnya waktu dan semakin pesatnya kemajuan teknologi.

Segala usaha telah di lakukan atas selesainya buku ini. Namun dalam usaha yang maksimal tentu masih terdapat beberapa kekurangan, isi pada tujuan pembelajaran belum sinkron dengan Tagihan mata kuliah (Tugas) maka dari itu akhir kata kami berharap akan saran dan pendapat lainnya dari para pembaca terhadap buku ini agar menjadi lebih baik lagi. Semoga bermanfaat, Amin.

Makassar, Januari 2018

Penulis

DAFTAR ISI

BAB 1

ARTIFICIAL INTELLIGENCE (AI)

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Mengetahui definisi AI.
2. Menjelaskan perbedaan kecerdasan buatan dan kecerdasan alami.
3. Menguraikan sejarah perkembangan AI.
4. Mengetahui penggunaan AI.
5. Mengetahui kelebihan AI.
6. Mengetahui jenis-jenis aplikasi dari AI.

B. Uraian Materi

1.1 Mengapa Mempelajari AI?

Manusia diciptakan dengan kecerdasan yang sangat luar biasa. Bayi yang baru lahir hanya bisa menangis saat lapar dan segera berhenti begitu sang ibu memberinya ASI (air susu ibu). Hal ini merupakan konsep belajar paling sederhana melalui sebuah pemetaan dijaringan syaraf otaknya: jika dia lapar dan menangis, maka ibunya pasti segera datang untuk memeberinya ASI. Pembelajaran yang diperoleh melalui stimulasi dari lingkungannya terjadi sangat cepat, eksponensial. Di usia sekitar dua tahun, seorang bayi umumnya sudah mulai mengucapkan beberapa kata dan mampu mengenali berbagai benda meskipun yang terlihat hanya bagian tertentu dari benda tersebut. Ketika melihat bagian kecil dari ekor cicak, dia akan mampu mengidentifikasi bahwa ada seekor cicak sedang bersembunyi dibalik bingkai foto yang tergantung di dinding. Satu atau dua tahun kemudian, dia sudah lihai berkomunikasi menggunakan satu kalimat lengkap dengan subyek-predikat-obyek-keterangan, padahal dia sama sekali tidak pernah diajari tata bahasa. Di usia lima tahun, dia sudah mahir berargumen menggunakan kalimat-kalimat majemuk yang kompleks. Pada usia selanjutnya, kecerdasannya akan berkembang dengan sangat pesat, membentuk kecerdasan majemuk (multiple intelligence). Sampai saat ini, belum ada satu mesinpun yang bisa menyamai kecerdasan majemuk manusia secara keseluruhan.

Selama bertahun-tahun para filsuf berusaha mempelajari kecerdasan manusia. Dari pemikiran para filsuff tersebut, lahirlah AI sebagai cabang ilmu yang juga berusaha memahami kecerdasan manusia. AI berusaha membangun entitas-entitas cerdas yang sesuai dengan pemahaman manusia. Entitas-entitas cerdas yang dibangun AI ini ternyata sangat menarik dan mempercepat proses pemahaman terhadap kecerdasan manusia. Oleh karena iitu, AI menjadi bidang yang sangat oenting dalam memahami kecerdasan manusia. Dengan didukung perkembangan *hardware* dan *software* yang sangat

beragam, AI telah menghasilkan banyak produk yang sangat penting dan berguna bagi kehidupan manusia. Hingga saat ini, AI terus dipelajari dan dikembangkan secara meluas dan mendalam. Saat ini, kita mengenal banyak bidang studi yang berawal dari AI, seperti *fuzzy systems*, *soft computing*, dan banyak lagi lainnya yang semakin fokus pada bidang kajian dan permasalahan tertentu. Pada buku ini, pembahasan AI difokuskan pada jaringan syaraf tiruan.

1.2 Definisi AI

Apakah Kecerdasan Buatan Itu? Kecerdasan buatan atau *artificial intelligence* (AI) merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiiring dengan perkebangan jaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

Manusia bisa menjadi pandai dalam menyelesaikan segala permasalahan didunia ini karena menusia mempunyai pengetahuan dan pengalaman pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki oleh seseorang tentu saja diharapkan akanl lebih mampu dalam menyelesaikan peermasalahan. Namun bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang mereka miliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian pula, dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik.

Lebih detilnya, kecerdasan buatan dapat dipandang dari berbagai sudut pandang, antara lain:

1. Sudut pandang kecerdasan
   Kecerdasan buatan akan membuat mesin menjadi 'cerdas' (mampu berbuat seperti apa yang dilakukan oleh manusia).
2. Sudut pandang penelitian
   Kecerdasan buatan adalah suatu studi bagaimana membuat agar komputer dapat melakukan sesuatu sebaik yang dikerjakan oleh manusia.

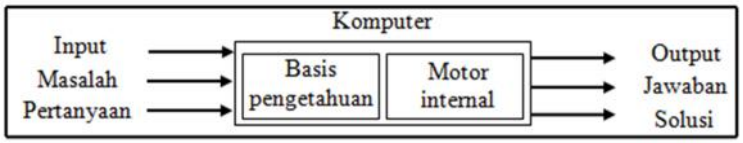
   Domain yang sering dibahas oleh para peneliti meliputi:

   a. *Mundane task*
      - Persepsi (*vision* & *speech*).
      - Bahasa alami (understanding, generation & translation).
      - Pemikiran yang bersifat commonsense.
      - Robot control.

   b. Formal *task*
      - Permainan / games.
      - Matematika (geometri, logika, kalkulus, integral, pembuktian)

   c. *Epert task*
      - Analisis finansial.
      - Analisis medikal.
      - Analisis ilmu pengetahuan.
      - Rekayasa (desain, pencarian kegagalan, perencanaan manufactur)
3. Sudut pandang bisnis
   Kecerdasan buatan adalah kumpulan peralatan yang sangat powerful dan metodologis dalam menyelesaikan masalah-masalah bisnis.
4. Sudut pandang pemrograman
   Kecerdasan buatan meliputi studi tentang pemrograman simbolik, penyelesaian masalah (*problem solving*) dan pencarian (*searching*).
   Untuk melakukan aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan, yaitu (Gambar 1.1):
   - Basis pengetahuan (*knowledge base*), berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.

- Motor interferensi (*inference engine*), yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.



Gambar 1.1 Penerapan Konsep Kecerdasan Buatan

Kecerdasan buatan dan kecerdasan alami

Jika dibandingkan dengan kecerdasan alami (kecerdasan yang dimiliki oleh manusia), kecerdasan buatan memiliki beberapa keuntungan secara komersial antara lain:

a. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer & komputer tidak mengubahnya.
b. Kecerdasan buatan lebih mudah diduplikasi & disebarkan. Mentransfer pengetahuan manusia dari orang ke orang lain membutuhkan proses yang sangat lama; dan juga suatu keahlian itu tidak akan pernah dapat diduplikasi dengan lengkap. Oleh karena itu, jika pengetahuan terletak pada pada suatu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer yang lain.
c. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebi murah dibandingkan harus mendatangkan seseeorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
d. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan kaarena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami akan senantiasa berubah-ubah.

e. Kecerdasan buatan dapat didokumentasi. Keputusan yang dibuat oleh komputer dapat didokumentasi dengan mudah dengan cara melacak setiap aktifitas dari sistem tersebut. Kecerdasan alami sangat sulit untk direproduksi.
f. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami.
g. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

Sedangkan keuntungan dari kecerdasan alami adalah:

a. Kreatif. Kemampuan untuk menambah ataupun memenuhi pengetahuan itu sangat melekat pada jiwa manusia. Pada kecerdasan buatan, untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun.
b. Kecerdasan alami memungkinkan orang untuk menggunakan pengalaman secara langsung. Sedangkan pada kecerdasan buatan harus bekerja dengan input-input simbolik.
c. Pemikiran manusia dapat digunakan secara luas, sedangkan kecerdasan buatan terbatas.

1.3 Sejarah AI

Istilah AI pertama kali dikemukakan pada tahun 1956 dikonferensi Darthmouth. Sejak saat itu AI terus dikembangkan sebab berbagai penelitian mengenai teori-teori dan prinsip-prinsipnya juga terus berkembang. Meskipun istilah AI baru muncul tahun 1956, tetapi teori-teori mengaraah ke AI sudah muncul sejak tahun 1941. Secara lengkap, berikut ini tahapan-tahapan sejarah perkembangan AI:

Era komputer elektronik (1941)

Pada tahun 1941 telah ditemukan alat penyimpanan dan pemrosesan informasi. Penemuan tersebut dinamakan komputer elektronik yang dikembangkan di USA dan jerman. Komputer pertama ini memerlukan ruangan yang luas dan ruang AC yang terpisah. Saat itu komputer me;ibatkan konfigurasi ribuan kabel untuk menjalankan suatu program. Hal ini sangat merepotkan bagi

para *programmer*. Pada tahun 1949, berhasil dibuat komputer yang mampu menyimpan program sehingga membuat pekerjaan untuk memasukkan program menjadi lebih mudah. Penemuan ini menjadi dasar pengembangan program yang mengarah ke AI.

Masa persiapan AI (1943 – 1956)

Pada tahun 1943, Warren McCulloch dan Walter Pitts mengemukakan tiga hal:pengetahuan fisiologi dasar dan fungsi sel syaraf dalam otak, analisis formal tentang logika proporsi (*propositional logic*), dan teori komputasi turing. Mereka berhasil membuat suatu model syaraf tiruan (*artificial neuron*) di mana setiap *neuron* digambarkan sebagai *on* dan *off*. Mereka menunjukkan bahwa setiap fungsi dapat dihitung dengan suatu jaringan sel syaraf dan bahwa semua hubungan logis dapat diimplementasikan dengan struktur jaringan yang sederhana.

Pada tahun 1950, Norbert Wiener mebuat penelitian mengenai prinsip-prinsip teori *feedback*. Contoh yang terkenal adalah *thermostat*. Penemuan ini juga merupakan awal perkembangan AI. Pada tahun 1956, John McCarthy (yang setelah lulus dari Princeton kemudian melanjutkan ke Dartmouth College) meyakinkan Minsky, Claude Shannon dan Nathaniel Rochester untuk membantunya melakukan penelitian dalam bidang *Automata*, jaringan sel syaraf dan pembelajaran intelejensia. Mereka mengerjakan proyek ini selama dua bulan di Dartmouth. Hasilnya adalah program yang mampu ber-pikir non-numerik dan menyelesaikan masalah pemikiran, yang dinamakan *Principia Mathematica*. Hal ini menjadikan McCarthy disebut sebagai *Father of AI* (Bapak AI).

Awal perkembangan AI (1952 – 1969)

Pada tahun-tahunn pertama pengembangannya, AI mengalami banyak kesuksesan. Diawali dengan kesuksesan Newell dan Simon dengan sebuah program yang disebut *General Problem Solver*. Program ini dirancang untuk memulai penyelesaian masalah secara manusiawi. Pada tahun 1958, McCarthy di MTT Lab Memo No. 1 mendefinisikan bahasa pemrograman tingkat tinggi yaitu

LISP, yang sekarang mendominasi pembuatan program-program AI. Kemudian, McCarthy membuat program yang dinamakan *Programs With Common Sense*. Di dalam program tersebut, dibuat rancangan untuk menggunakan pengetahuan dalam mencari solusi. Pada tahun 1959, Nathaniel Rochester dari IBM dan mahasiswa-mahasiswanya mengeluarkan program AI *Geometry Theorm Prover*. Program ini dapat membuktikan suatu teorema menggunakan axioma-axioma yang ada. Pada tahun 1963, program yang dibuat James Slagle mampu menyelesaikan masalah integral tertutup untuk mata kuliah kalkulus. Pada tahun 1968, program analogi buatan Tom Evan menyelesaikan masalah analogi geometris yang ada pada tes IQ.

Perkembangan AI melambat (1966 – 1974)

Prediksi Herbert Simon pada tahun 1957 yang menyatakan bahwa AI akan menjadi ilmu pengetahuan yang akan berkembang dengan pesat ternyata meleset. Pada 10 tahun kemudian, perkembangan AI melambat. Hal ini disebabkan adanya 3 kesulitan utama yang dihadapi AI, yaitu:

1. Masalah pertama: program-program AI yang bermunculan hanya mangandung sedikit atau bahkan tidak mengandung sama sekali pengetahuan (*knowledge*) pada subjeknya. Program-program AI berhasil hanya karena manipulasi sintetis yang sederhana. Sebagai contoh adalah Weizenbaum's ELIZA program (1965) yang dapat melakukan percakapan serius pada berbagai topik, sebenarnya hanyalah peminjaman dan manipulasi kalimat-kalimat yang diketikkan oleh manusia.
2. Masalah kedua: banyak masalah yang harus diselesaikan oleh AI. Karena terlalu banyaknya masalah yang berkaitan, maka tidak jarang banyak terjadi kegagalan pada pembuatan program AI.
3. Masalah ketiga: ada beberapa batasan pada struktur dasar yang digunakan untuk menghasilkan perilaku intelejensia. Sebagai contoh adalah pada tahun 1969 buku Minsky dan Papert Perceptrons membuktikan bahwa program-program *perceptrons* dapat mempelajari segala sesuatu, tetapi program-

program tersebut hanya mempresentasikan sejumlah kecil saja. Sebagai contoh dua masukan *perceptrons* yang bebeda tidak dapat dilatihkan untuk mengenali kedua masukan yang berbeda tersebut.

Sistem berbasis pengetahuan (1969 – 1979)

Pengetahuan adalah kekuatan pendukung AI. Hal ini dibuktiikan dengan program yang dibuat oleh Ed Feigenbaum, Bruce Buchanan dan Joshua Lederberg yang membuat program untuk memecahkan masalah struktur molekul dari informasi yang didapatkan dari *spectometer* massa. Program ini dinamakan *Dendral programs* yang berfokus pada segi pengetahuan kimia. Dari segi diagnosis media juga sudah ada yang menemukannya, yaitu Saul Amarel dalam proyek *computer in biomedicine*. Proyek ini diawali dari keinginan untuk mendapatkan diagnosa penyakit berdasarkan pengetahuan yang ada pada mekanisme penyebab proses penyakit.

AI menjadi sebuah industri (1980 – 1988)

Industrialisasi AI diawali dengan ditemukannya *expert system* (sistem pakar) yang dinamakan R1 yang mampu mengkonfigurasi sistem-sistem komputer baru. Program tersebuut mulai dioperasikan di *Digital Equipment Corporation* (DEC), McDermott, pada tahun 1982. Pada tahun 1986, program ini telah berhasil menghemat US$40 juta per tahun. Pada tahun 1988, kelompo AI di DEC menjalankan 40 sistem pakar. Hampir semua perusahaan besar di USA mempunyai divisi AI sendiri yang menggunakan ataupun mempelajari sistem pakar. *Booming* industri AI ini juga melibatkan perusahaan-perusahaan besar seperti Carnegie Group, Inference, Intellicorp, dan Technoledge yang menawarkan *software tools* untuk membangun sistem pakar. Perusahaan *hardware* seperti LISP dan Machines Inc., Texas Instruments, Symbolics, dan Xerox juga turut berperan dalam membangun *workstation* yang dioptimasi untuk pembangunan program LISP. Sehingga, perusahaan yang sejak tahun 1982 hanya menghasilan beberapa juta US dolar pper tahun meningkat menjadi 2 milyar US dolar per tahun pada tahun 1988.

Kembalinya jaringan syaraf tiruan (1986 – sekarang)

Meskipun bidang ilmu komputer menolak jaringan syaraf tiruan setelah diterbitkannya buku "*perceptrons*" karangan Minsky dan Papert, tetapi para ilmuan masih mempelajari bidang ilmu tersebut dari sudut pandang yang lain yaitu fisika. Para ahli seperti Hopfield (1982) menggunakan teknik-teknik mekanika statistika untuk menganalisa sifat-sifat penyimpanan dan optimasi pada jaringan syaraf. Para ahli psikollogi, David Rumelhart dan Geoff Hinton, melanjutkan penelitian mengenai model syaraf pada memori. Pada tahun 1985-an sedikitnya empat kelompok riset menemukan kembali algoritma belajar propagasi balik (*Back-Propagation Learning*). Algoritma ini berhasil diimplementasikan ke dalam bidang ilmu komputer dan psikologi.

1.4 AI Masa Depan

Ray Kurzweil dalam buku Suyanto (2014:9) menyampaikan tiga buah pertanyaan: mampukah suatu kecerdasan membuat kecerdasan lain yang lebih cerdas daripada dirinya sendiri? Apakah kita lebih cerdas daripada proses evolusi yang menciptakan kita? Selanjutnya, akankah kecerdasan yang kita buat melebihi kecerdasan kita sendiri? Jawaban untuk ketiga pertanyaan tersebut adalah 'Ya, mungkin saja'. Dalam buku tersebut, Ray Kurzweil mebuat prediksi-prediksi kejadian 100 tahun yang akan datang, mulai dari 2009 hingga 2099, ke dalam beberapa tahapan berikut ini:

Tahun 2009

Sebuah PC seharga US$ 1000 akan dapat melakukan sekitar satu trilliun kalkulasi per detik. Komputer akan manjadi sangat kecil, menempel pada pakaian dan perhiasan. Sebagian besar transaksi bisnis rutin berada di antara manusia dan personalitas *virtual*. Telepon dengan terjemmahannya (*translating telephone*), pemanggil dan yang dipanggil bisa menggunakan dua bahasa berbeda, akan digunakan luas di masyarakat.

Tahun 2019

Sebuah PC seharga US$ 1000 akan setara dengan kemamppuan komputasional otak manusia. Komputer semakin mudah dioperasikan, tidak terlihat dan menempel di mana saja. *Virtual reality* sudah dalam tiga dimensi. Sebagian besar interaksi dengan komputer sudah melalui isyarat tubuh (*gesture*) dan komunikasi uucapan bahasa dua arah. Lingkungan realistis yang mancakup segala hal (audio, visual, dan fisik) membuat manusia mampu melakukan sesuatu, secara *virtual*, dengan manusia lain, meskipun ada batasan secar fisik. Manusia mulai memiliki hubungan dengan personalitas otomatis seperti teman dan guru.

Tahun 2029

Sebuah PC seharga US$ 1000 akan setara dengan kemampuan komputasional seribu otak manusia. Komputer telah terhubung langsung ke otak manusia dengan koneksi *high-bandwidth*. Pencakokan otak berhasil dilakukan untuk meninngkatkan persepsi dan interpretasi secara audio dan visual, memori dan penalaran. Komputer mampu membaca semua literatur dan material multimedia yang dibangkitkan oleh mesin maupun manusia. Muncul diskusi tentang legalitas komputer dan konstitusi manusia.

Tahun 2049

Makanan yang diproduksi menggunakan *nano technology* mulai digunakan secara umum. Makanan tersebut mempunyai komposisi gizi yang baik, mempunyai rasa dan tekstur yang sama dengan makanan organik.

Tahun 2072

Picoengineering atau teknologi pada skala picometer atau 10^{-12} meter berhasil diapikasikan didunia nyata.

Tahun 2099

Ada kecenderungan kuat untuk membuat gabungan atara pemikiran manusia dengan kecerdasan mesin. Tidak ada lagi perbedaan yang jelas antara manusia dan mesin. Sebagian besar entitas tidak mempunyai kehadiran fisik yang permanen. Kecerdasan berbasis

mesin yang dituurunkan dari model-model kecerdasan manusia yang diperluas mengklaim dirinya sebagai manusia. Sebagian besar kecerdasan ini tidak terikat terhadap suatu unit pemrosesan komputasi yang khusus. Julah manusia-manusia berbasis *software* jauh melebihi manusia-manusia berbasis sel-sel syaraf alami (karbon). Bahkan diantara kecerdasan manusia yang masih menggunakan sel-sel sysaraf berbasis karbon tersebut, terdapat penggunnaan teknologi pencangkokan sel syaraf yang memberikan peningkatan besar-besaran terhadap kemampuan perseptual dan kognitif manusia. Manusia-manusia yang tidak menggunakan pencangkokan tersebut tidak dapat berpartisipasi dalam dialog dengan manusia-manusia lain yang mennggunakan pencangkokan sel syaraf.

Mungkinkah prediksi-prediksi pada tahapan-tahapan berikutnya akan berhasiil diwujudkan? Jika kita amati pada saat ini, komputer sekecil ponsel pintar dalam genggaman tangan memang sudah banyak beredar, namun komputer belum menempel pada pakaian dan perhiasan. Sebagian besar transaksi bisnis rutin berada diantara manusia dan personalitas virtual? Sepertinya sudah banyak terlihat dengan banyaknya transaksi *online*. Telepon dengan terjemahannya? Hingga saat ini, telepon dengan penerjemahan antarbahasa belum banyak digunakan secara luas di masyarakat. Kemungkinan masih digunakan secara terbatas dalam kelompok tertentu, seperti agen rahasia dan militer.

1.5 Aplikasi-Aplikasi Dari AI

Kecerdasan buatan (AI) telah dipelajari selama kira-kira 45 tahun. Hingga saat ini telah dihasilkan beberapa produk aplikasi AI secara komersial. Produk-produk tersebut umumnya dapat dijalankan pada perangkat keras komputer mulai dari komputer pribadi (PC) seharga USA$50,000. Secara khas masukan untuk produk-produk tersebut terbentuk data simbolis. Aplikasi-aplikasi AI antara lain:

a. *Game playing*
b. Sistem bahasa alami
c. Sistem perancangan dan pembuatan CAD/CAM

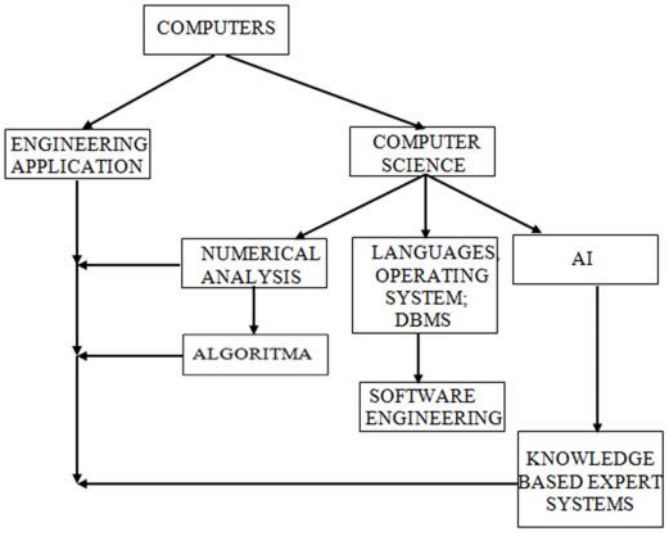
d. Sistem pakar reparasi perangkat keras
e. Manajemen data cerdas
f. Sistem otomisasi kantor
g. Analisa kecerdasan militer
h. Kendali dan pemanggilan informasi disk video
i. Kendali robot
j. Analisa program komputer
k. Diagnosis penyakit
l. Konfigurasi komputer
m. Ramalan senyawa kimia
n. Sistem ucapan
o. Sistem pakar operator komputer
p. Manajemen kendali senjata.

Apa AI hari ini adalah sesuatu yang baru sekarang? Bagian-bagian AI antara lain:

a. *Parallel Distributed Processing (Neural Network)*
b. *Machine Vision*
c. *Automaaatic Programming Tools*
d. *Memory Management*
e. *Pattern Recognition*
f. *Natural Language Processing*
g. *Development Of Knowledge Base*

Kecerdasan buatan (*Artificial intelligence*) adalah sub bagian dari ilmu komputer yang merupakan suatu teknik perangkat lunak yang pemrogramannya dengan cara menyatakana data, pemprosesan data dan penyelesaian masalah secara simbolik, dari pada secara numerik. Masalah-masalah dalam bentuk simbolik ini adalah masalah yang sering kita jumpai dalam kehidupan sehari-hari. Masalah-masalah ini lebih berhubungan dengan simbol dan konsep dari pada dengan angka-angka. Di sini dengan kecerdasan buatan diusahakan untuk membuat komputer seakan dapat berpikir secara cerdas. Apa yang dimaksud dengan kecerdasan buatan dan proses simbolik untuk mudah dapat dimengerti adalah dengan cara membandingkannya dengan program konvensional yang melakukan operasi numerik.

Program konvensional dapat menjawaab “X + Y” jika harga X dan Y diberikan, tetapii program ini tidak dapat menjawab bahwa “X + Y = 2X”, atau tidak dapat menjawab mengapa mobiil tidak dapat disaart.



Gambar 1.2 Hubungan antara pengembangan di dalam computer science dan engineering applications

1.6 Penggunaan AI

Artificial Intellegence dapat diaplikasikan untuk berbagai jenis:

1. Pengontrol
   a. Kontrol Process
      1) Pengendai tekanan
      2) Pengendali level
      3) Diagnosis kegagalan
   b. Kontrol Sumber Penggerak
      (a) Pengendalian motor listrik
      (b) Pengatur sumber tenaga pada Sentral/distribusi
   c. Kontrol Manifacturing
      (a) Penjadwalan proses kontrol
      (b) Dekomposition proses kontrol

   d. Kontrol Robot
      (a) Kontrol posisi
      (b) Kontrol jalur laluan
2. Peramalan
   a. Peramalan Pembebanan (*Load Forecasting*)
   b. Peramalan Pemadaman (*Blackout Forecasting*)
   c. Peramalan Saham (*Forex Forecasting*)
3. Perencanaan (*Planning*)
   a. Perencanaan proyek (*Project Planning*)
   b. Penjadwalan Kerja (*Job Shop Scheduling*)
   c. Perencanaan Aggaran (*Budget Planning*)
4. Proses Image (*Image Processing*)
   a. *Computer Vision*
   b. *Speech Recognition*
   c. *Object Recognition*

1.7 Kelebihan AI

Artificial Intelligence mempunyai banyak kelebihan dibandingkan dengan sistem konvensional:

1. Tidak memerlukan persamaan matematik. obyek perancangan sistem konvensional selalu memerlukan persamaan matematik dari objek yang akan diproses, yang mana untuk sistem-sistem linear hal itu masih mudah untuk didapatkan, namun untuk sistem yang non linear maka sangat sulit sekali untuk mendapatkannya. Perancangan sistem kecerdasan buatan cukup memerlukan informasi penalaran dari perilaku proses, yang dapat berupa informasi dalam bentuk bahasa (*linguistic information*)
2. Dapat melakukan proses pembelajaran. Memiliki kemampuan untuk mengambil kesimpulan berdasarkan data masukan-keluaran yang terdahulu.
3. Dapat bersifat adaptif. Memiliki kemampuan untuk mengubah parameter atau mengadaptasi parameter interna sistem secara mandiri.

4. Dapat kokoh terhadap perubahan parameter obyek. Kemampuan menagntisipasi perubahana parameter obyek dengan mengadaptasi parameter internalnya.

C. Rangkuman

1. Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia.
2. Jika dibandingkan dengan kecerdasan alami (kecerdasan yang dimiliki oleh manusia), kecerdasan buatan memiliki beberapa keuntungan secara komersial antara lain:
   a. Kecerdasan buatan lebih bersifat permanen
   b. Kecerdasan buatan lebih mudah diduplikasi & disebarkan.
   c. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami.
   d. Kecerdasan buatan bersifat konsisten.
   e. Kecerdasan buatan dapat didokumentasi.
   f. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami.
   g. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

   Sedangkan keuntungan dari kecerdasan alami adalah:

   a. Kreatif. Kemampuan untuk menambah ataupun memenuhi pengetahuan itu sangat melekat pada jiwa manusia.
   b. Kecerdasan alami memungkinkan orang untuk menggunakan pengalaman secara langsung.
   c. Pemikiran manusia dapat digunakan secara luas, sedangkan kecerdasan buatan terbatas.
3. Aplikasi-aplikasi AI antara lain:

a. *Game playing*
b. Sistem bahasa alami
c. Sistem perancangan dan pembuatan CAD/CAM
d. Sistem pakar reparasi perangkat keras
e. Manajemen data cerdas
f. Sistem otomisasi kantor
g. Analisa kecerdasan militer
h. Kendali dan pemanggilan informasi disk video
i. Kendali robot
j. Analisa programm komputer
k. Diagnosis penyakit
l. Konfigurasi komputer
m. Ramalan senyawa kimia
n. Sistem ucapan
o. Sistem pakar operator komputer
p. Manajemen kendali senjata.

4. Artificial Intellegence dapat diaplikasikan untuk berbagai jenis:
   a. Pengontrol
   b. Peramalan
   c. Perencanaan (*Planning*)
   d. Proses Image (*Image Processing*).
5. Artificial Intelligence mempunyai banyak kelebihan dibandingkan dengan sistem konvensional:
   a. Tidak memerlukan persamaan matematik.
   b. Dapat melakukan proses pembelajaran.
   c. Dapat bersifat adaptif.
   d. Dapat kokoh terhadap perubahan parameter obyek.

D. Tugas

Carilah referensi lain dan buatlah makalah mengenai kecerdasan buatan!

E. Tes Formatif

1. Jelaskan apa yang dimaksud dengan kecerdasan buatan!
2. Jelaskan perbedaan kecerdasan buatan dan kecerdasan alami!
3. Sebutkan dan jelaskan beberapa dari aplikasi kecerdasan buatan yang anda ketahui! (pendapat sendiri).

BAB 2

JARINGAN SYARAF TIRUAN

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan pengertian jaringan syaraf tiruan.
2. Menguraikan sejarah perkembangan jaringan syaraf tiruan.
3. Mengetahui konsep dasar jaringan syaraf tiruan.
4. Menjelaskan istilah-istilah yang ada dalam jaringan syaraf tiruan.

B. Uraian Materi

2.1 Pengertian Jaringan Syaraf Tiruan

Jaringan syaraf tiruan bisa dibayangkan seperti otak buatan di dalam cerita-cerita fiksi ilmiah. Otak buatan ini dapat nerpikir seperti manusia, dan juga sepandai manusia dalam menyimpulkan sesuatu dari potongan-potongan informasi yang diterimanya. Khayalan manusia tersebut mendorong para peneliti untuk mewujudkannya. Komputer diusahakan agar bisa berpikir sama seperti cara berpikir manusia. Caranya adalah dengan melakukan peniruan terhadap aktivitas-aktivitas yang terjadi di dalam sebuah jaringan syaraf biologis.

Ketika manusia berpikir, aktivitas-aktivitas yang terjadi adalah aktivitas mengingat, memahami, menyimpan, dan memanggil kembali apa yang pernah dipelajari oleh otak. Sesungguhnya apa yang terjadi di dalam otak manusia jauh lebih rumit dari apa yang telah disebutkan di atas. Para ahli bedah ootak sering membicarakan mengenai adanya pengatifan neuron, pembuatan koneksi baru, atau pelatihan kembali pola-pola tingkah laku pada otak manusia. Sayangnya hingga saat ini bagaimana sesunggunhnya aktivitas-aktivitas tersebut berlangsung belum ada yang mengetahui secara pasti. Itulah sebabnya mengapa jaringan saraf tiruan hanya mengambil ide dari cara kerja jaringan syaraf biologis.

Salah satu contoh pengambilan ide dari jaringan syaraf biologis adalah adanya elemen-elemen pemrosesan pada jaringan syaraf tiruan yang sering terhubung dan beroperasi secara paralel. Ini meniru jaringan syaraf biologis yang tersusun dari yang tersusun dari sel-sel syaraf (neuron). Cara kerja dari elemen pemrosesan jaringan syaraf tiruan juga sama seperti cara neuron meng-encode informasi yang diterimanya.

Hal yang perlu mendapat perhatian istimewa adalah bahwa jaringan syaraf tiruan tidak diprogram untuk menghasilkan keluaran tertentu. Semua keluaran atau kesimpulan yang ditarik oleh jaringa didasarkan pada pengalaman selama mengikuti proses pembelajaran.

Pada proses pembelajaran, ke dalam jaringan syaraf tiruan dimasukkan pola-pola input (dan output) lalu jaringan akan diajari untuk memmberikan jawaban yang bisa diterima.

2.2 Sejarah

Sejarah dari ditemukannya jaringan syaraf tiruan telah mengalami tahap-tahap perkembangan, antara lain:

- Pada tahun 1990-an, para ilmuwan menemukan bahwa psikologi dari otak sama dengan mode pemrosesan yang dilakukan oleh peralatan komputer.
- Pada tahun 1943, McCulloch dan Pitts merancang model formal yang yang pertama kali sebagai perhitungan dasar neuron.
- Pada tahun 1949, Hebb menyatakan bahwa informasi dapat disimpan dalam koneksi-koneksi dan mengusulkan adanya skema pembelajaran untuk memperbaiki koneksi-koneksi antar neuron tersebut.
- Pada tahun 1954, Farley dan Clark mensetup model-model untuk relasi adaptif simulus-respon dalam jaringan random.
- Pada tahun 1958, Rosenblatt mengembangkan konsep dasar tentang perceptron untuk klasifikasi pola.
- Pada tahun 1960, Widrow dan Hoff mengembangkan ADALINE untuk kendali adaptif dan pencocokan poa yang dilatih dengan aturan pembelajaran *Least Mean Square* (LMS).
- Pada tahun 1974, Werbos memperkenalkan algoritma *backpropogation* untuk melatih perceptron dengan banyak lapisan.
- Pada tahun 1975, Little dan Shaw menggambarkan jaringan syaraf dengan menggunakan model probabilistik.
- Pada tahun 1982, Kohonen mengembangkan metode pembelajaran jaringan syaraf yang tidak terawasi (*unsupervised learning*) untuk pemetaan.
- Pada tahun 1982, Grossberg mengembangkan teori jaringan yang diinspirasi oleh perkembangan psikologi. Bersama

Carpenter, mereka mengenalkan sejumlah arsitektur jaringan, antara lain: *Adaptive Resonance Theory* (ART1), ART2, dan ART3.

- Pada tahun 1982, Hopfield mengembangkan jaringan syaraf *reccurent* yang dapat digunakan untuk menyimpan informasi dan optimasi.
- Pada tahun 1985, algoritma pembelajaran dengan dengan menggunakan mesin Boltzmann yang menggunakan model jringan syaraf probabilistik mulai dikembangkan.
- Pada tahun 1987, Kosko mengembangkan jaringan *Adaptive Bidirectional Associative Memory* (BAM).
- Pada tahun 1988, mulai dikembangkan fungsi radial basis.

2.3 Mengapa Jaringan Syaraf Perlu Dipelajari?

Ada banyak alasan mengapa jaringan syaraf tiruan perlu dipelajari, antara lain:

1. Ada banyaknya teknik (algoritma) jaringan syaraf tiruan yang tersedia. Teknik-teknik yang ada saat ini memiliki arsitektur jaringan syaraf tiruan pada masa-masa awal perkembangan jaringan syaraf tiruan. Pada waktu itu model yang ada sangat sederhana sehingga aplikasinya pun sangat terbatas.
2. Adanya komputer digital berkecepatan tinggi. Hal ini semakin mempermudah proses simulasi jaringan syaraf tiruan.
3. Aplikasi yang sangat luas

Bidang-bidang penelitian yang memanfaatkan jaringan syaraf tiruan diantaranya:

a. Aerospace
Autopilot pesawat terbang, simulasi jalur penerbangan, sistem kendali pesawat, perbaikan autopilot dan simulasi kompponen pesawat.

b. Otomotif
Sistem kendali otomatis mobil

c. Keuangan dan perbankan
Pendekatan uang palsu, evaluator aplikasi kredit, pengidentifikasian pola-pola data pasar saham.

Gambar 2. 1 Pengenalan uang untuk deteksi uang palsu. Pola-pola uang dideteksi menggunakan jaringan syaraf *decision-based* (DBDN). Kotak yang di pinggir menunjukkan fitur tekstur mata uang sementara kotak yang diujung menunjukkan lokasi pola-pola uangnya.

d. Pertahanan (militer)

Pengendali senjata, pendeteksi bom, penelusuran target, pembedaan objek, pengendali sensor, sonar, radar, dan pengolahan sinyal citra yang meliputi kompresi data, ekstraksi bagian istimewa dan penghilangan derau, pengenalan sinyal atau citra.

e. Elektronik

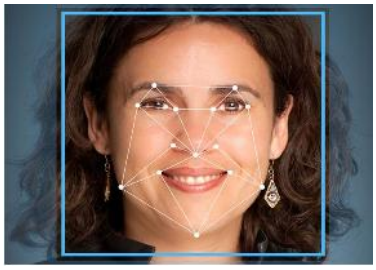
Pembuatan perangkat keras yang bisa mengimplementasikan jaringan syaraf tiruan secara efisien (pendesainan VLSI), *machine vision*, pengontrol gerakan dan penglihatan robot, sintesis suara.

f. Broadcast

Pencarian klip berita melalui pengenalan wajah.

g. Keamanan

Jaringan syaraf tiruan digunakan untuk mengenali mobil dan mengenali wajah oknum.



Gambar 2. 2 Hasil online situasi pengenalan wajah. Daerah fitur ditandai dengan kotak dan posisi deteksi terakhir akan ditandai dengan lingkaran.

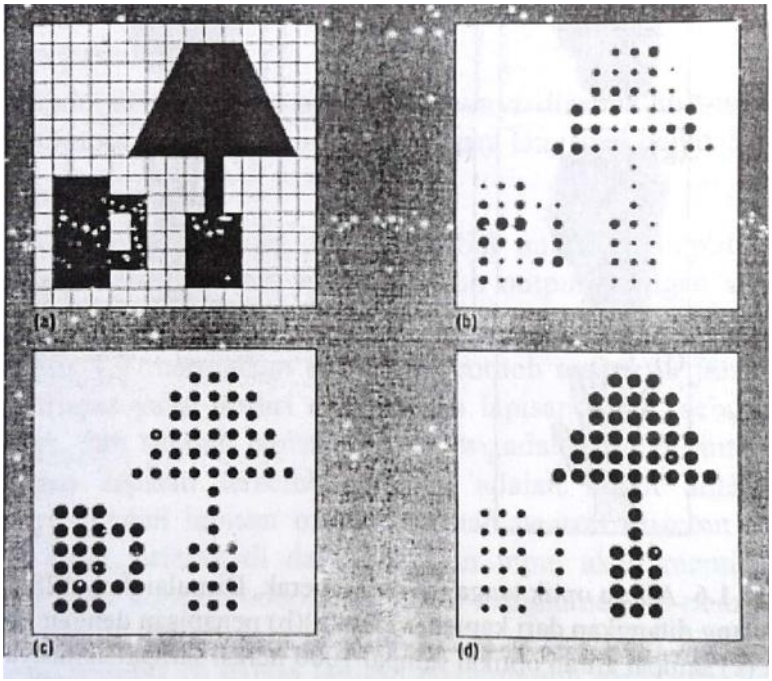
h. Medis
Analisis sel kanker payudara, pendeteksi kanker kulit.

i. Pengenalan suara
Pengenalan percakapan, klasifikasi suara.

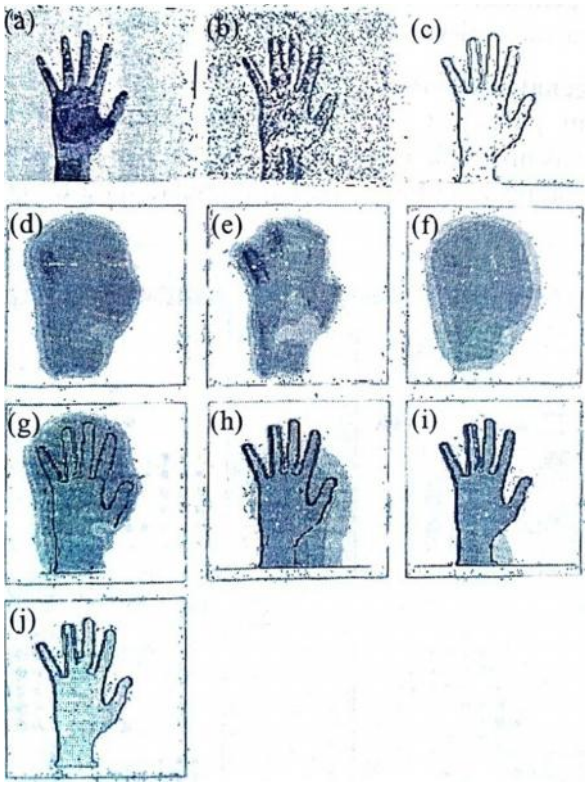
j. Pengenalan tulisan
Pengenalan tulisan tangan, penerjemahan tulisan ke dalam tulisan latin.

k. Matematika
Alat pemodelan masalah dimana bentuk ekplisit dari hubungan antara variabel-variabel tertentu tidak diketahui.

l. Pengenalan benda bergerak
Selain pola citra diam, jaringan syaraf tiruan juga bisa digunakan untuk mendeteksi citra bergerak dari video seperti citra orang yang bergerak, citra tangan yang bergerak, dan lain-lain.



Gambar 2. 3 Segmentasi pola citra-citra yang terhubung (a) Citra cangkir dan lampu meja disajikan dala citra berukuran 15x15 dari osilator; (b) (c) (d) melambangkan penyajian citra sesuai dengan aktivitas osilator. (b) Hasil tangkapan aktivitas jaringan syaraf tiran beberapa saat setelah osilator jaringan syaraf tiruan mulai bekerja. (c) Hasil tangkapan setelah beberapa waktu memperlihatkan efek pengelompokan. Osilator-osilator yang menjadi milik dari objek yang sama memiliki aktivitas yang hampir sama. Juga terlihat bahwa aktivitas osilator cangkir lebih tinggi dari pada aktivitas osilator lampu meja. (d) hasil tangkapan di mana osilator lampu meja mancapai aktivitas tinggi sedangkan osilator cangkir aktivitasnya di bawah aktivitas osilator lampu meja. Model osilator ini bisa diinterpretasikan secara biologis sebagai pendekatan terhadap sel-sel syaraf yang terangsang dan terhambat.



Gambar 2. 4 Aliran optik tangan yang bergerak. Dimulai dari 128x128 citra piksel yang ditangkap dari kamera video (a)(b) penapisan dan Lapacian Gaussian (c) sampai aliran optikal sampai 1000 iterasi sampai akhirnya diperoleh citra terakhir di mana jari-jari tangan memiliki kecepatan yang lebih cepat dari tangan sehingga dibutuhkan beberapa siklus untuk memutus gerakan bersambung untuk menangkap sekitar garis luar tangan. Jaringan yang digunakan adalah jaringan resistif.

m. Lain-lain

Jaringan syaraf tiruan juga bisa digunakan sebagai detektor virus komputer, penginderaan bau, pengenalan pola-pola unik dala penambangan data (*data mining*), dan mendukung sistem pendukung keputusan (DDS- *Descission Support System*).

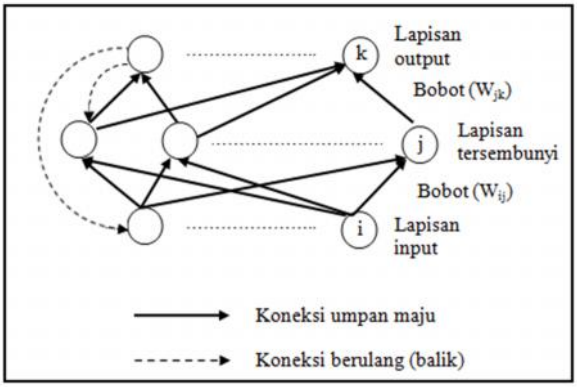
2.4 Konsep dasar jaringan syaraf tiruan

Pembagian arsitektur jaringan syaraf tiruan bisa dilihat dari kerangka kerja dan skema interkoneksi. Kerangka kerja jaringan syaraf tiruan bisa dilihat dari jumlah lapisan (*layer*) dan jumlah node pada setiap lapisan.

Lapisan-lapisan penyusun jaringan syaraf tiruan dapat dibagi menjadi tiga, yaitu:

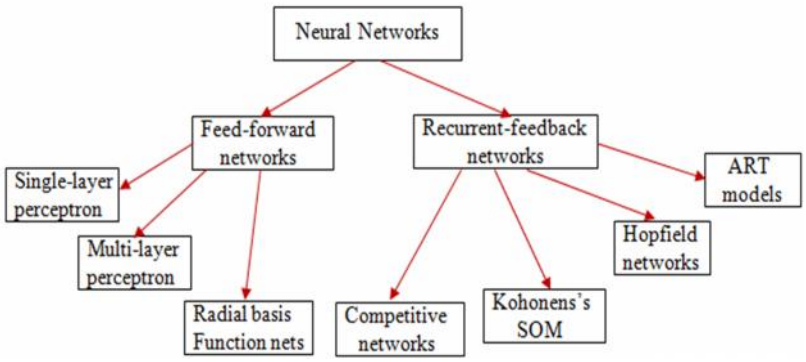
1. **Lapisan input**
   Node-node di dalam lapisan input disebut unit-unit input. Unit-unit input menerima input dari dunia luar. Inpu yang dimasukkan merupakan penggambaran suatu masalah.
2. **Lapisan tersembunyi**
   Node-node di dalam lapisan tersembunyi disebut unit-unit tersembunyi. Output dari lapisan ini tidak secara langsung dapat diamati.
3. **Lapisan output**
   Node-node pada lapisan output disebut unit-unit output. Keluaran atau output dari lapisan ini merupakan output jaringan syaraf tiruan terhadap suatu permasalahan.

Gambar 2.5 merupakan salah satu contoh arsitektur jaringan syaraf tiruan multilapis yang terdiri dari sebuah lapisan input, sebuah lapisan tersembunyi, dan sebuah lapisan output. W_{ij} adalah bobot antara lapisan input dengan lapisan tersembunyi, W_{jk} adalah bobot antara lapisan tersembunyi dengan lapisan output. Sebuah neuron (disebut juga node atau unit) yang terletak di dalam lapisan input akan memiliki fungsi aktivasi dan pola koneksi bobot yang dengan neuron-neuron lainnya yang terletak didalam lapisan input. Demikian pula halnya sebuah neuron yang yang terletak di dalam lapisan tersembunyi akan memiliki aktivasi dan pola koneksi bobot yang sama dengan nneuron-neuron lainnya yang terletak di dalam lapisan tersembunyi. Demikian seterusnya, untuk neuron-neuroonpada lapisan output juga berlaku hal yang sama.



Gambar 2. 5 Sebuah jaringan syaraf tiruan multilapis (multilayer neural networks)

Mengenai pengelompokan jaringan syaraf tiruan, ada pula yang membagi ke dalam dua kelompok, yaitu jaringan syaraf tiruan umpan maju (*feed-forward networks*) dan jaringan syaraf tiruan berulang/umpan balik (*Recurrent/feedback networks*). Jaringan syaraf tiruan umpan maju adalah graf yang tidak mempunyai loop, sedangkan jaringan syaraf tiruan berulang/umpan balik dicirikan dengan adanya loop-loop koneksi balik. Jika mengikuti aturan ini, taksonomi jaringan syaraf tiruan akan tampak seperti pada gambar 2.6.



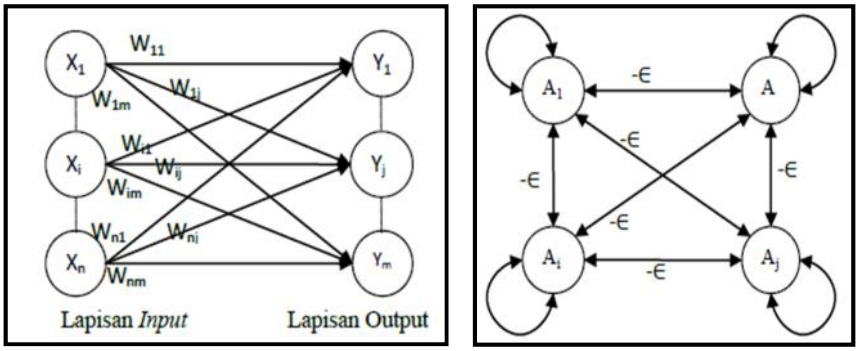
Gambar 2. 6 Taksonomi jaringan syaraf tiruan beserta nama-nama algoritma pembelajaran yang termasuk ke dalamnya

Kembali ke gambar 2.5, yang digambarkan dengan panah dengan garis terputus-putus adalah koneksi balik (*recurrent networks* atau *feedback networks*). Ciri dari jaringan syaraf tiruan yang memiliki arsitektur ini adalah adanya loop-loop koneksi umpan balik. Sedangkan tanda panah dengan garis tidak terputus-putus merupakan jaringan syaraf tiruan koneksi umpan maju (feed forward networks) yang bergerak maju dan tidak memiliki loop. Dengan menerapkan fungsi aktivasi ke dalam bobot dan input dilakukan perhitungan yang hasilnya dianggap sebagai sinyal berbobot yang diteruskan ke lapisan di atasnya. Sinyal berbobot inilah yang menjadi input bagi lapisan di atasnya. Kemudian diterapkan lagi fungsi aktivasi pada lapisan ini untuk menghitung output jaringan. Proses ini akan berulang sampai kondisi berhenti terpenuhi.

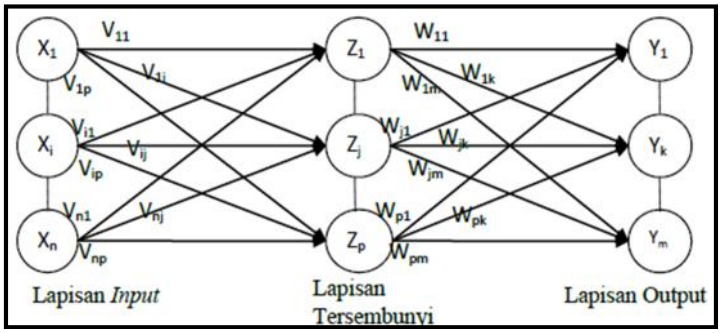
Pendapat lain mengenai struktur jaringan syaraf tiruan adalah sebagai berikut: jaringan syaraf tiruan dibagi menjadi 3 arsitektur, yaitu:

1. Jaringan lapis tunggal
   Jaringan yang memiliki arsitektur jenis ini hanya memiliki satu buah lapisan bobot koneksi. Jaringan lapisan tunggal terdiri dari unit-unit input yang menerima sinyal dari dunia luar, dan unit-unit output dimana kita bisa membaca respons dari jaringan syaraf tiruan tersebut.
2. Jaringan multilapis
   Jaringan multilapis merupakan jaringan dengan satu atau lebih lapisan tersembunyi. Multilayer net ini memiliki kemampuan lebih dalam memecahkan masalah bila dibandingkan dengan singel layer net, namun pelatihannya mungkin lebih rumit.
3. Jaringan kompetitif
   Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi akttif.

Untuk lebih jelasnya, ketiga macam arsitektur jaringan syaraf tiruan ini dapat dilihat pada gambar berikut:



(a) (b)



(c)

Gambar 2. 7 Jaringan syaraf tiruan

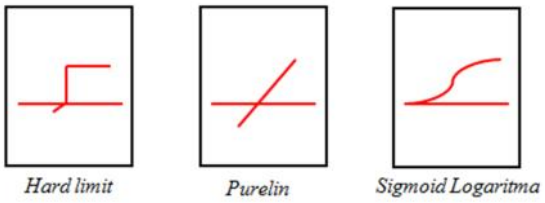
(a) Jaringan lapis tunggal (*single-layer net*)
(b) Jaringan kompetitif (*competitive layer*)
(c) Jaringan multilapis (*multilayer net*)

2.5 Istilah-istilah Jaringan Syaraf Tiruan

Berikut ini beberapa istilah jaringan syaraf tiruan yang sering ditemui:

a. **Neuron atau node atau unit:** sel syaraf tiruan yang merupakan elemen pengolahan jaringan syaraf tiruan. Setiap neuron menerima data input, memproses input tersebut, dan mengirimkan hasilnya berupa sebuah output.

b. **Jaringan:** kumpulan neuron yang saling terhubung dan membentuk lapisan.
c. **Input dan masukan:** berkorespon dengan sebuah atribut tunggal dari sebuah pola atau data lain dari dunia luar. Sinyal-sinyal input ini kemudian di teruskan ke lapisan selanjutnya.
d. **Output dan keluaran:** solusi atau hasil pemahaman jaringan terhadap data input. Tujuan pembangunan jaringan syaraf tiruan sendiri adalah untuk mengetahui nilai output.
e. **Lapisan tersembunyi (hidden layer):** lapisan yang tidak secara langsung berinteraksi dengan dunia luar.
f. **Bobot:** bobot dalam jaringan syaraf tiruan merupakan nilai matematis dari koneksi, yang mentransfer data dari satu lapisan ke lapisan lainnya. Bobot ini digunakan untuk mengatur jaringan sehingga jaringan syaraf tiruan bisa menghasilkan output yang diinginkan sekaligus bertujuan membuat jaringan tersebut belajar.
g. **Summantion function:** fungsi yang digunakan untuk mencari rata-rata bobot dari semua elemen input.
h. **Aktivasi atau fungsi transfer:** fungsi yang menggambarkan hubungan hubungan antara tingkat aktivasi internal (*summation function*) yang mungkin membentuk linear atau nonlinear. Beberapa fungsi jaringan syara tiruan diantaranya: hard imit, purelin, dan sigmoid. Yang populer digunakan dalah fungsi sigmoid yang memiliki beberapa varian: sigmoid logaritma, sigmoid biner, sigmoid bipolar, dan sigmoid tangen.



i. **Paradigma pembelajaran:** cara berlangsungnya pembelajaran atau pelatihan jaringan syaraf tiruan, apakah terawasi

(supervised learning), tidak terawasi (unsupervised learning), atau merupakan gabungan keduanya (*hybrid*).
Pada pembelajaran terawasi, kumpulan input yang digunakan, output-outputnya telah diketahui. Perbedaan antara output-output yang digunakan untuk mengoreksi bobot jaringan syaraf tiruan dapat menghasilkan jawaban sedekat (semirip) mungkin dengan jawaban yang benar yang telah diketahui oleh jaringan syaraf tiruan.

Pada pembelajaran tak terawasi, atau pembelajaran tanpa guru, jaringan syaraf mengorganisasi dirinya sendiri untuk membentuk vektor-vektor input yang serupa tanpa menggunakan data atau contoh-contoh pelatihan. Struktur menggunakan dasar data atau korelasi antara pola-pola data yang dieksplorasi. Paradigma pembelajaran ini mengorganisasi pola-pola ke dalam kategori-kategori berdasarkan korelasi yang ada.

Paradigma pembelajaran hibrida merupakan kombinasi dari kedua aturan di atas. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi.

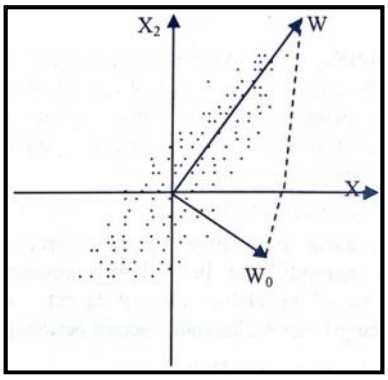
j. Aturan pembelajaran: Aturan kerja secara umum dari teknik/ algoritma jaringan saraf tiruan. Ada 4 tipe dasar aturan pembelajaran, yaitu aturan pengoreksian kesalahan (*error correcting*}, aturan Boltzmann, aturan Hebbian, dan aturan pembelajaran kompetitif (competitive learning).

1. Aturan Pengoreksian Error
   Prinsip dasar dari aturan pembelajaran pengoreksian error adalah memodifrkasi bobot-bobot koneksi dengan menggunakan sinyal kesalahan (output target - output aktual) untuk mengurangi besamya kesalahan secara bertahap.
2. Aturan Pembelaj aran Boltzmann
   Mesin Boltzmann merupakan jaringan saraf tiruan balik yang simetris, terdiri dari unit-unit biner (+1 dan -1, masing-masing untuk *on* dan *off*). Dengan

kesimetrisannya, bobot pada koneksi dari unit i ke unit j sama dengan bobot koneksi dari unit j ke unit i ($w_{ij} = w_{ji}$). Setiap neuron pada mesin Boltzmann merupakan sebuah unit stokastik yang menghasilkan output menurut distribusi mekanik statistik Boltzmann.

Aturan Boltzmann juga dapat dikatakan sebagai kasus lain dari aturan pembelajaran pengoreksian error. Bedanya, kesalahan (*error*) diukur bukan sebagai perbedaan langsung antara output aktual dengan output yang diinginkan, melainkan sebagai perbedaan antara korelasi output-output dari 2 buah neuron dalam kondisi operasi *clamped* dan *free-running*. Pada *clamped*, neuron-neuron visibel ditahan pada keadaan-keadaan tertentu. Pada *free-running*, baik neuron visibel maupun hidden dapat beroperasi dengan bebas. Neuron-neuron yang berinteraksi dengan lingkungan disebut neuron yang visibel, sedangkan neuron-neuron yang tidak berinteraksi dengan lingkungan disebut neuron tersembunyi (*hidden neurons*).

3. Aturan Hebbian
   Kekuatan koneksi antara 2 buah neuron akan meningkat jika kedua neuron memiliki tingkah laku yang sama (keduanya memiliki aktivasi positif atau keduanya memiliki aktivasi negatif).



Gambar 2. 8 Orientasi selektivitas sebuah neuron tunggal yang dilatih menggunakan aturan Hebbian

Gambar 2.8 menunjukkan keselektifan orientasi dari sebuah neuron tunggal yang dilatih dengan menggunakan aturan Hebbian. Titik-titik yang digambarkan berasal dari distribusi Gauss 2 dimensi dan digunakan untuk melatih neuron. Vektor bobot pada neuron diinisialisasi sebagai w_0 seperti ditunjukkan dalam gambar. Selama pelatihan berlangsung, vektor bobot bergerak semakin dekat ke arah variansi maksimal data (w). w merupakan vektor eigen matriks kovariansi data yang bersesuaian dengan nilai eigen terbesar.

4. Aturan Pembelajaran Kompetitif

   Unit-unit output pada aturan pembelajaran kompetitif ini harus saling bersaing untuk beraktivasi. Jadi hanya satu unit output yang aktif pada satu waktu. Fenomena ini dikenal sebagai *winner-take-all*. Bobot-bobotnya diatur setelah satu node pemenang terpilih.

   Pada Tabel 1.1 diperlihatkan paradigma, aturan pembelajaran, arsitektur, algoritma pembelajaran jaringan syaraf tiruan beserta bidang aplikasinya yang sudah umum diketahui.

Tabel 2.1 Paradigma, aturan pembelajaran, arsitektur, algoritma dan aplikasi jaringan syaraf tiruan

Paradigma	Aturan Pembelajaran	Arsitektur	Algoritma Pembelajaran	Bidang Kerja
Terawasi (supervised)	Pengoreksian error	Perceptron lapis tunggal atau multilapis	- Perceptron - Propagasi Balik - Adaline dan Madaline	- Klasifikasi pola - Aproksimasi fungsi - Prediksi, kontrol
	Boltzman	Recurrent (berulang)	Boltzmann	Klasifikasi pola
	Hebbian	Umpan maju multilapisan (multilayer feedforward)	Analisis diskriminan linear	- Analisis data - Klasifikasi pola
	Kompetitif	Kompetitif	LVQ (Learning Vector Quantization)	- Kategori kelas - Kompresi data
		Jaringan ART	ARTMap	- Klasifikasi pola - Kategori kelas
Tidak terawasi (unsupervised)	Pengkoreksian error	Multilayer feedforward	Proyeksi sammon	Analisis data
	Hebbian	Umpan maju atau kompetitif	Analisis komponen prinsipal	- Analisis data - Kompresi data
		Hopfield net	Aturan memori asosiatif	Memori asosiatif
	kompetitif	kompetitif	Kuantisari vektor	- Kategorisasi - Kompresi data
				- Kategorisas

		Kohonen SOM Jaringan ART	Kohonen SOM ART1, ART2	- Analisis data Kategorisasi
Hibrida (hybrid)	Pengkoreksian error dan kompetitif	Jaringan RBF	Algoritma RBF	- Klasifikasi pola - Aproksimasi fungsi - Prediksi, kontrol

2.6 Algoritma Umum Jaringan Syaraf Tiruan

Algoritma pembelajaran/pelatihan jaringan syaraf tiruan:
Dimasukkan n contoh pelatihan ke dalam jaringan syaraf tiruan.
Lakukan:

Langkah 1

1. Inisialisasi bobot-bobot jaringan. Set I = 1.
2. Masukkan contoh ke-i (dari sekumpulan contoh pembelajaran yang terdapat dalam set pelatihan) ke dalam jaringan pada lapisan input.
3. Cari tingkatan aktivasi unit-unit output menggunakan algoritma aplikasi.

   If kinerja jaringan memenuhi standar yang ditentukan sebelumnya (memenuhi syarat berhenti)

   Then exit.
4. Update bobot-bobot dengan menggunakan aturan pembelajaran jaringan.
5. *If* i = n, *then* *reset* i = 1
   Else i = i – 1.

Ke langkah 2

Algoritma aplikasi/inferensi jaringan syaraf tiruan:
Dimasukkan sebuah contoh pelatihan ke dalam jaringan syaraf tiruan.

Lakukan:

1. Masukkan kasus ke dalam jaringan pada lapisan input.
2. Hitung tingkat aktivasi node-node jaringan.
3. Untuk jaringan koneksi umpan maju, jika tingkat aktivasi dari semua unit outputnya telah dikalkulasi, maka *exit*. Untuk jaringan koneksi balik, jika tingkat aktivasi dari semua unit output menjadi konstan atau mendekati konstan, maka *exit*. J jika tidak, kembali ke langkah 2, jika jaringannya tidak stabil, maka *exit* dan *fail*.

C. Rangkuman

1. Ada banyak alasan mengapa jaringan syaraf tiruan perlu dipelajari, antara lain:
   a. Ada banyaknya teknik (algoritma) jaringan syaraf tiruan yang tersedia. Teknik-teknik yang ada saat ini memiliki arsitektur jaringan syaraf tiruan pada masa-masa awal perkembangan jaringan syaraf tiruan. Pada waktu itu model yang ada sangat sederhana sehingga aplikasinya pun sangat terbatas.
   b. Adanya komputer digital berkecepatan tinggi. Hal ini semakin mempermudah proses simulasi jaringan syaraf tiruan.
   c. Aplikasi yang sangat luas.
2. Lapisan-lapisan penyusun jaringan syaraf tiruan dapat dibagi menjadi tiga, yaitu:
   a. Lapisan input
   b. Lapisan tersembunyi .
   c. Lapisan output
3. Istilah-istilah dalam jaringan syaraf tiruan, antara lain:
   a. Neuron atau node atau unit
   b. Jaringan
   c. Input dan masukan
   d. Output dan keluaran
   e. Lapisan tersembunyi (hidden layer)

f. Bobot
g. Summantion function
h. Aktivasi atau fungsi transfer
i. Paradigma pembelajaran
j. Aturan pembelajaran

D. Tugas

Buatlah kesimpulan mengenai jaringan syaraf tiruan yang telah dibahas pada bab 2 ini !

E. Tugas Formatif

1. Berikan pengertian jaringan syaraf tiruan!
2. Sebutkan dan jelaskan kelebihan dari jaringan syaraf tiruan!
3. Menurut anda bagaimanakah perbandingan jaringan syaraf tiruan dengan konvensional?
4. Sebutkan 3 jenis arsitektur jaringan syaraf tiruan?

BAB 3

PERCEPTRON

A. Tujuan Pembelajaran

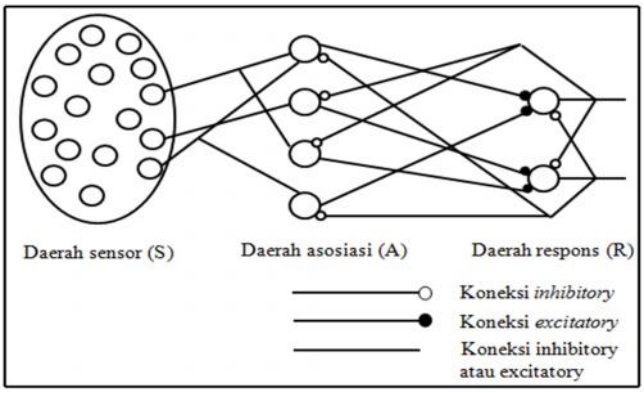
Setelah mempelajari uraian materi berikut, diharapkan mahasiswa dapat :

1. Menjelaskan dan memberikan contoh arsitektur, algoritma, dan contoh aplikasi perceptron lapis tunggal.
2. Menjelaskan mengenai perceptron multilapis.
3. Menguraikan keterbatasan-keterbatasan yang ada pada perceptron lapis tunggal.

B. Uraian Materi

3.1 Perceptron Lapis Tunggal

Teknik perceptron ditemukan oleh psikolog bernama Frank Rosenblatt di penghujung tahun 1950-an. Teknik ini merupakan pemodelan sederhana dari mata manusia. Sebuah *photoperceptron* yang berespon terhadap pola-pola optik digambarkan seperti pada gambar 3.1. *photoperceptron* menerima cahaya pada titik sensor (S) dari struktur retinanya. Impuls-impuls yang dibangkitkan oleh titik S kemudian dikirim ke unit-unit asosiator (A) pada lapisan asosiasi.



Gambar 3.1 Sebuah perceptron sederhana

Setiap unit A terhubung secara acak dengan sekumpulan titik-titik S yang dengan koneksi tersebut mungkin terangsang (*excitatory*) atau terlihat (*inhibitory*) dengan kemungkinan nilai-nilainya adalah +1, -1 dan 0. Nilai untuk koneksi *excitatory*, -1 untuk koneksi *inhibitory* maupun *excitatory*. Unit A akan aktif bila jumlah dari input-inputnya melampaui nilai ambang, lalu ia akan menghasilkan output yang dikirim ke lapisan respons.

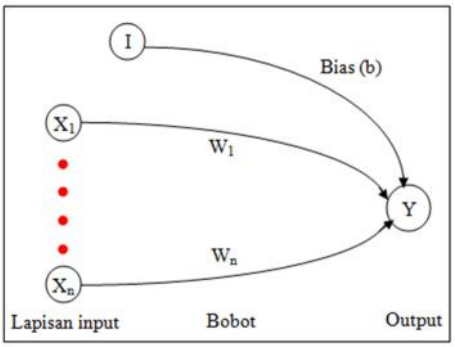
Unit-unit pada lapisan respons bekerja dengan cara yang serupa dengan unit-unit pada lapisan sensor namun ada tambahan koneksi *inhibitory* antar unit-unit R. jika nilai imputnya

melampaui nilai ambang maka output yang dihasilkan adalah 1, selain itu berikankan nilai output -1.

Pola yang serupa akan membangkitkan unit *respons* R yang sama. Hal ini dapat dimanfaatkan untuk mengklasifikasikan pola-pola ke dalam kategori-kategori tertentu. Caranya adalah dengan melakukan pemisahan kelompok berbeda, yaitu yang dikenal dengan sebutan pemisahan linear (*linear separability*).

3.1.1 Arsitektur

Perceptron lapis tunggal dapat dikatakan salah satu teknik jaringan syaraf tiruan yang sederhana. Teknik ini hanya mempunyai sebuah lapisan input dan sebuah unit output seperti tanpak pada Gambar 3.2. pada gambar tersebut terdapat bias (b), yaitu unit yang aktivasinya selalu 1 dan berperilaku sebagai layaknya bobot (W).



Gambar 3.2 Arsitektur perceptron lapis tunggal

3.1.2 Algoritma Pelatihan

Algoritma pelatihan digunakan untuk melatih jaringan syaraf tiruan, yaitu dengan cara mengajarinya dengan contoh-contoh kasus/pola sampai jaringan syaraf tiruan berhasil mengenali pola tersebut. Setiap kali output yang dihasilkan jaringan tidak sesuai dengan target yang diharapkan maka setiap kali pula bobotnya di-update. Hal ini terus-menerus dilakukan sampai tidak ada lagi bobot yang berubah untuk setiap pasangan

latihan sensor dan terget. Bobot-bobot terakhir yang diperoleh pada saat pelatihan jaringan syaraf tiruan inilah yang akan digunakan pada saat pengaplikasian (dengan menggunakan algoritma aplikasi perceptron).

Algoritma pelatihan perceptron adalah sebagai berikut:

Langkah 0. Inisialisasi bobot-bobot dan bias.
Tentukan angka pembelajaran α ($0<\alpha\leq 1$)
Tentukan nilai ambang θ($0 < \theta\leq 1$)

Langkah 1. Ulangi

Langkah 2. Untuk setiap pasangan latihan **s:t**, lakukan

Langkah 3. Tentukan aktivasi unit-unit input:

$$x_i = s_i$$

Langkah 4. Hitung respons dari unit output:

$$y_{in} = b + \sum_n x_i$$

$$y = \begin{cases} 1 \ \ jika \quad y_{in} > \ ^{-} & ....(3-1) \\ 0 \ \ jika \ ^{-} \leq y_{in} \leq \ ^{-} & ....(3-2) \\ -1 \ jika \quad y_{in} < \ ^{-} & ....(3-3) \end{cases}$$

Langkah 5. Updatelah bobot-bobot dan bias jika error terjadi pada pola y

Jika $y \neq t$ maka
W_1(baru) = w_1(lama) + α t x_i,
B (baru) = b (lama) + α t

Selain itu
w_1(baru) = w_1(lama)
b (baru) = b (lama)

Langkah 6. Sampai kondisi berhenti terpenuhi.
Kondisi berhenti adalah kondisi di mana tidak terdapat bobot yang berubah pada langkah 2.

Keterangan:

s	sensor
t	target
x_i	unit input ke-i
s_i	unit sensor ke-i
w_i	bobot ke-i
b	bias
y	unit respons (output)
α	angka pembelajaran
θ	nilai ambang
i	1,..., n di mana n adalah banyaknya unit input

Pada saat penginilisasian, sebaiknya bobot-bobot dan bias diset sama dengan 0, sedangkan untuk angka pembelajaran (α) diset sama dengan 1.

Sekarang mari kita memperhatikan langkah 4 yaitu langkah untuk menghitung respons dari unit output. Perhatikan persamaan 2-1. Ini berarti respons positif akan diberikan jika b + Σ x_i w_i lebih besar dari nilai θ. Dengan demikian bila untuk 2 buah unit input, pernyataan diatas bisa dituliskan kembali sebagai:

$$x_i w_i + x_2 w_2 + b > -$$

dengan garis pembatasnya

$$x_2 = \frac{-x_1 w_1 - b + -}{w_2}$$

atau

$$x_2 = \frac{-w_1}{w_2} \quad x_1 - \frac{b - -}{w_1}$$

untuk mendapatkan daerah dengan respons negatif (lihat persamaan 2-3), jaringan harus memenuhi pertidaksamaan:

$$x_2 = \frac{-x_1 w_1 - b - -}{w_2}$$

atau

$$x_2 = \frac{-w_1}{w_2} x_1 - \frac{b + \theta}{w_2}$$

pada pertidak samaan di atas terlihat ada sebuah daerah yang tidak menghasilkan baik respons positif maupun respons negatif. Ini memenuhi persamaan 3-2, yaitu $-\theta \leq x_1w_1 + x_2w_2 + b \leq \theta$. Daerah tersebut dinamakan *undecided bound* yang memisahkan daerah respons positif dengan daerah respons negatif.

Pada teknik pereceptron juga dikenal istilah *linear separability*, yaitu kemapuan jaringan syaraf untuk memisahkan pola-pola pelatihan berdasarkan kelompoknya dengan menggunakan sebuah garis lurus, bidang, atau *hyperplane* yang memiliki persamaan linear yang mendefenisikannya. Sebagai contoh adalah ruang berdimensi n -1 (*hyperplane*). Ruang berdimensi n-1 pemisahnya berupa sebuah permukaan objek berdimensi n-2. Begitu seterusnya, sampai ke bidang yang memiliki pemisah berupa sebuah garis.

3.1.3 Algoritma Aplikasi Perceptron

Langkah 0. Terapkan algoritma pelatihan untuk mengeset bobot-bobot.

Langkah 1. Untuk setiap vektor input x yang ingin diklasifikasikan, lakukan langkah 3-3.

Langkah 2. Set aktivasi unit-unit input (x_i), I = 1,...,n

$$x_1 = s_1$$

langkah 3. Hitung respons unit output (y):

$$y_{in} = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & jika \quad y_{in} > \theta \\ 0 & jika \quad -\theta \leq y_{in} \leq \theta \\ -1 & jika \quad y_{in} < \theta \end{cases}$$

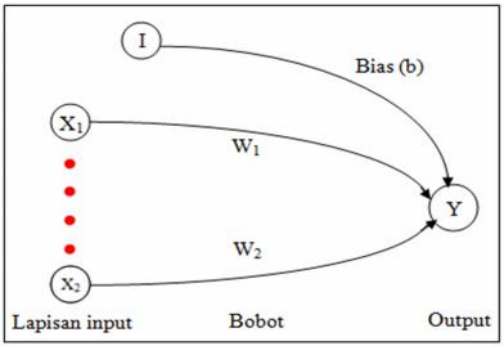
3.1.4 Aplikasi

Berikut ini salah satu contoh aplikasi perceptron untuk pengenalan fungsi logika OR. Tabel fungsi logika OR (dalam biner) terlihat pada tabel 3.1

Tabel 3.1 Fungsi logika OR

INPUT (x_1 x_2 x_3)	TARGET
(1 1 1)	1
(1 0 1)	1
(0 1 1)	1
(0 0 1)	0

Sementara arsitektur perceptron yang digunakan pada percobaan ini tampak seperti pada Gambar 3.3.



Gambar 3.3 Arsitektur perceptron lapis tunggal untuk fungsi logika OR

Langkah persiapan jaringan syaraf tiruan (perceptron) adalah sebagai berikut: pertama-tama kita perlu menginisialisasi jaringan terlebih dahulu. Angka pembelajaran (α) yang diambil adalah 1 sementara bobot w_1 dan w_2 serta bias b kesemuanya diberi nilai awal 0. Nilai ambang (*treshold* θ) ditetapkan sebesar 0,1.

Ada tiga cara representasi data yang dilakukan, yaitu:
a) Baik input maupun target berbentuk biner.
b) Input berbentuk biner, target berbentuk bipolar.
c) Baik input maupun target berbentuk bipolar.

Hasil Percobaan Pengenalan Fungsi Logika OR
a) Input : biner
Target : biner

Tabel 3.2 Pengenalan fungsi ogika OR dengan input dan target biner

Counter	INPUT (x_1 x_2 x_3)	NET Y_n	OUTPUT y	TARGET t	BOBOT (w_1 w_2 w_3)	Perubahan bobot (Δw_1 Δw_2 Δw_3)	keterangan
					(0 0 0)		inisialisasi
							Epoch ke-1
1	(1 1 1)	0	0	1	(1 1 1)	(1 1 1)	
2	(1 0 1)	2	1	1	(1 1 1)	(0 0 0)	y=t bobot tetap
3	(0 1 1)	2	1	1	(1 1 1)	(0 0 0)	y=t
4	(0 0 1)	1	1	0	(1 1 1)	(0 0 0)	
							Epoch ke-2
5	(1 1 1)	3	1	1	(1 1 1)	(0 0 0)	y=t
6	(1 0 1)	2	1	1	(1 1 1)	(0 0 0)	y=t
7	(0 1 1)	2	1	1	(1 1 1)	(0 0 0)	y=t
8	(0 0 1)	1	1	0	(1 1 1)	(0 0 0)	stagnasi

Keterangan:
Epoch adalah satu kali presentasi yang mencakup semua pola pelatihan.

Perhatikan Tabel 3.2 di atas. Dari Tabel 3.2 terlihat bahwa sampai epoch kedua habis. Bobot dan biasnya konstan,

tidak mengalami perubahan. Oleh karena itu dapat dipastikan bahwa jaringan mengalami stagnasi. Bahkan bila perhitungan terus dilakukan maka output jaringan (Y) tidak akan pernah sama dengan target (T) karena Y tidak pernah berubah. Perhatikan langkah 5 pada algoritma pelatihan perceptron. Nilai target 0 pada percobaan di ataslah yang mengakibatkan terjadinya kemandekan tersebut. Oleh sebab itu penggunaan input dan target biner pada kasus fungsi logika OR (atau fungsi-fungsi lain yang melibatkan target bernilai 0) tidak disarankan karena jaringan tidak akan pernah mengenali semua pola yang diajarkan kepadanya.

b) Input : biner
Target : bipolar

Tabel 3.3 Perccobaan pengenalan fungsi logika OR dengan input berbentuk biner dan target berbentuk bipolar

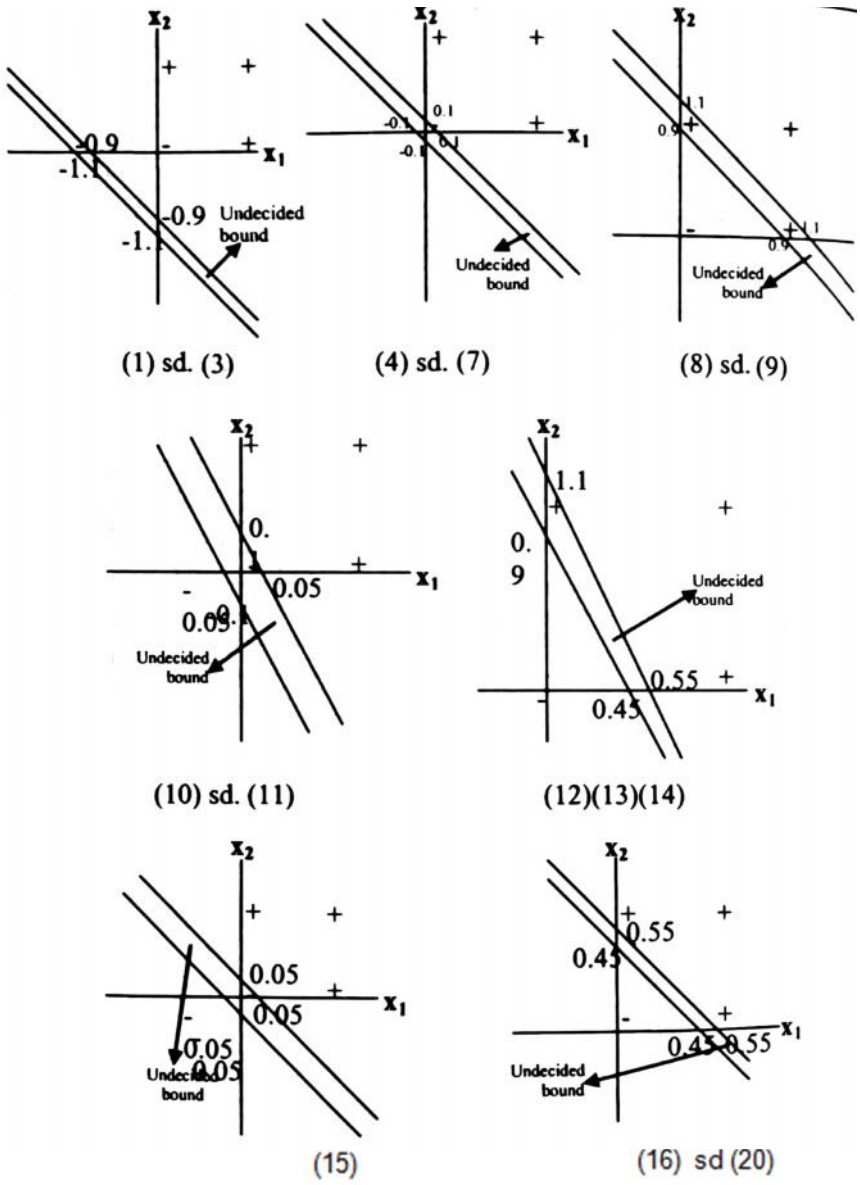
Co unt- er	INPUT (x_1 x_2 1)	NET Y_n	OUTPUT y	TARGET t	BOBOT (w_1 w_2 b)	Perubahan bobot (Δw_1 Δw_2 Δb)	keterangan
					(0 0 0)		Inisialisasi
							Epoch ke-1
1	(1 1 1)	0	0	1	(1 1 1)	(1 1 1)	
2	(1 0 1)	2	1	1	(1 1 1)	(0 0 0)	y=t bobot tetap
3	(0 1 1)	2	1	1	(1 1 1)	(0 0 0)	y=t
4	(0 0 1)	1	1	-1	(1 1 0)	(0 0 -1)	
							Epoch ke-2
5	(1 1 1)	2	1	1	(1 1 0)	(0 0 0)	y=t
6	(1 0 1)	1	1	1	(1 1 0)	(0 0 0)	y=t
7	(0 1 1)	1	1	1	(1 1 0)	(0 0 0)	y=t
8	(0 0 1)	0	0	-1	(1 1 -1)	(0 0 0)	
							Epoch ke-3
9	(1 1 1)	1	1	1	(1 1 -1)	(0 0 0)	y=t
10	(1 0 1)	0	0	1	(2 1 0)	(1 0 1)	
11	(0 1 1)	1	1	1	(2 1 0)	(0 0 0)	y=t
12	(0 0 1)	0	0	-1	(2 1 -1)	(0 0 -1)	
							Epoch ke-4
13	(1 1 1)	2	1	1	(2 1 -1)	(0 0 0)	y=t
14	(1 0 1)	1	1	1	(2 1 -1)	(0 0 0)	y=t
15	(0 1 1)	0	1	1	(2 0 0)	(0 1 1)	
16	(0 0 1)	0	0	-1	(2 2 -1)	(0 0 -1)	
							Epoch ke-5
17	(1 1 1)	3	1	1	(2 2 -1)	(0 0 0)	y=t
18	(1 0 1)	1	1	1	(2 2 -1)	(0 0 0)	y=t

19	(0 1 1)	1	1	1	(2 2 -1)	(0 0 0)	y=t
20	(0 0 1)	-1	-1	-1	(2 2 -1)	(0 0 0)	y=t; sukses

Simak Tabel 3.3. perhatikan bahwa jaringan perceptron berhasil mengenali fungsi logika OR pada epoch ke-5. Pergeseran batasan keputusan fungsi logika OR selama proses pelatihan berlangsung dapat dilihat pada Gambar 3.4. pertidaksamaan-pertidaksamaan yang menghasilkan daerah dengan respons positif dan daerah dengan respons negatif pada setiap langkah dapat dilihat pada Tabel 3.4.

Tabel 3.4 Range respons positif dan respons negatif untuk fungsi logika OR dengan daerah input biner dan bipolar

Counter	Daerah respons positif $w_1x_1 + w_2x_2 + b > \theta$	Daerah respons negatif $w_1x_1 + w_2x_2 + b < -\theta$
1	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 + b < -0.1$
2	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 + b < -0.1$
3	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 + b < -0.1$
4	$x_1 + x_2 > 0.1$	$x_1 + x_2 < -0.1$
5	$x_1 + x_2 > 0.1$	$x_1 + x_2 < -0.1$
6	$x_1 + x_2 > 0.1$	$x_1 + x_2 < -0.1$
7	$x_1 + x_2 > 0.1$	$x_1 + x_2 < -0.1$
8	$x_1 + x_2 - 1 > 0.1$	$x_1 + x_2 - 1 < -0.1$
9	$x_1 + x_2 - 1 > 0.1$	$x_1 + x_2 - 1 < -0.1$
10	$2x_1 + x_2 > 0.1$	$2x_1 + x_2 < -0.1$
11	$2x_1 + x_2 > 0.1$	$2x_1 + x_2 < -0.1$
12	$2x_1 + x_2 - 1 > 0.1$	$2x_1 + x_2 -1 < -0.1$
13	$2x_1 + x_2 - 1 > 0.1$	$2x_1 + x_2 -1 < -0.1$
14	$2x_1 + x_2 - 1 > 0.1$	$2x_1 + x_2 -1 < -0.1$
15	$2x_1 + 2x_2 > 0.1$	$2x_1 + 2x_2 < -0.1$
16	$2x_1 + 2x_2 - 1 > 0.1$	$2x_1 + 2x_2 -1 < -0.1$
17	$2x_1 + 2x_2 - 1 > 0.1$	$2x_1 + 2x_2 -1 < -0.1$
18	$2x_1 + 2x_2 - 1 > 0.1$	$2x_1 + 2x_2 -1 < -0.1$
19	$2x_1 + 2x_2 - 1 > 0.1$	$2x_1 + 2x_2 -1 < -0.1$
20	$2x_1 + 2x_2 - 1 > 0.1$	$2x_1 + 2x_2 -1 < -0.1$



Gambar 3.4 Pergeseran batasan keputusan fungsi logika OR dengan input biner dan target bipolar

c) Input : bipolar
Target : bipolar

Tabel 3.5 Percobaan pengenalan fungsi logika OR dengan input dan target berbentuk bipolar

Co unt -er	INPUT (x_1 x_2 1)	NE T Y_n	OUTPU T Y	TARG ET T	BOBOT (Δw_1 Δw_2 Δ_b)	Perubahan bobot (Δw_1 Δw_2 $\Delta b_)$	keterangan
					(0 0 0)		inisialisasi
							Epoch ke-1
1	(1 1 1)	0	0	1	(1 1 1)	(1 1 1)	
2	(1 -1 1)	1	1	1	(1 1 1)	(0 0 0)	y=t bobot tetap
3	(-1 1 1)	1	1	1	(1 1 1)	(0 0 0)	y=t
4	(-1 -1 1)	-1	-1	-1	(1 1 1)	(0 0 0)	y=t
							Epoch ke-2
5	(1 1 1)	3	1	1	(1 1 1)	(0 0 0)	y=t
6	(1 -1 1)	1	1	1	(1 1 1)	(0 0 0)	y=t
7	(-1 1 1)	1	1	1	(1 1 1)	(0 0 0)	y=t
8	(-1 -1 1)	-1	-1	-1	(1 1 1)	(0 0 0)	sukses

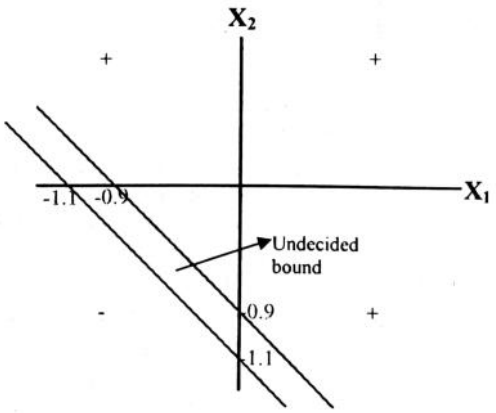
Hasil percobaan:
Unjuk kerja jaringan menunjukkan peningkatan bila data berada dalam format bipolar, baik input maupun targetnya.

Dari ketiga percobaan di atas dapat terlihat bahwa dibutuhkan 20 langkah (lima epoch) sampai jaringan mengenali fungsi logika OR bila menggunakan input biner dan target bipolar. Bandingkan dengan yang menggunakan input bipolar dan terget bipolar. Percobaan hanya membutuhkan 8 langkah (dua epoch) saja!

Pergeseran batasan keputusan fungsi logika OR selama proses pelatihan berlangsung dapat dilihat pada Gambar 3.5. pertidaksamaan-pertidaksamaan yang menghasilkan daerah dengan respons positif dan daerah respons negatif pada setiap langkah dapat dilihat pada Tabel 3.6 berikut ini.

Tabel 3.6 Range respons positif dan respons negatif untuk fungsi logika OR dengan input dan target bipolar

Counter	Daerah respons positif $w_1x_1 + w_2x_2 + b > \theta$	Daerah respons negatif $w_1x_1 + w_2x_2 + b < -\theta$
1	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$
2	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$
3	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$
4	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$
5	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$
6	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$
7	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$
8	$x_1 + x_2 + 1 > 0.1$	$x_1 + x_2 + 1 < -0.1$



Gambar 3.5 Batasan keputusan fungsi logika OR dengan input dan target bipolarGambar 3.15 memperlihatkan bahwa daerah output positiff dengan daerah output negatif sudah terpisahkan (bila diberi input -1 dan -1 outputnya adalah -1, input selain itu menghasilkan output +1).

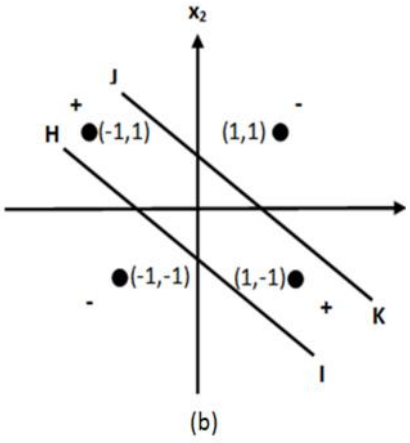
3.2 Perceptron Multilapis

Marvin Minsky dan Seymour dalam buku Diyah Puspita Ningrum (2006:44) memaparkan hasil penelitian mereka mengenai kelebihan-kelebihan dan keterbatasan-keterbatasan yang dimiliki oleh perceptron lapis tunggal. Keterbatasan yang paling utama adalah ketidakmampuan perceptron lapis tunggal dalam membedakan pola-pola secara linear tidak dapat dipisahkan. Hal ini sangat membatasi bidang aplikasi yang bisa diselesaikan oleh perceptron lapis tunggal. Hasil penelitian Minsky dan Papert di atas menyebabkan kegairahan akan jaringan syaraf tiruan menurun drastis.

Salah satu contoh kasus sederhana yang tidak bisa diselesaikan oleh perceptron lapis tunggal adalah masalah fungsi logika XOR. Gambar 3.6 mengilustrasikan masalah ini.

x_1	x_2	Output
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

(a)



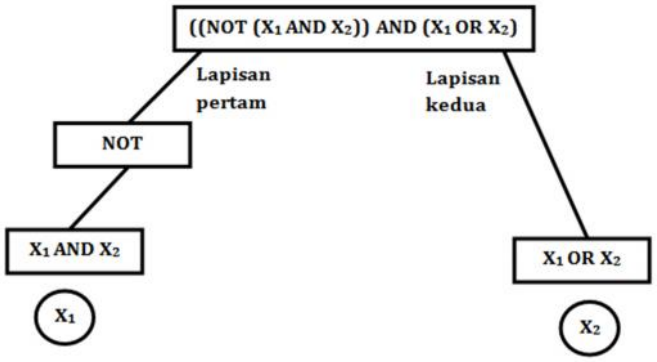
(b)

Gambar 3.6 masalah XOR. Tidak ada satu garis di manapun yang bisa memisahkan antara daerah respons positif dengan daerah respons negatif (a) Tabel fungsi logika XOR (dalam bentuk vipolar) (b) Dengan 2 buah garis pemisah (HI dan JK), bidang pada masalah XOR terbagi menjadi 3 bidang solusi

Pada Gambar 3.6(a) tabel fungsi ogika XOR pada dasarnya merupakan pemisah antara daerah dengan respons -1

dan +1. Gambar 3.6(b) adalah bidang x_1x_2 dengan empat titik input: (1,1), (1,0), (0,1), dan (0,0). Simbol + dan – masing-masing menggambarkan daerah respons +1 dan -1. Ternyata tidak ada sebuah garispun yang bisa memisahkan antara daerah + dengan daerah -. Untuk memisahkan kedua daerah tersebut dibutuhkan setidaknya 2 buah garis. Ketidakmampuan perceptron lapis tunggal dalam memecahkan masalah XOR dapat dibuktikan dengan melakukan modifikasi pada program Ormenu.cpp.

Penambahan lapisan tengah/lapisan tersembunyi memberikan fleksibilitas pada jaringan untuk memecahkan masalah XOR. Adanya lapisan tersembunyi mengijinkan adanya 2 garis yang bisa memisahkan sebuah bidang ke dalam 3 buah daerah. Diasumsikan bahwa perceptron lapis tunggal bisa mengenali fungsi-fungsi logika AND, OR, dan NOT. Fungsi XOR sendiri bisa dinyatakan dengan cara lain sebagai: (NOT (X_1 dan X_2) AND (X_1 OR X_2). Sebuah neuron perceptron lapis tunggal digunakan untuk melakukan NOT (X_1 AND X_2), sebuah lagi untuk melakukan X_1 OR X_2 dan sebuah lagi untuk melakukan operasi meng-AND-kan kedua operasi sebelumnya (lihat Gambar 3.17). jadi banyaknya lapisan neuron yang digunakan berjumlah lebih dari satu sehingga dinamakan perceptron multilapis.



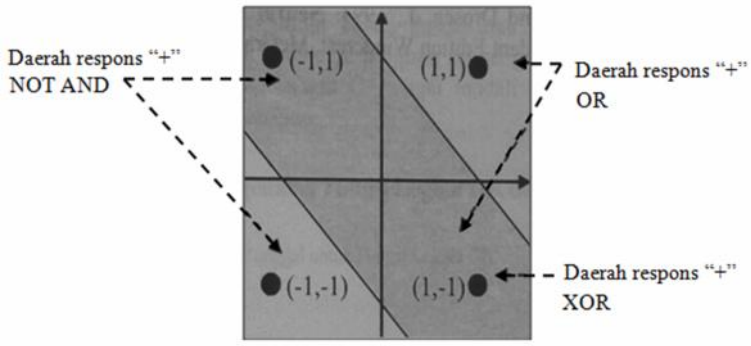
Gambar 3.7 Jaringan perceptron dua-lapis untuk masalah XOR

Perceptron multilapis adalah jaringan syaraf tiruan umpan maju (feedforward neural networks) dan merupakan jaringan

yang pembelajarannya terawasi sehingga ke dalam jaringan perlu dimasukkan contoh-contoh respons untuk dikenali. Seperti halnya teknik jaringan syaraf tiruan lainnya yang memiliki jenis pembelajaran terawasi, perceptron multilapis belajar mentransformasikan keluaran/respons seperti yang diingini. Teknik perceptron multiapis ini banyak digunakan untuk pengklasifikasian pola.

Kembali ke Gambar 3.6 (b), penyelesaian untuk masalah XOR adalah sebagai berikut: fungsi NOT (X_1 AND X_2) menghasilkan daerah respons positif pada daerah yang terletak di bawah garis JK. Sedangkan untuk fungsi X_1 OR X_2 daerah respons positifnya adalah yang terletak di atas garis HI. Fungsi XOR adalah (NOT(X_1 AND X_2)) AND (X_1 OR X_2), berarti mencakup irisan antara kedua daerah tersebut. Jadi daerah respons positif XOR terletak diasat garis HI dan dibawah garis JK. Daerah diluar daerah tersebut adalah daerah dengan respons negatif.

Perhatikan daerah-daerah yang bertumpukan untuk kasus XOR pada Gambar 3.8. Daerah-daerah yang bertumpukan semacam ini dapat sangat rumit dalam penentuan garis pemisahnya.



Gambar 3.8 Daerah XOR yang menumpuk di atas daerah NOT AND dan OR

C. Rangkuman

1. Perceptron lapis tunggal dapat dikatakan salah satu teknik jaringan syaraf tiruan yang sederhana. Teknik ini hanya mempunyai sebuah lapisan input dan sebuah unit output yang terdapat bias, yaitu unit yang aktivasinya selalu 1 dan berperilaku sebagai layaknya bobot (W).

2. Algoritma pelatihan perceptron adalah sebagai berikut:
   Langkah 0. Inisialisasi bobot-bobot dan bias.
   Tentukan angka pembelajaran α $(0<\alpha\leq 1)$
   Tentukan nilai ambang $\theta(0 < \theta\leq 1)$
   Langkah 1. Ulangi
   Langkah 2. Untuk setiap pasangan latihan **s:t**, lakukan
   Langkah 3. Tentukan aktivasi unit-unit input:

   $x_i = s_i$

   Langkah 4. Hitung respons dari unit output:

   $$y_{in} = b + \sum_n x_i$$

   $$y = \begin{cases} 1 \ jika & y_{in} > \ ^{-} & ....(3-1) \\ 0 \ jika & ^{-} \leq y_{in} \leq \ ^{-} & ....(3-2) \\ -1 \ jika & y_{in} < \ ^{-} & ....(3-3) \end{cases}$$

   Langkah 5. Updatelah bobot-bobot dan bias jika error terjadi pada pola y

   Jika $y \neq t$ maka

   $W_1(\text{baru}) = w_1(\text{lama}) + \alpha\, t\, x_i$,

   $B\ (\text{baru}) = b\ (\text{lama}) + \alpha\, t$

   Selain itu

   $W_1(\text{baru}) = w_1(\text{lama})$

   $b\ (\text{baru}) = b\ (\text{lama})$

   Langkah 6. Sampai kondisi berhenti terpenuhi
   Kondisi berhenti adalah kondisi di mana tidak terdapat bobot yang berubah pada langkah 2.
   Keterangan:

S	sensor
t	target
X_i	unit input ke-i
S_i	unit sensor ke-i
w_i	bobot ke-i
b	bias
y	unit respons (output)
α	angka pembelajaran
θ	nilai ambang
i	1,..., n di mana n adalah banyaknya unit input

3. Algoritma aplikasi perceptron:

   Langkah 0. Terapkan algoritma pelatihan untuk mengeset bobot-bobot.

   Langkah 1. Untuk setiap vektor input x yang ingin diklasifikasikan, lakukan langkah 3-3.

   Langkah 2. Set aktivasi unit-unit input (x_i), I = 1,...,n

   $x_i = s_i$

   langkah 3. Hitung respons unit output (y):

$$y_{in} = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & jika \quad y_{in} > \theta \\ 0 & jika \quad -\theta \le y_{in} \le \theta \\ -1 & jika \quad y_{in} < -\theta \end{cases}$$

4. Perceptron multilapis adalah jaringan syaraf tiruan umpan maju (feedforward neural networks) dan merupakan jaringan yang pembelajarannya terawasi sehingga ke dalam jaringan perlu dimasukkan contoh-contoh respons untuk dikenali.

D. Tugas

1. Buatlah kesimpulan mengenai perceptron pada bab 3 ini !
2. Carilah referensi lain mengenai perceptron yang telah dibahas pada bab 3 ini!

E. Tes Formatif

1. Jelaskan pengertian dari perceptron!
2. Apa yang dimaksud dengan perceptron multilapis?
3. Apa tujuan dari algoritma pelatihan dan bagaimana cara penerapannya?

LAMPIRAN BAB 3

A. Listing Program Bagian 3.1

Berikut adalah program sederhana yang mengilustrasikan konsep perceptron seperti dijelaskan pada Bab 3. Program ini ditulis dengan menggunakan bahasa pemrograman codevision. Meskipun demikian program ini juga bisa ditulis dalam Pascal atau BASIC atau bahasa pemrograman lainnya.

Ormenu.CPP: Pengenalan Fungsi Logika OR dengan Perceptron

```
//Ormenu.CPP
//Model Perceptron Lapis Tunggal untuk Fungsi Logika OR

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
#include <iostream.h>

struct dt_OR {
        int X1;
        int X2;
        int bias;
        int target;
} dt_OR {
        int X1;
        int X2;
        int bias;
        int target;
}dt_OR1;
char buffer[128];
float alpha, ambang;
struct dt_hsl {
        float j, s1, s2, t;
        float y, y_in, x1, x2, bias,
        w1, w2, b, dw1, dw2, db, sum_xw;
} dt_hsl1;
FILE *fp,*fp2;
/*Menampilkan menu*/
```

```
menu()
{
        clsrcr();
        gotoxy(20,3);printf("Pengenalan Fungsi Logika OR dengan
        Perceptron");gotoxy(12,8);printf("  MENU  ");
        gotoxy(12,9);
        printf("~~~~~~~~~~~~~~~~~~~~~~~~");
        gotoxy(13,10); printf("1.Membuat file data_OR.dat");
        gotoxy(13,11); printf("2.Menampilkan isi file data ke layar");
        gotoxy(13,12); printf("3.Training jaringan syaraf tiruan");
        gotoxy(13,13); printf("4.Menampilkan hasil training");
        gotoxy(13,14); printf("5.Aplikasi/testing jaringan");
        gotoxy(13,15); printf("5.Aplikasi/testing jaringan");
        gotoxy(12,16);
        printf("~~~~~~~~~~~~~~~~~~~~~~~~");
        gotoxy(20,18); printf("pilihan Anda:");
        return(0);
}
/*Membaca dan menulis data structure ke dalam file data*/
isi_data()
{
  char berhenti;
  int i;

  clsrcr()
}

/*Membaca dan menulis data structure ke dalam file data*/
isi data()
{
      char berhenti
      int i;

      clrscr();
      i = 1;
      berhenti = 't'

   /*Meminta masukan data */
   While (toupper(berhenti)!= 'Y' {
```

```
                printf("data ke-%d",i);i++
                printf("\n------");
                printf("\X1     :");
                gets(buffer);
                dt_OR1.X1 = atof(buffer);
                printf("X2      :")
                gets(buffer);
                dt_OR1.X2 = atof(buffer);
                dt_OR1.bias = 1;
                printf("Target :");
                gets(buffer);
                dt_OR1.target = atof(buffer);

    /*simpan ke file data(data_OR.dat)*/
    fwrite(&dt_OR1,sizeof(dt_OR1), 1, fp);

                printf("\nBerhenti (Y/T) ?");
                cin>>berhenti;
    printf("\n\n");
        }
        return ();
}
/* menampilkan isi file data ke layar */
baca_data()
{
        int i = 1;

        clrscr();
        /* mencetak judul */
        printf("    DATA OR \n");
        printf("---------------\n");
        printf(" No. X1 X2 Bias Target \n");
        printf("---------------\n");
        while (fread(&dt_OR!),1,fp)==1) {
                printf(" %2d %3d 52d %4d %6d\n", i++, dt_OR1.X1,
dt_OR1.X2, dt_bias, dt_OR1.target);
        }
        printf("---------------\n");
        getc();
```

```
        return ();
}

/* Memproses data berdasarkan algoritma pelatihan perceptron */
training()
{
        double temp1, temp2, temp3;
        int cek;

    /*inisialisasi */
    int i=1;
        alpha = 1;
        ambang = 0.1;

        dt_hsl1.j = 0;
        dt_hsl1.w1 = 0;
        dt_hsl1.w2 =0;
        dt_hsl1.b =0;
        dt_hsl1.dw1 = -999;
        dt_hsl1.dw2 = -999;
        dt_hsl1.db = -999;

/* lakukan selama kondisi berhenti tidak teroenuhi untuk semua
pasangan s:t */
        while(dt_hsl.dw1!=0 ||dt_hsl1.dw2!=0||dt_hsl1.db!=0||cek!=0) {
                fseek(fp,0,SEEK_SET);
                cek = 0;
                while(fread&dt_OR1,sizeof(dt_OR1),1,fp)==1) {
                        dt_hsl1.s1 = dt_OR1.X1;
                        dt_hsl1.x1 = dt_hsl1.s1;
                        dt_hsl1.s2 = dt_OR1.X2;
                        dt_hsl1.t = dt_OR1.target;
                        dt_hsl1.bias = dt_OR1.bias;

                        dt_hsl1.sum_xw = dt_hsl1.x1 *dt_hsl1.w1+dt_hsl1.x2*
                        dt_hsl.w2;
                        dt_hsl1.y_in = dt_hsl1.b + dt_hsl1.y = -1;
                        else
                                if(dt_hsl1.y_in < _ambang) dt_hsl1.y = -1;
```

```
                    else
                        dt_hsl1.y = 0;
                temp1 = dt_hsl1.w1;
                temp2 = dt_hsl1.w2;
                temp3 = dt_hsl1.b;

                if(dt_hsl.y !=dt_hsl1.t){
                    dt_hsl1.w1 = dt_hsl1.w1 + alpha *dt_hsl1.t*
                    dt_hsl1.x1;
                    dt_hsl1.w2 = dt_hsl1.w2 + alpha *dt_hsl1.t*
                    dt_hsl1.x2;
                    dt_hsl1.b = dt_hsl1.b + alpha *dt_hsl1.t;
                }

                dt_hsl1.dw1 = dt_hsl1.w1 - temp1;
                dt_hsl1.dw2 = dt_hsl1.w2 - temp2;
                dt_hsl1.db = dt_hsl1.b - temp3;
                dt_hsl1.j++;

                /* Menyimpan hasil pelatihan ke file hasil
(Hasil_OR.dat) */
                fwrite(&dt_hsl1,sizeof(dt_hsl1),1,fp2);

                if(dt_hsl1.dw1!=0||dt_hsl1.dw2!=0||dt_hsl1.db!=0)
                    cek = cek++;
              }
            }
            clscr();
            if(dt_hsl1.y!=dt_hsl1.t) {
                cout<<"Training mengalami stagnasi ...";
                cout<<"\nHasil aplikasi mungkin tidak sesuai dengan
yang diharapkan. ";

                }
            else cout<<"Training sukses ...";
            printf("\n\n\nTekan <ENTER> untuk kembali ke menu
utama ...");
            getch();
            return 0;
```

```
}

/* Menampilkan hasil ke layar dan menyimpan file ke file teks */
baca_hasil()
{
        int i=1;

        //isi file hasil (Hasil_OR.dat) ditampilkan ke layar
        //disamping itu disimpan pula ke file hasil (Training.txt)
        clrscr();
        printf("\n i x1 x2 1 y_in y t w1 w2 b dw1 dw2 db");
        printf("\n-----------------------------------");
        fprintf(fp2,"\n 1 x1 x2 1 y_in y t w1 w2 b dw1 dw2 db");
        fpintf(fp2,"\n-----------------------------------");

        while(fread(7dt_hsl1,sizeof(dt_hsl1),1,fp)==1) {
                printf("\n%2d %3.0f %3.0f %3.0f "
                ,i++,dt_hsl1.x1, dt_hsl1.x2, dt_hsl1.bias);
                printf("%3.2f %3.0f %3.0f ",
                dt_hsl1.y_in, dt_hsl1.t);
                printf(" %3.2f %3.2f %3.2f %3.2f %3.2f %3.2f",
                dt_hsl1.w1, dt_hsl.w2, dt_hsl1.b,
                dt_hsl1.dw1, dt_hsl1.dw2, dt_hsl1.db);

                fprintf(fp2,"\n%2d %3.0f %3.0f %3.0f "
                ,i, dt_hsl1.x1, dt_hsl1.x2, dt_hsl1.bias);
                fprintf(fp2,"%3.2f %3.0f %3.0f ",
                dt_hsl1.y_in, dt_hsl.y, dt_hsl1.t);
                fprintf(fp2," %3.2f %3.2f %3.2f %3.2f %3.2f %3.2f",
                dt_hsl1.w1,dt_hsl1.w2, dt_hsl1.b,
                dt_hsl1.dw1,dt_hsl1.dw2, dt_hsl1.db);
        }
        printf("\n-----------------------------------------\n");
        fprintf(fp2,"\n-----------------------------------------\n");
        getch();
        return 0;
}

/* Aplikasi perceptron */
```

```
aplikasi()
{
        int x1, x2;
        float SUM_XW, Y_IN, Y;

        clrscr();
        printf("peringatan :\n");
        printf("Harus sudah pernah mengambil menu 3.....\n\n\n");
        Printf("Unit input X1 : ");     scanf(%d",&x1);
        printf("Unit input X2 : ");     scanf(%d",&x2);
        printf("~~~~~~~~~~~~~");
        SUM_XW = x1 *dt_hsl1.w1 + x2 *dt_hsl1.w2;
        Y_IN = dt_hsl1.b + SUM_XW;

        if(Y_IN > ambang) Y = 1;
        else
            if(Y_IN < -ambang) Y =-1;
            else
                Y = 0;

        printf("\Output=%3.0f",Y);
        printf("\n\n\nTekan <ENTER> untuk kembali ke menu
utama ...");
        getch();
        return 0;
}
//~~~~~~~~~~~~~~~~~
//  PROGRAM UNTAMA
//~~~~~~~~~~~~~~~~~
main(int argc, char *argv[])
{
        char pilih = '0';
        argv[1]="D:\\Data_OR.dat";      //Nama file data (.dat)
        argv[2]="D:\\Hasil_OR.dat";     //Nama file hasil (.dat)
        argv[3]="D:\\Training.txt";     //Nama file hasil (.txt)

while (pilih !='6') {
        menu();
        pilih = getche();
```

```
switch(pilih)
case 'I':
        if ((fp=fopen(argv[1],"wb"))==NULL) {
                printf("Tidak dapat membuka file !!");
                exit(1);
        }
        isi_data();
        Fclose(fp);
        break;
case '2':
        if((fp=fopen(argv[1],"rb"))NULL) {
                printf("Tidak dapat membuka file !!");
                exit(1);
        }
        baca_data();
        fclose(fp);
        break;
case '3':
        if((fp=fopen(argv[1],"rb"))NULL) {
                printf("Tidak dapat membuka file !!");
                exit(1);
        }
        if ((fp2=fopen(argv[2],"wb"))==NULL) {
                 printf("Tidak dapat membuka file !!");
                 exit(1);
        }
        training();
        fclose(fp);
        fclose(fp2);
        break;
case '4':
        if ((fp2=fopen(argv[2],"wb"))==NULL) {
                printf("Tidak dapat membuka file !!");
                exit(1);
        }
        if ((fp2=fopen(argv[2],"wb"))==NULL) {
                printf("Tidak dapat membuka file !!");
                exit(1);
        baca_hasil();
```

```
                fclose(fp);
                break;
        case '5':
                if ((fp2=fopen(argv[2],"wb"))==NULL) {
                        printf("Tidak dapat membuka file !!");
                        exit(1);
                }
                aplikasi();
                fclose(fp);
                break;
        case '6':
                break;
        }
}
return(0);
```

B. Penjelasan Program bagian 3.1

Program ini melibatkan 3 buah file, yaitu Data_OR.dat, Hasil_OR.dat, dan Training.txt yang semuanya terletak di direktori D. Pembaca bisa mengganti nama direktori ini dengan nama lain.

Pembaca harus memilih 1 terlebih dahulu untuk mengisi file Data_OR.dat. sekali sudah pernah mengisi file Data_OR.dat, pembaca tidak perlu lagi memilih pilihan 1 setiap kali menjalankan program karena datanya sudah ada, kecuali jika pembaca ingin mengubah data tersebut. Misalnya inginmengubah dari bentuk pasangan biner –bipolar atau bipolar-biner ataupun ke bentuk bipolar-bipolar.

Bila pembaca menjalankan program Rrmenu.cpp, tampilan yang akan muncul adalah sebagai berikut:

Pengenalan Fungsi Logika OR dengan Perceptron

MENU

1. **Membuat file Data_OR.dat**
2. **Menampilkan isi file data ke layar**
3. **Training jaringan syaraf tiruan**
4. **Menampilkan hasil training**
5. **Aplikasi / testing jaringan**
6. **Selesai**

Pilihan Anda:

Berikut ini diberikan contoh pengaplikasian perceptron untuk kasus XOR dengan data biner-biner. Pilih 1 pada menu di atas untuk membuat file Data_OR.dat. Istilah sesuai tabel kebenaran fungsi logika OR.

Contoh:

Data ke-1	**Data ke-3**
X1 : 1	**X1 : 0**
X2 : 1	**X2 : 1**
Target : 1	**Target : 1**
Berhenti (Y/T) ?t	Berhenti (Y/T) ?y
Data ke-2	**Data ke-4**
X1 : 1	**X1 : 0**
X2 : 0	**X2 : 0**
Target : 1	**Target : 0**
Berhenti (Y/T) ?t	**Berhenti (Y/T) ?y**

Pilihan ke-2 berguna untuk menampilkan isi file data
Pilihan ke-3 berguna untuk melatih jaringan syaraf tiruan dengan algoritma pelatihan perceptron. Dari sini jaringan belajar mengenali pola yang diberikan kepadanya
Pada pilihan ke-4, bila terjadi stagnasi, jaringan gagal mengenali pola yang dilatihkan kepadanya, maka akan muncul peringatan.

Training mengalami stagnasi

Hasil aplikasi mungkin tidak sesuai dengan yang diharapkan.

Sebaliknya, bila pola berhasil dikenali (pelatihan berjalan sesuai dengan yang diharapkan), maka akan muncul tulisan:

Training sukses....

Pilihan ke-5 (Aplikasi/testing jaringan) berguna untuk menguji coba kemampuan jaringan syaraf tiruan, apakah benar jaringan telah mengenali pola yang diberikan kepadanya selama pelatihan. Demikian pula jika data yang dimasukkan pada pilihan 5 ini adalah data X1 dan X2 di luar pola-pola set pelatihan. Jaringan syaraf tiruan (perceptron) akan berusaha mengenalinya sesuai penafsirannya terhadap apa-apa yang pernah dipelajarinya. Kemampuan jaringan syaraf tiruan dalam mengenali pola-pola yang telah diajarkan kepadanya disebut *kemampuan memorisasi*, sedangkan kemampuan jaringan syaraf tiruan dalam mengenali pola-pola yang belum pernah dipelajarinya disebut *kemampuan generalisasi*.

Contoh masukan pada pilihan 5:

Unit input X1 : -1
Unit input X2 : -1
- - - - - - - - - - - - - - - - - -
Output = -1

Berbagai macam bentuk data (biner-biner, bipolar-bipolar, bipolar-biner, bipolar-bipolar) dapat pembaca coba sendiri dengan menggunakan program ini.

C. Penjelasan Program untuk bagian 3.2

Ketidak mampuan perceptron lapis tunggal dalam memecahkan masalah XOR dapat dibuktikan dengan sedikit memodifikasi program Ormenu.cpp dan menyimpannya dengan nama lain XORmenu.cpp. jangan lupa mengganti nama-nama filenya menjadi sebagai berikut:

```
argv[1]="D:\\Data_XOR.dat";        //Nama file data (.dat)
```

```
argv[2]="D:\\Hasil_XOR.dat";        //Nama file haisl (.dat)
argv[3]="D:\\TrainingXOR.txt";      //Nama file hasl (txt)
```

Pada contoh di atas file-file disimpan di direktori D.

Pilih pilihan 1 pada menu untuk membuat file Data_XOR.dat. Isikan seperti dibawah ini:

Data ke-1	**Data ke-3**
X1 : 0	**X1 : 1**
X2 : 0	**X2 : 0**
Target : 0	**Target : 1**
Berhenti (Y/T) ?t	**Berhenti (Y/T) ?y**
Data ke-2	**Data ke-4**
X1 : 0	**X1 : 1**
X2 : 1	**X2 : 1**
Target : 1	**Target : 0**
Berhenti (Y/T) ?t	**Berhenti (Y/T) ?y**

Pada saat mengambil pilihan 2 maka akan tampil data yang baru saja dibuat, yaitu sebagai berikut:

DATA XOR

No.	X1	X2	Bias	Target
1	0	0	1	0
2	0	1	1	1
3	1	0	1	1
4	1	1	1	0

Setelah memasukkan data, data tersebut perlu dilatih dengan jaringan syaraf tiruan. Pilihan ketiga digunakan untuk ini. Akan muncul tulisan yang menyatakan bahwa:

Training mengalami stagnasi ...

Yang dimaksud dengan stagnasi adalah suatu keadaan di mana pelatihan akan memberikan hasil yang sama terus-menerus

tanpa pernah sampai ke tujuan (jaringan syaraf tiruan tidak bisa mengenali kesemua pola). Untuk lebih jelasnya, ambil pilihan 4, yaitu hasil pelatihan/training.

I	X1	X2	1	y_in	y	t	w1	w2	b	dw1	dw2	db
1	0	0	1	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
2	0	1	1	0.00	0	1	0.00	1.00	0.00	0.00	1.00	0.00
3	1	0	1	0.00	0	1	1.00	1.00	0.00	1.00	0.00	0.00
4	1	1	1	2.00	1	0	1.00	1.00	0.00	0.00	0.00	0.00
5	0	0	1	0.00	0	0	1.00	1.00	0.00	0.00	0.00	0.00
6	0	1	1	1.00	1	1	1.00	1.00	0.00	0.00	0.00	0.00
7	1	0	1	1.00	1	1	1.00	1.00	0.00	0.00	0.00	0.00
8	1	1	1	2.00	1	0	1.00	1.00	0.00	0.00	0.00	0.00

Tiba pada bagian aplikasi/testing jaringan, ketika diuji dengan pasangan input (0 0), (0 1), (1 0), hasilnya sesuai target. Tetapi ketika diuji dengan (1 1), karena tujuan pelatihan jaringan syaraf tiruan tidak tercapai, maka hasil yang diperolehpun tidak seperti yang diharapkan.

Perhatikan hasil pengujian di bawah ini:

Peringatan :
Harus sudah pernah mengambil menu 3 .....

Unit Input X1 : 1
Unit Input X2 : 1

Output = 1

Perceptron lapis tunggal gagal mengenali fungsi XOR!

BAB 4

JARINGAN HOPFIELD DISKRIT

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

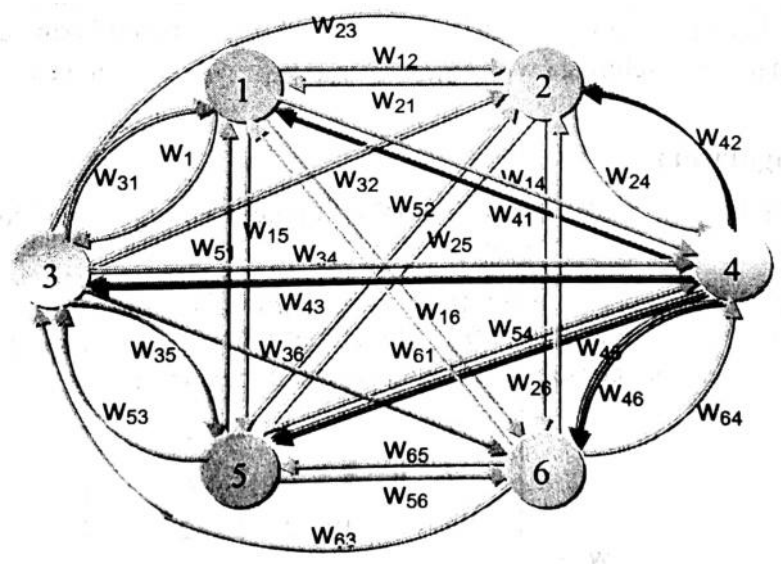
1. Menjelaskan mengenai arsitektur jaringan Hopfield Diskrit.
2. Menjelaskan mengenai algoritma jaringan Hopfield Diskrit.
3. Menjelaskan mengenai aplikasi pada jaringan Hopfield Diskrit.

B. Uraian Materi

Jaringan Hopfield diskrit merupakan jaringan syaraf tiruan yang terhubung penuh (*fully conneccted*), yaitu bahwa setiap unit terhubung dengan setiap unit lainnya. Jaringan ini memiliki bobot-bobot yang simetris. Pada jaringan Hopfield,setiap unit tidak memiliki hubungan dengan dirinya sendir. Seccara matematik hal ini memenuhi $w_{ij} = w_{ji}$ untuk $I \neq j$, dan $w_{ij} = 0$ untuk $I = j$.

Jaringan syaraf tiruan merupakan kumpulan dari neuron-neuron (sel-sel syaraf) di mana sebuah neuron berhubungan dengan sebuah neuron lainnya dengan cara mengirimkan informasi dalam bentuk fungsi aktivasi. Fungsi aktivasi yang digunakan dalam jaringan Hopfield adalah fungsi energi *Lyapunov*, yaitu sebuah fungsi yang terbatas dan menurun untk mmendapatkan kestabilan pada aktivasinya.

4.1 Arsitektur



Gambar 4.1 Jaringan Hopfield dengan 6 buah neuron

Gambar 4.1 menunjukkan sebuah jaringan Hopfield dengan 6 buah neuron yang terhubung satu sama lain. Setiap unit tidak

memiliki hubungan dengan dirinya sendiri. Hubungan antar-neuron tersebut memiliki bobot positif atau negatif. Berikut bobot-bobot tersebut digambarkan sebagai vektor W:

$$\begin{bmatrix} 0 & w_{12} & w_{13} & w_{14} & w_{15} & w_{16} \\ w_{21} & 0 & w_{23} & w_{24} & w_{25} & w_{26} \\ w_{31} & w_{32} & 0 & w_{34} & w_{35} & w_{36} \\ w_{41} & w_{42} & w_{43} & 0 & w_{45} & w_{46} \\ w_{51} & w_{52} & w_{53} & w_{54} & 0 & w_{56} \\ w_{61} & w_{62} & w_{63} & w_{64} & w_{65} & 0 \end{bmatrix}$$

Perhatikan bahwa bobot-bobot yang terletak pada diagonal utamanya adalah nol yang menunjukkan bahwa neuron-neuron pada jaringan Hopfield tidak memiliki hubungan dengan dirinya sendiri ($w_{ij} = 0$; i=j). sementara itu kesimetrisan vektor bobot berarti berlakunya $w_{ij} = w_{ji}$ di mana i≠j, sehingga $w_{12} = w_{12}$, $w_{13} = w_{31}$, $w_{23} = w_{32,...,}$ dan seterusnya.

4.2 Algoritma

Berikut ini adalah langkah-langkah menuju ke algoritma Hopfield diskrit. Misal terdapat 2 buah pola yang ingin dikenali, yaitu pola A (1,0,1,0,1,0) dan pola B (0,1,0,1,0,1). Bobot jaringan syaraf tiruan kemudian, sebagai contoh, ditentukan seperti dibawah ini :

$$\begin{bmatrix} 0 & -2 & 2 & -2 & 2 & -2 \\ -2 & 0 & -2 & 2 & -2 & 2 \\ 2 & -2 & 0 & -2 & 2 & -2 \\ -2 & 2 & -2 & 0 & -2 & 2 \\ 2 & -2 & 2 & -2 & 0 & -2 \\ -2 & 2 & -2 & 2 & -2 & 0 \end{bmatrix}$$

Bobot tersebut simetris ($w_{ij} = w_{ji}$; di mana I = baris dan j = kolom) dan diagonal utamanya adalah 0.

Pola A dan pola B diperlukan sebagai vektor. Dot product antara A dengan B diperoleh dengan cara mengalikan komponen kedua vektor tersebut dengan vektor bobot w sesuai kolom/node yang diingini dan kemudian menambahkannya seperti berikut ini.

Aktivasi node pertama untuk pola A:

$$\begin{pmatrix}1 & 0 & 1 & 0 & 1 & 0\end{pmatrix} \bullet \begin{bmatrix}0\\-2\\2\\-2\\2\\-2\end{bmatrix} = 0+0+2+0+2+0=4$$

Aktivasi node kedua:

$$\begin{pmatrix}1 & 0 & 1 & 0 & 1 & 0\end{pmatrix} \bullet \begin{bmatrix}-2\\0\\-2\\2\\-2\\2\end{bmatrix} = (-2)+0+(-2)+0+(-2)+0=-6$$

Dan seterusnya dengan cara yang sama ntuk node ketiga, keempat, kelima, dan keenam diperoleh masing-masing 4, -6, 4, -6.

Masih dengan cara yang sama untuk pola B, hasil aktivasi node pertama sampai keenampun diperoleh, masing-masing -6, 4, -6, 4, -6, 4. Kemudian fungsi ambang (*treshold function*) ditentukan agar jaringan syaraf tiruan bisa menghasilkan pola A dan pola B sebagai berikut:

$$f(t)=\begin{cases}1 & jika \quad t \geq \\ 0 & jika \quad t < \end{cases}$$

Di mana:

t = aktivasi node

θ = nilai ambang

Untuk contoh ini diambil sebagai θ adalah 0.

Bila dimasukkan 4 dan -6 ke dalam fungsi di atas maka akan diperoleh f(4) = 1 dan f(-6) = 0. Pola output untuk pola A dan pola B kemudian bisa ditentukan. Untuk pola A yang dihasilkan adalah (1,0,1,0,1,0). Ini sama seperti pola inputnya. Dikatakan bahwa jaringan syaraf tiruan sukses dalam mengenali atau berhasil memanggil kembali pola A yang dimasukkan ke dalam jaringan. Sedangkan untuk pola B, aktivasi (-6,4,-6,4,-6,4) menghasilkan pola output (0,1,0,1,0,1) yang berarti pola B sukses dipanggil kembali.

Sekarang pertimbangkan kedua pola berikut: C(1,0,1,0,0,0) dan D(0,0,0,1,0,1). Dengan melihat sepintas dapat diketahui bahwa semua pola C mirip dengan pola A dan pola D mirip dengan pola B. memang benar bahwa pola C dan pola D dianggap sebagai citra dari pola A dan pola B adalah sebagai berikut:

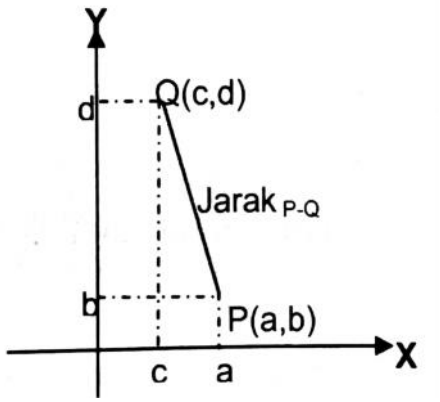
- Pengujian terhadap pola C menghasilkan aktivasi pada node pertama sampai keenam, masing-masing adalah (2,-4,2,-4,4,-4) dengan outputnya (1,0,1,0,1,0). Ini berarti ketika pola C dimasukkan ke dalam jaringan maka ia memanggil pola A sebagai inputnya.
- Hasil pengujian terhadap pola D sengaja ditinggalkan sebagai latihan bagi pembaca dengan hasil akhir menunjukkan ke pemanggilan kembali pola B.

Kesimpulan yang dapat diambil adalah bahwa jaringan Hopfield yang tengah dirancang ini akan menghasilkan output sesuai dengan kedekatan pola input terhadap pola target. Pada kasus di atas yang menjadi output adalah pola tersimpan yang lebih dekat dengan pola input.

Secara matematika kedekatan antara 2 buah titik berdimensi dua P(a,b) dan Q(c,d) dapat dicari sebagai berikut:

i. Lihat Gambar 3.2. jarak antara 2 buah titik berdimensi dua p(a,b) dan Q(c,d) dapat dicari sebagai berikut:

$$jarak_{p-q} = \sqrt{(a-c)^2 + (b-d)^2}$$



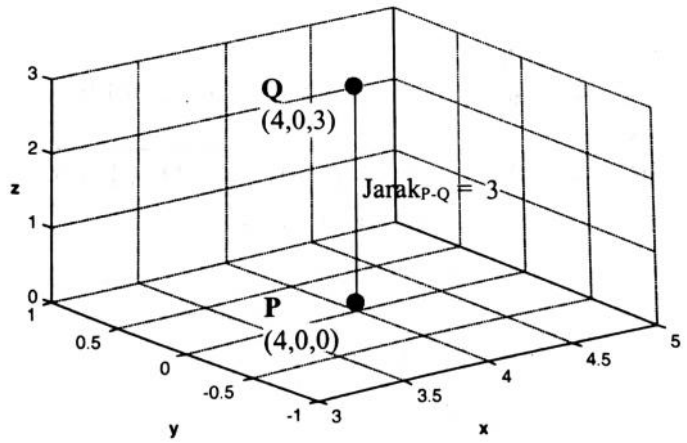
Gambar 4.2 jarak antara 2 buah titik pada bidang dimensi dua

ii. Dengan menggunakan pemikiran yang sama untuk 2 buah titik berdimensi tiga (lihat Gambar 4.3) diperoleh:

$$jarak_{p-q} = \sqrt{(a-d)^2 + (b-e)^2 + (c-f)^2}$$

dengan P(a,b,c) dan Q(d,e,f).

Jarak antara 2 Titik Berdimensi Tiga P(4,0,0) dan Q(4,0,3)



Gambar 4.3 Jarak antara 2 titik dimensi tiga

Demikian berlaku untuk titik-titik berdimensi empat, lima, dan seterusnya, sehingga jarak antara 2 buah titik berdimensi 6 adalah:

$$jarak_{p-q} = \sqrt{(a-g)^2 + (b-h)^2 + (c-i)^2 + (d-j)^2 + (e-k)^2 + (f-i)^2}$$

dengan P(a,b<c,d,e,f) dan Q(g,h,I,j,k,l).

pengujian kedekatan antara pola C(1,0,1,0,0,0) dan D(0,0,0,1,0,1) terhadap pola A(1,0,1,0,1,0) dan B(0,1,0,1,0,1) dilakukan dengan menghitung jarak pola C atau D terhadap pola A dan B.

- Perhitungan jarak antara C(1,0,1,0,0,0,0) dengan A(1,0,1,0,1,0):

$$jarak_{C-A} = \sqrt{(1-1)^2 + (0-0)^2 + (1+1)^2 + (0-0)^2 + (0-1)^2 + 0-0)^2}$$

$$= \sqrt{0+0+0+0+1+0}$$

$$= 1$$

- Perhitungan jarak antara C(1,0,1,0,0,0) dengan B(0,1,0,1,0,1):

$$jarak_{C-A} = \sqrt{(1-0)^2 + (0-1)^2 + (1-0)^2 + (0-1)^2 + (0-0)^2 + (0-1)^2}$$

$$= \sqrt{1+1+1+1+0+1}$$

$$= \sqrt{5}$$

Terlihat bahwa jarak C lebih dekat ke A dibandingkan dengan jarak C ke B. kesimpulan: Pola C lebih dekat dengan pola A.

Pengujian untuk pola D:

- Perhitungan jarak antara D(0,0,0,1,0,1) dengan A(1,0,1,0,1,0):

$$jarak_{D-A} = \sqrt{(0-0)^2 + (0-1)^2 + (0-0)^2 + (1-1)^2 + (0-0)^2 + (1-1)^2}$$

$$= \sqrt{1+0+1+1+1+1}$$

$$= \sqrt{5}$$

- Perhitungan jarak antara D(0,0,0,1,0,1) dengan B(0,1,0,1,0,1)

$$jarak_{D-B} = \sqrt{(0-0)^2 + (0-1)^2 + (0-0)^2 + (1-1)^2 + (0-0)^2 + (1-1)^2}$$

$$= \sqrt{0+1+0+0+0+0}$$

$$= \sqrt{1}$$

Terlihat bahwa jarak D lebih dekat ke B dibandingkan dengan ke A. Kesimpulan: pola D lebih dekat dengan pola B.

Telah disebutkan di atas bahwa jaringan Hpfield akan menghasilkan output sesuai dengan kecenderungan/kedekatan pola input terhadap sebuah pola target. Sekarang pertimbangkanlah suatu kasus dimana ternyata jarak sebuah pola input yang dimasukkan ke dalam jaringan ternyata sama dekatnya terhadap kedua pola target. Sebagai contoh adalah pola E(1,0,1,1,0,1) yang memiliki jarak yang sama terhadap pola A dan pola B.

Jarak antara E terhadap A dan B adalah sebagai berikut:

- Hitung jarak antara E(1,0,1,1,0,1) dengan A(1,0,1,0,1,0).

$$jarak_{E-A} = \sqrt{(1-1)^2 + (0-0)^2 + (1-1)^2 + (1-0)^2 + (0-1)^2 + (1-0)^2}$$

$$= \sqrt{0+0+0+1+1+1}$$

$$= \sqrt{3}$$

- Hitung jarak antara E(1,0,1,1,0,1) dengan B(0,1,0,1,0,1):

$$jarak_{E-B} = \sqrt{(1-1)^2 + (0-1)^2 + (1-0)^2 + (1+1)^2 + (0-0)^2 + (1-1)^2}$$

$$= \sqrt{1+1+1+0+0+0}$$

$$= \sqrt{3}$$

Terlihat bahwa jarak E terhadap A dan B adalah sama.

Dengan menggunakan langkah-langkah yang serupa dengan sebelumnya diperoleh aktivasi node pertama sampai keenam untuk vektor E adalah (-2,0,-2,-2,0,-2) dengan output (0,1,0,0,1,0). Vektor outputnya berupa sebuah pola yang sama sekali baru. Karena yang diinginkan adalah jaingan di mana output yang dihasilkan pola A atau pola B maka algoritma yang diterapkan sebelumnya perlu dimofdifikasi. Ide yang kemudian timbul adalah meng-update jaringan secara tidak sinkron (*ashynchronous update*), artinya peng-update-an tidak dilakukan secara bersamaan ke semua output yang diumpankan kembali sebagai input melainkan hanya dilakukan pada satu komponen vektor pada satu waktu. Algoritma tersebut dirumuskan seperti berikut ini.

Algoritma jaringan Hopfield diskrit:

0. Inisialisasi matriks bobot W,
1. Masukkan vektor input (invec), lalu inisialisasi vektor output (outvec) sebagai berikut:
2. Mulai dengan counter i=1
3. While Invec≠Outvec do langkah 4-7
   (jika I sudah mencapai nilai maksimum, I akan mereset ke 1 untuk melanjutkan siklus).
4. Hitung $nilai_i$ = DotProduct($Invec_i$, $Kolom_i$, dari W)
5. Hitung $Outvace_i$ = f($Nilai_i$) di man f adalah fungsi ambang (*treshold function*).
   Untuk pola input biner:

$$f(t) = \begin{cases} 1 & jika \quad t \geq \theta \\ 0 & jika \quad t < \theta \end{cases}$$

di mana θ biasanya sama dengan 0.
Untuk pola input bipolar:

$$f(t) = \begin{cases} 1 & jika \quad t > \theta \\ -1 & jika \quad t \leq \theta \end{cases}$$

di mana *treshold* θ biasanya sama dengan 0.

6. Update input jaringan dengan komponen Outvec;
7. i=i+1;

Catatan:
Jaringan Hopfield dikatakan sampai kepada nilai maksimum jika sebuah pola tertentu stabil dipanggil ulang. Batas iterasi biasanya cukup satu kali siklus setelah pola tertentu dipanggil secara stabil.

Penerapan algoritma jaringan Hopfield pada vektor input E yang memiliki jaraj yang sama terhadap pola-pola yang dikenali oleh jaringan memberikan hasi seperti pada Tabel 4.1 ternyata pola yang dipanggil kembali secara stabil oleh vektor E adalah pola B (0,1,0,1,0,1).

Tabel 4.1 Aplikasi algoritma jaringan Hopfield diskrit pada vektor E

langkah	i	Invec	Kolom vektor bobot	Nilai aktivasi	Outvec	Catatan
0		101101			101101	inisialisasi
1	1	101101	0 -2 2 -2 2 -2	-2	001101	Siklus pertama Kolom ke-1 outvec diubah sesuai output aktivasi
2	2	001101	-2 0 -2 2 -2 2	2	011101	Kolom ke-2 outvec diubah sesuai output aktivasinya
3	3	011101	2 -2 0 -2 2 -2	-6	010101	Kolom ke-3 outvec diubah sesuai output aktivasi
4	4	010101	-2 2 -2 0 -2 2	4	010101	Kolom ke-4 outvec tetap karena sesuai output aktivasi
5	5	010101	2 -2 2 -2 0 -2	- 6	010101	Kolom ke-6 outvec tetap karena sesuai ooutput aktivasi
6	6	010101	-2 2 -2 2 -2 0	4	010101	kolom ke-6 outvec tetap karena sesuai output aktivasi
7	1	010101	0 -2 2 -2 2 -2	-6	010101	Siklus ke-2 Kolom ke-1 outvec tetap
8	2	010101	-2 0 -2 2 -2 2	4	010101	Kolom ke-2 outvec tetap
9	3	010101	2 -2 0 -2 2 -2	-6	010101	Kolom ke-3 outvec

						tetap
10	4	010101	-2 2 -2 0 -2 2	4	010101	Kolom ke-4 outvec tetap
11	5	010101	2 -2 2 -2 0 -2	-6	010101	Kolom ke-5 outvec tetap
12	6	010101	-2 2 -2 2 -2 0	4	010101	Kolom ke-6 outvec tetap. Pola yang dipanggil ulang secara stabil = 010101

Mengenai penginilisasian matriks bobot:

(i) Set bobot-bobot pada diagonal utamanya dengan 0; ini mengingat ketentuan bahwa jaringan terhubung kecuali kepada dirinya sendiri (w_{ij} = 0 ; I =j, untuk I = baris dan j = kolom).

(ii) Set bobot-bobot selain yang terletak pada diagonal utama dengan bilangan sembarang sedemikian sehingga vektor output yang dihasilkan sama persis dengan pola input. Ini memerlukan pemilihan bobot secara hati-hati dengan memperhatikan kesimetrisan matriks bobot inisial. Pengertian dari simetris di sini adalah: antara matriks bobot dengan transpos dengan respons dari matriks bobot sama. Untuk menjamin hal tersebut w_{ij} harus sama dengan w_{ij} dimana I ≠ j (I = baris, j = kolom).

4.3 Aplikasi

Jaringan Hopfield diskrit dapat digunakan untuk menentukan apakah suatu vektor input "tidak dikenali" oleh jaringan. Disebut "dikenali" apabila output aktivasi yang dihasilkan jaringan sama dengan salah satu vektor yang disimpan oleh jaringan. Sebaliknya, jika vektor input "tidak dikenali" dan jaringan berkonvergensi menghasilkan sebuah vektor yang yang tidak merupakan salah satu pola yang disimpan dalam jaringan maka keadaan seperti ini disebut keadaan stabil palsu (*spurious stable state*).

Berikut ini diberikan salah satu aplikasi jaringan Hopfield diskrit (contoh 4.1) untuk mengenali pola "=" dan pola "x". contoh 4.2 menjelaskan mengenai *spurious stable state.*

Contoh 4.1 pengenalan pola "=" dan "x"

Gambar 4.4 Pola-pola yang ingin dikenali
(a) Pola "=" (b) pola "x"

Masing-masing pola pada Gambar 3.4 diterjemahkan sebagai data bipolar:

- Pola "sama dengan" = (1, 1, 1, -1, -1, -1, 1,1,1)
- Pola "kali" = (1, -1, 1, -1, 1, -1, 1, -1, 1)
  Sementara untuk matriks bobot inisial yang digunakan untuk mengenali kedua pola di atas ditentukan melalui serangkaian pemilihan yang hati-hati agar menghasilkan vektor output sama dengan vektor input. Matriks bobot inisial:

$$w = \begin{bmatrix} 0 & 0 & 3 & -3 & 0 & -3 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 3 & 0 & 0 & -3 & 0 & -3 & 3 & 0 & 3 \\ -3 & 0 & -3 & 0 & 0 & 3 & -3 & 0 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 \\ -3 & 0 & -3 & 3 & 0 & 0 & -3 & 0 & 3 \\ 3 & 0 & 3 & -3 & 0 & -3 & 0 & 0 & 3 \\ 0 & 3 & 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 3 & 0 & 3 & -3 & 0 & -3 & 3 & 0 & 0 \end{bmatrix}$$

Cara menghitung nilai aktivasi node pertama sampai node ke-9 adalah dengan cara menggunakan cara yang sama seperti pada vektor biner. Untuk pola input "sama dengan" diperoleh nilai aktivasi (3,3,3,-9,-6,-9,12,6,15) sedangkan untuk pola input "kali" diperoleh (9,-9,9,-9,6,-9,6,-6,9). Vektor output untuk pola "sama dengan" dan pola "kali" masing-masing adalah (1,1,1,-1,-1,-1,1,1,1)

dan (1,-1,1,-1,1,-1,1,-1,1). Ini berarti jaringan Hopfield diskrit sukses daam memanggil kembali pola "sama dengan" dan pola "kali".

Contoh 4.2 Keadaan Stabil Palsu (Spurious Stable State)

Untuk melihat contoh mengenai spurious stable state, coba masukkan vektor input berikut ke dalam jaingan pada contoh 4.1 di atas: (-1, -1, -1, 1, -1, 1, -1, -1, -1).

C. Rangkuman

1. Jaringan Hopffield diskrit merupakan jaringan syaraf tiruan yang terhubung penuh (*fully connected*), yaitu bahwa setiap unit terhubung dengan setiap unit lainnya. Jaringan ini memiliki bobot-bobot yang simetris. Pada jaringan Hopfield,setiap unit tidak memiliki hubungan dengan dirinya sendir. Seccara matematik hal ini memenuhi $w_{ij} = w_{ji}$ untuk $I \neq j$, dan $w_{ij} = 0$ untuk $I = j$.
2. Jaringan syaraf tiruan merupakan kumpulan dari neuron-neuron (sel-sel syaraf) di mana sebuah neuron berhubungan dengan sebuah neuron lainnya dengan cara mengirimkan informasi dalam bentuk fungsi aktivasi. Fungsi aktivasi yang digunakan dalam jaringan Hopfield adalah fungsi energi *Lyapunov*, yaitu sebuah fungsi yang terbatas dan menurun untk mmendapatkan kestabilan pada aktivasinya.
3. Algoritma jaringan Hopfield diskrit:
   0. Inisialisasi matriks bobot W,
   1. Masukkan vektor input (invec), lalu inisialisasi vektor output (outvec) sebagai berikut:
   2. Mulai dengan counter i=1
   3. While Invec≠Outvec do langkah 4-7
      (jika I sudah mencapai nilai maksimum, I akan mereset ke 1 untuk melanjutkan siklus).
   4. Hitung $nilai_i$ = DotProduct($Invec_i$, $Kolom_i$, dari W)

5. Hitung $Outvace_i$ = f($Nilai_i$) di man f adalah fungsi ambang (*treshold function*).
   Untuk pola input biner:
   $$f(t)=\begin{cases}1 & jika \quad t \geq \theta \\ 0 & jika \quad t < \theta\end{cases}$$
   di mana θ biasanya sama dengan 0.
   Untuk pola input bipolar:
   $$f(t)=\begin{cases}1 & jika \quad t > \theta \\ -1 & jika \quad t \leq \theta\end{cases}$$
   di mana θ biasanya sama dengan 0.
6. Update input jaringan dengan komponen Outvec;
7. i=i+1;

   Catatan:
   Jaringan Hopfield dikatakan sampai kepada nilai maksimum jika sebuah pola tertentu stabil dipanggil ulang. Batas iterasi biasanya cukup satu kali siklus setelah pola tertentu dipanggil secara stabil.

D. Tugas

Carilah refernsi lain dan buatlah kesimpulan mengenai jaringan hopfield diskrit yang telah dibahas pada bab 4 ini!

E. Tes Formatif

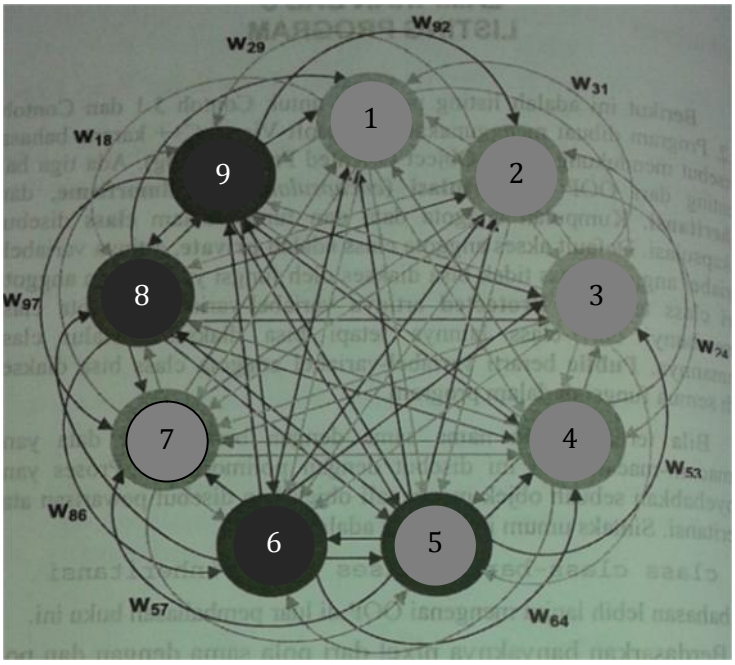
1. Apa yang anda ketahui mengenai jaringan Hopfield diskrit?
2. Sebutkan 3 contoh jaringan hopfield diskrit yang anda ketahui!
3. Buatlah gambar alur proses dari jaringan syaraf tiruan secara umum!

LAMPIRAN BAB 4
LISTING PROGRAM

Berikut ini adalah listing program untuk contoh 4.1 dan contoh 4.2. program dibuat menggunakan Microsoft Visual C++ karena bahasa tersebut mendukung OOP (Object Oriented Programming). Ada tiga hal penting dari OOP: enkapsulasi, polimorfisme, dan inheritasi. Kumpulan anggota data dan fungsi dalam class disebut enkapsulasi. Default akses anggota class adalah private, artinya variabel anggota class tidak bisa diakses oleh fungsi yang bukan anggota dari class tersebut. Protected artinya variabel-variabel anggota class tersembunyi dari class lainnya tetapi bisa sukses melalui class turunannya. Public berarti variabel-variabel anggota class bisa diakses oleh semua fungsi di dalam program.

Berdasarkan banyaknya pixel dari pola sama dengan dan pola kali (lihat gambar 4.4), dibutuhkan 9 buah neuron untuk mengenal kedua pola tersebut. Arsitektur jaringan hopfield diskrit untuk kasus ini digambarkan seperti pada gambar 4.5. semua unit terhubung penuh kecuali ke dirinya sendiri.

Berikutnya adalah menentukan matriks bobot inisial. Matriks bobot yang terpilih seperti pada contoh 4.1. sebenarnya dapat dipilih sembarang bilangan dengan 2 syarat: harus bilangan 0 pada diagonal utama, dan bilangan yang dipilih harus mampu memanggil kembali pola-pola yang ingin dikenali. Cara menentukan bilangan dengan *trial and error*.



Gambar 4.5 Arsitektur jaringan hopfield diskrit dengan 9 buah neuron

Pengecekan:

- Keluaran fungsi treshold dari doct product antara pola ke-1 dengan kolom ke-1 matriks bobot inisial harus sama dengan pixel pertama dari pola ke-1,
- Keluaran fungsi treshold dari doct product antara pola ke-1 dengan kolom ke-2 matriks bobot inisial harus sama dengan pixel kedua dari pola ke-1,
- Keluaran fungsi treshold dari doct product antara pola ke-1 dengan kolom ke-3 matriks bobot inisial harus sama dengan pixel ketiga dari pola ke-1, dan seterusnya,
- Keluaran fungsi treshold dari doct product antara pola ke-1 dengan kolom ke-9 matriks bobot inisial haris sama dengan pixel kesembilan dari pola ke-1.

Demikian pula pada pola ke-2:

- Keluaran fungsi treshold dari doct product antara pola ke-1 dengan kolom ke-1 matriks bobot inisial harus sama dengan pixel pertama dari pola ke-2,
- Keluaran fungsi treshold dari dot product antara pola ke-1 dengan kolom ke-2 matriks bobot inisial harus sama dengan pixel kedua dari pola ke-2,
- Keluaran fungsi treshold dari dot product antara pola ke-1 dengan kolom ke-3 matriks bobot inisial harus sama dengan pixel ketiga dari pola ke-2, dan seterusnya hingga
- Keluaran fungsi treshold dari dot product antara pola ke-1 dengan kolom ke-9 matriks bobot inisial harus sama dengan pixel kesembilan dari pola ke-2.

Perhatikan bahwa untuk pola input bipolar fungsi treshold yang berlaku:

$$f(t) = \begin{cases} 1 & jika \quad t > \\ -1 & jika \quad t \leq \end{cases}$$

Di mana treshold θ biasanya sama dengan 0.

Pembangunan Aplikasi Contoh 4.1

Langkah selanjutnya adalah membuat program sesuai algoritma jaringan hopfield diskrit. Pada program berikut ini class dideklarasikan dalam file header sedangkan definisi anggota fungsi dan fungsi main terletak di source code (APLIKASI_31.cpp).

Pertama deklarasi class pada file header APLIKASI_31.h

```
/* ************************************************************************
   program     : Apikasi_31.h
   program description :
         Headr file untuk program jaringan Hopfield diskrit
         dengan 9 buah neuron untuk mengenali pola "=" dan pola "x".
 ************************************************************************
*/
   #Include <stdio.h>
   #include <iostream.h>
   #include <match.h>

   class neuron
   {
      int aaktivasi;
      fried class jaringan;
   public:
      int bobotw[9];
      neuron() {};                  //konstruktor
      neuron(int *j) ;              //konstruktor
      int dotproduct (int,int*);
   };
   class jaringan
   {
   public:
      neuron nrn[9];
      int output[9];
      int threshold (int j[9]);
      void aktivasi (int j[9];
      jaringan (int*, int*, int*, int*, int*, int*, int*, int*, int*);
   //konstruktor
      };
```

Pada listing header file di atas terdapat 2 class yaitu class neuron dan class jaringan. Konstruktor adalah fungsi khusus yang bernama sama dengan class dan otomatis menginisialisasi objek class saat aplikasi dibangun. Class neuron memiliki 2 buah konstruktor, sebuah untuk menciptakan neuron sasaran tanpa menginisialisasi anggota data, yang

sebuah lagi untuk menciptakan neuron sasaran sekaligus menginisialisasi bobot-bobot koneksi antar neuron.
Source code :

```
/*
*****************************************************************
  Nama program : Aplikasi_31,cpp
  program description :
     program jaringan Hopfield diskrit dengan 9 buah
        neuron untuk mengenali pola "=" dan pola "x"
*****************************************************************
*/
  #include *Aplikasi_31.h"
  neuron::neuron(int *j)
  {
  int i;
  for(i=0;i<9;i++)
     {
     bobot[i]= *(j+i);
     }
  }

  int neuron::dotproduct(int m, int *x)
  {
  innt i;
  int a=0;

  for(i=0;i<m;i++)
     {
     a += x[i] *bobotw[i];
     }
  return a;
  }

  int jaringan:treshold(int k)
  {
  if(k>0)
     return (1);
```

```
    else
        return (-1);
    }

 jaringan::jaringan(int a[9], int b[9], int c[9], int d[9], int e[9],int f[9], int
g[9], int h[9], int h[9], int i[9])
    {
    nrn[0] = neuron(a) ;
    nrn[1] = neuron(b) ;
    nrn[2] = neuron(c) ;
    nrn[3] = neuron(d) ;
    nrn[4] = neuron(e) ;
    nrn[5] = neuron(f) ;
    nrn[6] = neuron(g) ;
    nrn[7] = neuron(h) ;
    nrn[9] = neuron(i) ;
    }

    void jaringan::aktivasi (int *pola)
    {

    int i,j,k;
    int langkahke;

    langkahke = 0;
  for (i=0;i<9;i++)
    {
    langkahke++;
    cout<<"\nLangkah ke-"<<langkahke;
    cout<<"\";
    for(k=0;k<9;k++)
            {
            cout<<"\nPola input = "<<pola[k];
            }
    cout<<*\n*;
    for(j=0;j<9;j++)
            {
            cout<<"\n nrn["<<i<<"].bobotw["<<j<<"} = "
                    <<nrn[1].bobotw[j];
```

```
        }
  nrn[i].aktivasi = nrn[i].dotproduct(9,pola);
  cout<<"\nAktivasi = ",,nrn[i].aktivasi;
  output[i]=threshold(nrn[i].aktivasi);
  cout<<"\nNilai output = "<<output[i]<<"\n";
  for(k=0;k<9;k++)
        {
        cout<<"\nPola output = "<<pola[k};
        }
  cout<<"\n-------------------------------------------";

  }
for (i=0;i<9;i++)
  {
  langkahke++;
  cout<<"\nLangkah ke-"<<langkahke;
  cout<<"\n";
  for(k=0;k<9;k++)
        {
        cout<<"\pola input = "<<pola[k];
        }
  cout<<"\n";
  for(j=0;j<9;j++)
        {
        cout<<"\n nrn["<<i<<"].bobotw["<<<j<<"] = "
              <<nrn[i].bobotw[j];
        }
  nrn[i].aktivasi = nrn[i]dotproduct(9,pola);
  cout<<"\nAktivasi = "<<nrn[i].aktivasi;
  output[1]=threshold{nrn[i].aktivasi);
  cout<<"\nNilai output = "<<output[i]<<"\n";
  pola[i] = output[i];
  for(k=0;k<9;k++)
        {
        cout<<"nPola output = "<<pola[k];
        }
  cout<<"\n---------------------------------------------------";
  }
  }
```

```
void jaringan::cek(int *pola)
{
    int i;
    cout<<"\n              HASIL AKHIR      \n*;
    for(i=0,i<9;i++)
    {
    if (output[i] == pola[i])
            cout<<"\n pola asli = "<<pola[i]<<
            " , output akhir = "<<output[i]<<"      komponen cocok";
    else
            if (output[i] == 1 && pola[i] == -1)
            cout<<"\n Pola asli = "<<pola[i]<<
            " Output akhir = "<<output[i]<<
            " noise pixel";
            else
            cout<<*\n Pola asli = "<<pola[i]<<
            " Output akhir = "<<output[i]<<
            " incomplete pixel";
    }
    cout<<"\n-----------------------------------------------"\n;
}

void main ()
{
    cout<<"\nPROGRAM      JARINGAN      HOPFIELD      DISKRIT
DENGAN 9 BUAH NEURON";
    cout<<"\nUNTUK MENGENALI POLA SAMA DENGAN [1,1,1,-
1,-1,-1,1,1,1,1] ";
    cout<<"\nDAN POLA SILANG [1,-1,1,-1,1,-1,1,-1,1]\n";

    // Inisialisasi
    int pola1[ ]= {1,1,1,-1,-1,-1,1,1,1,1};
    int pola2[ ]= {1,-1,1,-1,1,-1,1,-1,1};
    int polake1[ ]= {1,1,1,-1,-1,-1,1,1,1,1};
    int polake2[ ]= {1,-1,1,-1,1,-1,1,-1,1};

    // Matriks bobot inisial
    int bobot1[ ]= {0,0,3,-3,0,-3,3,0,3};
```

```
        int bobot2[ ]= {0,0,0,0,0,0,0,3,0};
        int bobot3[ ]= {3,0,0,-3,0,-3,3,0,3};
        int bobot4[ ]= {-3,0,-3,0,0,3,-3,0,-3};
        int bobot5[ ]= {0,0,0,0,0,0,0,-3,0};
        int bobot6[ ]= {-3,0,-3,3,0,0,-3,0,-3};
        int bobot7[ ]= {3,0,3,-3,0,-3,0,0,3};
        int bobot8[ ]= {0,3,0,0,-3,0,0,0,0};
        int bobot9[ ]= {3,0,3,-3,0,-3,3,0,0};

        //definisi vektor input ke-1
    jaringan
jar{bobot1,bobot2,bobot3,bobot4,bobot5,bobot6,bobot7,bobot8,bobot9};
        // mencari respons jaringan syaraf tiruan ketika pola ke-1
dimasukkan
            jar.aktivasi (pola);
        // mengecek apakah pola ke-1 bisa dipanggil ulang oleh jaringan
        // caranya dengan membandingkan antara keluaran jaringan dengan
            jar.cek(polake1);

            // pola ke-2
            jar.aktivasi (pola2);
            jar.cek(polake2);
        }
```

Output program C++ untuk jaringan Hopfield diskrit ini dapat disimpan ke dalam file teks. Misal source code dikompilasi dengan Microsoft visual C++ 6.0 dan file-file Aplikasi_31.cpp serta APLIKASI_31.h terletak di direktori C:\>Debug di mana file APLIKASI_31.exe berada lalu ketik APLIKa~1>HASIL31.TXT. Output program akan tersimpan di file HASIL31.TXT dan bisa dibuka dengan menggunakan Windows Explorer atau dengan mengetik type HASIL31.TXT.

Hasil31.TXT :
PROGRAM JARINGAN HOPFIELD DISKRIT DENGAN 9 BUAH NEURON
UNTUK MENGENALI POLA SAMA DENGAN [1,1,1,-1,-1,-1,1,1,1]
DAN POLA SILANG [1,-1,1,-1,1,-1,1,-1,1]

Langkah ke-1

pola input = 1
pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[0].bobotw[0] = 0
nrn[0].bobotw[1] = 0
nrn[0].bobotw[2] = 3
nrn[0].bobotw[3] = -3
nrn[0].bobotw[4] = 0
nrn[0].bobotw[5] = -3
nrn[0].bobotw[6] = 3
nrn[0].bobotw[7] = 0
nrn[0].bobotw[8] = 3
Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

---

langkah ke-2

pola input = 1
pola input = 1
pola input = 1
pola input = -1

pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[1].bobotw[0] = 0
nrn[1].bobotw[1] = 0
nrn[1].bobotw[2] = 0
nrn[1].bobotw[3] = 0
nrn[1].bobotw[4] = 0
nrn[1].bobotw[5] = 0
nrn[1].bobotw[6] = 0
nrn[1].bobotw[7] = 3
nrn[1].bobotw[8] = 0
Aktivasi = 3
Nilai output = 1

Langkah ke 3

pola input = 1
pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[2].bobotw[0] = 3
nrn[2].bobotw[1] = 0
nrn[2].bobotw[2] = 0
nrn[2].bobotw[3] = -3
nrn[2].bobotw[4] = 0
nrn[2].bobotw[5] = -3
nrn[2].bobotw[6] = 3
nrn[2].bobotw[7] = 0
nrn[2].bobotw[8] = 3

Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-4

pola input = 1
pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[3].bobotw[0] = -3
nrn[3].bobotw[1] = 0
nrn[3].bobotw[2] = -3
nrn[3].bobotw[3] = 0
nrn[3].bobotw[4] = 0
nrn[3].bobotw[5] = 3
nrn[3].bobotw[6] = -3
nrn[3].bobotw[7] = 0
nrn[3].bobotw[8] = -3
Aktivasi = -15
Nilai output = -1

Pola output = 1
Pola output = 1

Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-5

pola input = 1
pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[4].bobotw[0] = 0
nrn[4].bobotw[1] = 0
nrn[4].bobotw[2] = 0
nrn[4].bobotw[3] = 0
nrn[4].bobotw[4] = 0
nrn[4].bobotw[5] = 0
nrn[4].bobotw[6] = 0
nrn[4].bobotw[7] = -3
nrn[4].bobotw[8] = 0
Aktivasi = -3
Nilai output = -1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1

Pola output = 1
Pola output = 1

Langkah ke-6

pola input = 1
pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[5].bobotw[0] = -3
nrn[5].bobotw[1] = 0
nrn[5].bobotw[2] = -3
nrn[5].bobotw[3] = 3
nrn[5].bobotw[4] = 0
nrn[5].bobotw[5] = 0
nrn[5].bobotw[6] = -3
nrn[5].bobotw[7] = 0
nrn[5].bobotw[8] = -3
Aktivasi = -15
Nilai output = -1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-7

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[6].bobotw[0] = 3
nrn[6].bobotw[1] = 0
nrn[6].bobotw[2] = 3
nrn[6].bobotw[3] = -3
nrn[6].bobotw[4] = 0
nrn[6].bobotw[5] = -3
nrn[6].bobotw[6] = 0
nrn[6].bobotw[7] = 0
nrn[6].bobotw[8] = 3
Aktivasi = 15
Nilai input = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-8

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1

pola input = 1
pola input = 1

nrn[7].bobotw[0] = 0
nrn[7].bobotw[1] = 3
nrn[7].bobotw[2] = 0
nrn[7].bobotw[3] = 0
nrn[7].bobotw[4] = -3
nrn[7].bobotw[5] = 0
nrn[7].bobotw[6] = 0
nrn[7].bobotw[7] = 0
nrn[7].bobotw[8] = 0
Aktivasi = 6
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-9

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[8].bobotw[0] = 3
nrn[8].bobotw[1] = 0
nrn[8].bobotw[2] = 3

nrn[8].bobotw[3] = -3
nrn[8].bobotw[4] = 0
nrn[8].bobotw[5] = -3
nrn[8].bobotw[6] = 3
nrn[8].bobotw[7] = 0
nrn[8].bobotw[8] = 0
Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-10

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[0].bobotw[0] = 0
nrn[0].bobotw[1] = 0
nrn[0].bobotw[2] = 3
nrn[0].bobotw[3] = -3
nrn[0].bobotw[4] = 0
nrn[0].bobotw[5] = -3
nrn[0].bobotw[6] = 3
nrn[0].bobotw[7] = 0
nrn[0].bobotw[8] = 3

Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-11

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[1].bobotw[0] = 0
nrn[1].bobotw[1] = 0
nrn[1].bobotw[2] = 0
nrn[1].bobotw[3] = 0
nrn[1].bobotw[4] = 0
nrn[1].bobotw[5] = 0
nrn[1].bobotw[6] = 0
nrn[1].bobotw[7] = 3
nrn[1].bobotw[8] = 0
Aktivasi = 3
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-12

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[2].bobotw[0] = 3
nrn[2].bobotw[1] = 0
nrn[2].bobotw[2] = 0
nrn[2].bobotw[3] = -3
nrn[2].bobotw[4] = 0
nrn[2].bobotw[5] = -3
nrn[2].bobotw[6] = 3
nrn[2].bobotw[7] = 0
nrn[2].bobotw[8] = 3
Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-13
pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[3].bobotw[0] = -3
nrn[3].bobotw[1] = 0
nrn[3].bobotw[2] = -3
nrn[3].bobotw[3] = 0
nrn[3].bobotw[4] = 0
nrn[3].bobotw[5] = 3
nrn[3].bobotw[6] = -3
nrn[3].bobotw[7] = 0
nrn[3].bobotw[8] = -3
Aktivasi = -15
Nilai output = -1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-14

pola input = 1
pola input = 1
pola input = -1
pola input = -1

pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[4].bobotw[0] = 0
nrn[4].bobotw[1] = 0
nrn[4].bobotw[2] = 0
nrn[4].bobotw[3] = 0
nrn[4].bobotw[4] = 0
nrn[4].bobotw[5] = 0
nrn[4].bobotw[6] = 0
nrn[4].bobotw[7] = -3
nrn[4].bobotw[8] = 0
Aktivasi = -3
Nilai output = -1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-15

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[5].bobotw.[0] = -3
nrn[5].bobotw.[1] = 0
nrn[5].bobotw.[2] = -3
nrn[5].bobotw.[3] = 3
nrn[5].bobotw.[4] = 0
nrn[5].bobotw.[5] = 0
nrn[5].bobotw.[6] = -3
nrn[5].bobotw.[7] = 0
nrn[5].bobotw.[8] = -3
Aktivasi = -15
Nilai output = -1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-16

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[6].bobotw[0] = 3
nrn[6].bobotw[1] = 0
nrn[6].bobotw[2] = 3
nrn[6].bobotw[3] = -3
nrn[6].bobotw[4] = 0
nrn[6].bobotw[5] = -3

nrn[6].bobotw[6] = 0
nrn[6].bobotw[7] = 0
nrn[6].bobotw[8] = 3
Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-17

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[7].bobotw[0] = 0
nrn[7].bobotw[1] = 3
nrn[7].bobotw[2] = 0
nrn[7].bobotw[3] = 0
nrn[7].bobotw[4] = -3
nrn[7].bobotw[5] = 0
nrn[7].bobotw[6] = 0
nrn[7].bobotw[7] = 0
nrn[7].bobotw[8] = 0
Aktivasi = 6
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = 1
Pola output = 1

Langkah ke-18

pola input = 1
pola input = 1
pola input = -1
pola input = -1
pola input = -1
pola input = 1
pola input = 1
pola input = 1

nrn[8].bobotw[0] = 3
nrn[8].bobotw[1] = 0
nrn[8].bobotw[2] = 3
nrn[8].bobotw[3] = -3
nrn[8].bobotw[4] = 0
nrn[8].bobotw[5] = -3
nrn[8].bobotw[6] = 0
nrn[8].bobotw[7] = 0
nrn[8].bobotw[8] = 3
Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = 1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Pola output = 1
Pola output = 1
Pola output = 1

HASIL AKHHIR

Pola asli = 1, output akhir = 1 komponen cocok
Pola asli = 1, output akhir = 1 komponen cocok
Pola asli = 1, output akhir = 1 komponen cocok
Pola asli = -1, output akhir = -1 komponen cocok
Pola asli = -1, output akhir = -1 komponen cocok
Pola asli = -1, output akhir = -1 komponen cocok
Pola asli = 1, output akhir = 1 komponen cocok
Pola asli = 1, output akhir = 1 komponen cocok
Pola asli = 1, output akhir = 1 komponen cocok

Langkah ke-1

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[0].bobotw[0] = 0
nrn[0].bobotw[1] = 0
nrn[0].bobotw[2] = 3
nrn[0].bobotw[3] = -3
nrn[0].bobotw[4] = 0
nrn[0].bobotw[5] = -3
nrn[0].bobotw[6] = 3
nrn[0].bobotw[7] = 0
nrn[0].bobotw[8] = 3
Aktivasi = 15

Nilai output = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-2

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[1].bobotw[0] = 0
nrn[1].bobotw[1] = 0
nrn[1].bobotw[2] = 0
nrn[1].bobotw[3] = 0
nrn[1].bobotw[4] = 0
nrn[1].bobotw[5] = 0
nrn[1].bobotw[6] = 0
nrn[1].bobotw[7] = 3
nrn[1].bobotw[8] = 0
Aktivasi = -3
Nilai output = -1

Pola output = 1
Pola output = -1
Pola output = 1

Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-3

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[2].bobotw[0] = 3
nrn[2].bobotw[1] = 0
nrn[2].bobotw[2] = 0
nrn[2].bobotw[3] = -3
nrn[2].bobotw[4] = 0
nrn[2].bobotw[5] = -3
nrn[2].bobotw[6] = 3
nrn[2].bobotw[7] = 0
nrn[2].bobotw[8] = 3
Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Pola output = -1
Pola output = 1

Langkah ke-4

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[3].bobotw[0] = -3
nrn[3].bobotw[1] = 0
nrn[3].bobotw[2] = -3
nrn[3].bobotw[3] = 0
nrn[3].bobotw[4] = 0
nrn[3].bobotw[5] = 3
nrn[3].bobotw[6] = -3
nrn[3].bobotw[7] = 0
nrn[3].bobotw[8] = -3
Aktivasi = -15
Nilai output = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-5

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[4].bobotw[0] = 0
nrn[4].bobotw[1] = 0
nrn[4].bobotw[2] = 0
nrn[4].bobotw[3] = 0
nrn[4].bobotw[4] = 0
nrn[4].bobotw[5] = 0
nrn[4].bobotw[6] = 0
nrn[4].bobotw[7] = -3
nrn[4].bobotw[8] = 0
Aktivasi = 3
Nilai output = -1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-6

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1

pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[5].bobotw[0] = -3
nrn[5].bobotw[1] = 0
nrn[5].bobotw[2] = -3
nrn[5].bobotw[3] = 3
nrn[5].bobotw[4] = 0
nrn[5].bobotw[5] = 0
nrn[5].bobotw[6] = -3
nrn[5].bobotw[7] = 0
nrn[5].bobotw[8] = -3
Aktivasi = -15
Nilai output = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-7

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[6].bobotw[0] = 3
nrn[6].bobotw[1] = 0
nrn[6].bobotw[2] = 3
nrn[6].bobotw[3] = -3
nrn[6].bobotw[4] = 0
nrn[6].bobotw[5] = -3
nrn[6].bobotw[6] = 0
nrn[6].bobotw[7] = 0
nrn[6].bobotw[8] = 3
Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-8

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[7].bobotw[0] = 0
nrn[7].bobotw[1] = 0
nrn[7].bobotw[2] = 0
nrn[7].bobotw[3] = 0

nrn[7].bobotw[4] = -3
nrn[7].bobotw[5] = 0
nrn[7].bobotw[6] = 0
nrn[7].bobotw[7] = 0
nrn[7].bobotw[8] = 0
Aktivasi = -6
Nilai output = -1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-9

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[8].bobotw[0] = 0
nrn[8].bobotw[1] = 0
nrn[8].bobotw[2] = 3
nrn[8].bobotw[3] = -3
nrn[8].bobotw[4] = 0
nrn[8].bobotw[5] = -3
nrn[8].bobotw[6] = 3
nrn[8].bobotw[7] = 0
nrn[8].bobotw[8] = 0

Aktivasi = 15
Nilai output = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-10

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[0].bobotw[0] = 0
nrn[0].bobotw[1] = 0
nrn[0].bobotw[2] = 3
nrn[0].bobotw[3] = -3
nrn[0].bobotw[4] = 0
nrn[0].bobotw[5] = -3
nrn[0].bobotw[6] = 3
nrn[0].bobotw[7] = 0
nrn[0].bobotw[8] = 3
Aktivasi = 15
Nilai input = 1

Pola output = 1
Pola output = -1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-11

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[1].bobotw[0] = 0
nrn[1].bobotw[1] = 0
nrn[1].bobotw[2] = 0
nrn[1].bobotw[3] = 0
nrn[1].bobotw[4] = 0
nrn[1].bobotw[5] = 0
nrn[1].bobotw[6] = 0
nrn[1].bobotw[7] = 3
nrn[1].bobotw[8] = 0
Aktivasi = -3
Nilai input = -1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Pola output = -1
Pola output = 1

Langkah ke-12

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[2].bobotw[0] = 3
nrn[2].bobotw[1] = 0
nrn[2].bobotw[2] = 0
nrn[2].bobotw[3] = -3
nrn[2].bobotw[4] = 0
nrn[2].bobotw[5] = -3
nrn[2].bobotw[6] = 3
nrn[2].bobotw[7] = 0
nrn[2].bobotw[8] = 3
Aktivasi = 15
Nilai input = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-13

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[3].bobotw[0] = -3
nrn[3].bobotw[1] = 0
nrn[3].bobotw[2] = -3
nrn[3].bobotw[3] = 0
nrn[3].bobotw[4] = 0
nrn[3].bobotw[5] = 3
nrn[3].bobotw[6] = -3
nrn[3].bobotw[7] = 0
nrn[3].bobotw[8] = -3
Aktivasi = -15
Nilai input = -1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-14

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1

pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[4].bobotw[0] = 0
nrn[4].bobotw[1] = 0
nrn[4].bobotw[2] = 0
nrn[4].bobotw[3] = 0
nrn[4].bobotw[4] = 0
nrn[4].bobotw[5] = 0
nrn[4].bobotw[6] = 0
nrn[4].bobotw[7] = -3
nrn[4].bobotw[8] = 0
Aktivasi = 3
Nilai input = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-15

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[5].bobotw[0] = -3
nrn[5].bobotw[1] = 0
nrn[5].bobotw[2] = -3
nrn[5].bobotw[3] = 3
nrn[5].bobotw[4] = 0
nrn[5].bobotw[5] = 0
nrn[5].bobotw[6] = -3
nrn[5].bobotw[7] = 0
nrn[5].bobotw[8] = -3
Aktivasi = -15
Nilai input = -1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-16

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[6].bobotw[0] = 3
nrn[6].bobotw[1] = 0
nrn[6].bobotw[2] = 3
nrn[6].bobotw[3] = -3
nrn[6].bobotw[4] = 0

nrn[6].bobotw[5] = -3
nrn[6].bobotw[6] = 0
nrn[6].bobotw[7] = 0
nrn[6].bobotw[8] = 3
Aktivasi = 15
Nilai input = 1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-17

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[7].bobotw[0] = 0
nrn[7].bobotw[1] = 3
nrn[7].bobotw[2] = 0
nrn[7].bobotw[3] = 0
nrn[7].bobotw[4] = -3
nrn[7].bobotw[5] = 0
nrn[7].bobotw[6] = 0
nrn[7].bobotw[7] = 0
nrn[7].bobotw[8] = 0
Aktivasi = -6

Nilai input = -1

Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Langkah ke-18

Pola input = 1
Pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
pola input = 1
pola input = -1
Pola input = 1

nrn[8].bobotw[0] = 3
nrn[8].bobotw[1] = 0
nrn[8].bobotw[2] = 3
nrn[8].bobotw[3] = -3
nrn[8].bobotw[4] = 0
nrn[8].bobotw[5] = -3
nrn[8].bobotw[6] = 3
nrn[8].bobotw[7] = 0
nrn[8].bobotw[8] = 0
Aktivasi = 15
Nilai input = 1

Pola output = 1
Pola output = -1
Pola output = 1

```
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
```

HASIL AKHIR

```
Pola asli = 1 , output akhir = 1    komponen cocok
Pola asli = -1 , output akhir = -1    komponen cocok
Pola asli = 1 , output akhir = 1    komponen cocok
Pola asli = -1 , output akhir = -1    komponen cocok
Pola asli = 1 , output akhir = 1    komponen cocok
Pola asli = -1 , output akhir = -1    komponen cocok
Pola asli = 1 , output akhir = 1    komponen cocok
Pola asli = -1 , output akhir = -1    komponen cocok
Pola asli = 1 , output akhir = 1    komponen cocok
```

Kasus Spurious Stable State (Contoh 3.2)

Source code :

```
/*
**************************************************************************
   Nama program : Aplikasi_31.cpp
      Program description : contoh kasus spurious stable state
         Yan diinginkan adalah pola program yang bisa mengenali
         pola tersimpan "=" dan "x". pada kenyataannya output
         jaringan syaraf ternyata stabil memanggil pola lain.
**************************************************************************
*/
#include *APLIKASI_31.h"

neuron::neuron (int *j)
(
int i;
for(i=0;<9;i++)
   {   bobotw[i]= *(j+i);
```

```
    }
}

int neuron::dotproduct(int m, int *x)
{
int i;      int a=0;
for (i=0;i<mi++)
        {       a += x[i] *bobotw[i];
        }
return a;
}

int jaringan::treshold(int k)
{
if(k>0) return (1);
else    return (-1);
}

jaringan::jaringan (int a[9],int b[9], int c[9], int d[9], int e[9],int f[9],
int g[9], int h[9], int i[9]
{
nrn[0] = neuron(a) ;
nrn[1] = neuron(b) ;
nrn[2] = neuron(c) ;
nrn[3] = neuron(d) ;
nrn[4] = neuron(e) ;
nrn[5] = neuron(f) ;
nrn[6] = neuron(g) ;
nrn[7] = neuron(h) ;
nrn[8] = neuron(i) ;
}

void jaringan::aktivasi (int *pola)
{
int i,j,k;      int langkahke;
langkahke = 0;
for(i=0;i<9;i++)
        {
        langkahke++;
```

```
        cout<<"\langkah ke-"<<langkahke;
        cout<*\n*;
        for(k=0;k<9;k++)
                {
                cout<<"\nPola input = "<<pola[k];
                }
        cout<<*\n*;
        for(j=0;j<9;j++)
                {
                cout<<*\n nrn["<<i<<"}.bobotw["<<j<<] = "
                        <<nrn[i].bobotw[j];
                }
        nrn[i].aktivasi = nrn[i].dot[roduct(9,pola);
        cout<<"\nAktivasi = "<<nrn[i].aktivasi;
        output[i]=threshold(nrn[i].aktivasi);
        cout<<"\nNilai output = "<<output[i]<<*\n";
        pola [i] = output[i];
        for (k=0;k<9;k++)
                {       cout<<"\nPola output = "<<pola[k];
                }
        cout<<"\n-------------------------------------";

        }

for(i=0;i<9;i++)
        {
        langkahke++;
        cout<<"\nLangkah ke-*<<langkahke;
        cout<<"n";
        for(k=0;k<9;k++)
                {       cout<<\nPola input = "<<pola[k];
                }
        cout<<"\n";
        for(j=0;j<9;j++)
                {
                cout<<*\n nrn["<<i<<].bobotw["<<j<<] = "
                        <<nrn[i].bobotw[j];
                }
        nrn[i].aktivasi = nrn[i].dotproduct(9,pola);
```

```
        cout<<"\nAktivasi = "<<nrn[i].aktivasi;
        output[i]=threshold(nrn[i].aktivasi);
        cout<<"\nNilai output = "<<output[i]<<"\n";
        pola[i] = output[i];
        for(k=0;k<9;k++)
              {       cout<<"\nPola output = "<<pola[k];
              }
        cout<<"\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" ;

        }
}

void jaringan::cek(int *pola)
{
        int i;
        cout<<"\n                    HASIL AKHIR      \n*;
        for(i=0;i<9;i++)
        {
        if output [i] == pola[i])
              cout<<"\n Pola asli = "<<pola[i]<<
              " , output akhir = "<<output[i]<<"  komponen cocok";
        else
              if (output[i] == 1 && pola[i] == -1)
              cout<<"\n Pola asli = "<<pola[i]<<
              " output akhir = "<<output[i]<<
              " noisy pixel";
              else
              cout<<"\n Pola asli = "<<pola[i]<<
              " output akhir = "<<output[i]<<
              " incomplete pixel";
        }
        cout<<"\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
}

void main ()
{
        cout<<"\nCONTOH KASUS SPURIOUS STABLE STATE\n*;

        // inisialisasi
```

```
        int pola1[ ]= {1,1,1,-1,-1,-1,1,1,1};
        int pola2[ ]= {1,-1,1,-1,1,-1,1,-1,1};
        int pola3[ ]= {-1,-1,-1,1,-1,1,-1,-1,-1};
        int polake1[ ]= {1,1,1,-1,-1,-1,1,1,1};
        int polake2[ ]= {1,-1,1,-1,1,-1,1,-1,1};
        int polake3[ ]= {-1,-1,-1,1,-1,1,-1,-1,-1};

        // Matriks bobot inisial
        int bobot1[ ]= {0,0,3,-3,0,-3,3,0,3};
        int bobot2[ ]= {0,0,0,0,0,0,0,3,0};
        int bobot3[ ]= {3,0,0,-3,0,-3,3,0,3};
        int bobot4[ ]= {-3,0,-3,0,0,3,-3,0,-3};
        int bobot5[ ]= {0,0,0,0,0,0,0,-3,0};
        int bobot6[ ]= {-3,0,-3,3,0,0,-3,0,-3};
        int BOBOT7[ ]= {3,0,3,-3,0,-3,0,0,3};
        int bobot8[ ]= {0,3,0,0,-3,0,0,0,0};
        int bobot9[ ]= {3,0,3,-3,0,-3,3,0,0};

        //definifi vektor input ke-1
jaringan
jar (bobot1,bobot2,bobot3,bobot4,bobot5,bobot6,bobot7,bobot8,bobot9);
        // mencari respons jaringan syaraff tiruan ketika pola ke-1
dimasukkan
        jar.aktivasi (pola1);

        // mengecek apakah pola ke-1 bisa dipanggil ulang oleh jaringan
        // caranya dengan membandingkan antara keluaran jaringan dengan
vektor input asli dari pola ke-1
        // vektor input asli dari pola ke-1
          jar.cek(polake1);

        // pola ke-2
        jar.aktivasi(polake2);          jar.cek(polake2);

        //pola ke-3
        jar.aktivasi(polake3);          jar.cek(polake3);
}
```

Cuplikan output program untuk pola ke-3:

langkah ke-1

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[0].bobotw[0] = 0
nrn[0].bobotw[1] = 0
nrn[0].bobotw[2] = 3
nrn[0].bobotw[3] = -3
nrn[0].bobotw[4] = 0
nrn[0].bobotw[5] = -3
nrn[0].bobotw[6] = 3
nrn[0].bobotw[7] = 0
nrn[0].bobotw[8] = 3
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-2

Pola input = -1
Pola input = -1
Pola input = -1

Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[1].bobotw[0] = 0
nrn[1].bobotw[1] = 0
nrn[1].bobotw[2] = 0
nrn[1].bobotw[3] = 0
nrn[1].bobotw[4] = 0
nrn[1].bobotw[5] = 0
nrn[1].bobotw[6] = 0
nrn[1].bobotw[7] = 3
nrn[1].bobotw[8] = 0
Aktivasi = -3
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-3

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1

Pola input = -1

nrn[2].bobotw[0] = 3
nrn[2].bobotw[1] = 0
nrn[2].bobotw[2] = 0
nrn[2].bobotw[3] = -3
nrn[2].bobotw[4] = 0
nrn[2].bobotw[5] = -3
nrn[2].bobotw[6] = 3
nrn[2].bobotw[7] = 0
nrn[2].bobotw[8] = 3
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-4

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[3].bobotw[0] = 3
nrn[3].bobotw[1] = 0
nrn[3].bobotw[2] = 0

nrn[3].bobotw[3] = -3
nrn[3].bobotw[4] = 0
nrn[3].bobotw[5] = -3
nrn[3].bobotw[6] = 3
nrn[3].bobotw[7] = 0
nrn[3].bobotw[8] = 3
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-5

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[4].bobotw[0] = 0
nrn[4].bobotw[1] = 0
nrn[4].bobotw[2] = 0
nrn[4].bobotw[3] = 0
nrn[4].bobotw[4] = 0
nrn[4].bobotw[5] = 0
nrn[4].bobotw[6] = 0
nrn[4].bobotw[7] = -3

```
 nrn[4].bobotw[8] = 0
Aktivasi = 3
Nilai output = 1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
```

```
Langkah ke-6

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

 nrn[5].bobotw[0] = -3
 nrn[5].bobotw[1] = 0
 nrn[5].bobotw[2] = -3
 nrn[5].bobotw[3] = 3
 nrn[5].bobotw[4] = 0
 nrn[5].bobotw[5] = 0
 nrn[5].bobotw[6] = -3
 nrn[5].bobotw[7] = 0
 nrn[5].bobotw[8] = -3
Aktivasi = 15
Nilai output = 1

Pola output = -1
```

Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-7

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[6].bobotw[0] = 3
nrn[6].bobotw[1] = 0
nrn[6].bobotw[2] = 3
nrn[6].bobotw[3] = -3
nrn[6].bobotw[4] = 0
nrn[6].bobotw[5] = -3
nrn[6].bobotw[6] = 0
nrn[6].bobotw[7] = 0
nrn[6].bobotw[8] = 3
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1

Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-8

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[7].bobotw[0] = 0
nrn[7].bobotw[1] = 3
nrn[7].bobotw[2] = 0
nrn[7].bobotw[3] = 0
nrn[7].bobotw[4] = -3
nrn[7].bobotw[5] = 0
nrn[7].bobotw[6] = 0
nrn[7].bobotw[7] = 0
nrn[7].bobotw[8] = 0
Aktivasi = -6
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-9

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[8].bobotw[0] = 3
nrn[8].bobotw[1] = 0
nrn[8].bobotw[2] = 3
nrn[8].bobotw[3] = -3
nrn[8].bobotw[4] = 0
nrn[8].bobotw[5] = -3
nrn[8].bobotw[6] = 3
nrn[8].bobotw[7] = 0
nrn[8].bobotw[8] = 0
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-10

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1

Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[0].bobotw[0] = 0
nrn[0].bobotw[1] = 0
nrn[0].bobotw[2] = 3
nrn[0].bobotw[3] = -3
nrn[0].bobotw[4] = 0
nrn[0].bobotw[5] = -3
nrn[0].bobotw[6] = 3
nrn[0].bobotw[7] = 0
nrn[0].bobotw[8] = 0
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-11

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

```
 nrn[1].bobotw[0] = 0
 nrn[1].bobotw[1] = 0
 nrn[1].bobotw[2] = 0
 nrn[1].bobotw[3] = 0
 nrn[1].bobotw[4] = 0
 nrn[1].bobotw[5] = 0
 nrn[1].bobotw[6] = 0
 nrn[1].bobotw[7] = 3
 nrn[1].bobotw[8] = 0
Aktivasi = -3
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1
```

```
Langkah ke-12

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

 nrn[2].bobotw[0] = 3
 nrn[2].bobotw[1] = 0
 nrn[2].bobotw[2] = 0
 nrn[2].bobotw[3] = -3
```

nrn[2].bobotw[4] = 0
nrn[2].bobotw[5] = -3
nrn[2].bobotw[6] = 3
nrn[2].bobotw[7] = 0
nrn[2].bobotw[8] = 3
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-13

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[3].bobotw[0] = -3
nrn[3].bobotw[1] = 0
nrn[3].bobotw[2] = -3
nrn[3].bobotw[3] = 0
nrn[3].bobotw[4] = 0
nrn[3].bobotw[5] = 3
nrn[3].bobotw[6] = -3
nrn[3].bobotw[7] = 0
nrn[3].bobotw[8] = -3

Aktivasi = 15
Nilai output = 1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-14

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[4].bobotw[0] = 0
nrn[4].bobotw[1] = 0
nrn[4].bobotw[2] = 0
nrn[4].bobotw[3] = 0
nrn[4].bobotw[4] = 0
nrn[4].bobotw[5] = 0
nrn[4].bobotw[6] = 0
nrn[4].bobotw[7] = -3
nrn[4].bobotw[8] = 0
Aktivasi = 3
Nilai output = 1

Pola output = -1
Pola output = -1

Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-15

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[5].bobotw[0] = -3
nrn[5].bobotw[1] = 0
nrn[5].bobotw[2] = -3
nrn[5].bobotw[3] = 3
nrn[5].bobotw[4] = 0
nrn[5].bobotw[5] = 0
nrn[5].bobotw[6] = -3
nrn[5].bobotw[7] = 0
nrn[5].bobotw[8] = -3
Aktivasi = 15
Nilai output = 1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1

Pola output = -1
Pola output = -1

Langkah ke-16

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[6].bobotw[0] = 3
nrn[6].bobotw[1] = 0
nrn[6].bobotw[2] = 3
nrn[6].bobotw[3] = -3
nrn[6].bobotw[4] = 0
nrn[6].bobotw[5] = -3
nrn[6].bobotw[6] = 0
nrn[6].bobotw[7] = 0
nrn[6].bobotw[8] = 3
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

Langkah ke-17

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[7].bobotw[0] = 0
nrn[7].bobotw[1] = 3
nrn[7].bobotw[2] = 0
nrn[7].bobotw[3] = 0
nrn[7].bobotw[4] = -3
nrn[7].bobotw[5] = 0
nrn[7].bobotw[6] = 0
nrn[7].bobotw[7] = 0
nrn[7].bobotw[8] = 0
Aktivasi = -6
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

~~~~~~

Langkah ke-18

Pola input = -1
Pola input = -1
Pola input = -1
Pola input = 1
Pola input = -1
~~~~~~

Pola input = 1
Pola input = -1
Pola input = -1
Pola input = -1

nrn[8].bobotw[0] = 3
nrn[8].bobotw[1] = 0
nrn[8].bobotw[2] = 3
nrn[8].bobotw[3] = -3
nrn[8].bobotw[4] = 0
nrn[8].bobotw[5] = -3
nrn[8].bobotw[6] = 3
nrn[8].bobotw[7] = 0
nrn[8].bobotw[8] = 0
Aktivasi = -15
Nilai output = -1

Pola output = -1
Pola output = -1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = 1
Pola output = -1
Pola output = -1
Pola output = -1

HASIL AKHIR

Pola asli = -1 , output akhir = -1 komponen cocok
Pola asli = -1 , output akhir = -1 komponen cocok
Pola asli = -1 , output akhir = -1 komponen cocok
Pola asli = 1 , output akhir = 1 komponen cocok
Pola asli = -1 , output akhir = 1 noisy pixel
Pola asli = 1 , output akhir = 1 komponen cocok
Pola asli = -1 , output akhir = -1 komponen cocok
Pola asli = -1 , output akhir = -1 komponen cocok
Pola asli = -1 , output akhir = -1 komponen cocok

Keluaran program memperlihatkan pola yang dipanggil kembali bukan pola "=" (1,1,1,-1,-1,-1,1,1,1) atau pola "x" (1,-1,1,-1,1,-1,1,-1,1), atau pola masukannya (-1,-1,-1,1,-1,1,-1,-1,-1), melainkan sebuah pola lain (-1,-1,-1,1,1,1,-1,-1,-1) yang tidak dikenali oleh jaringan. Kondisi seperti ini disebut keadaan stabil palsu.

BAB 5

METODE PROPAGASI BALIK

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan mengenai algoritma propagasi balik.
2. Memahami arsitektur dan algoritma propagasi balik.
3. Menjelaskan mengenai pilihan-pilihan dalam pengaplikasian metode propagasi balik.

B. Uraian Materi

Metode propagasi balik merupakan metode yang sangat baik dalam menangani masalah pengenalan pola-pola kompleks. Metode ini merupakan metode jaringan syaraf tiruan yang populer. Beberapa contoh aplikasi yang melibatkan metode ini adalah pengompresian data, pendeteksian virus komputer, pengidentifikasian objek, sintesis suara dari teks, dan lain-lain. Beragam aplikasi jaringan syaraf tiruan dapat dilihat di Bab 8.

Istilah "propagasi balik" (atau "penyiaran kembali") diambil dari cara kerja jaringan ini, yaitu bahwa gradien error unit tersembunyi diturunkan dari penyiaran kembali error-error yang diasosiasikan dengan unit-unit output. Hal ini karena nilai target untuk unit-unit tersembunyi tidak diberikan.

Metode ini menurunkan gradien untuk meminimkan penjumlahan error kuadrat output jaringan. Nama lain dari propagasi balik adalah aturan delta yang digeneralisasi (*generalized delta rule*).

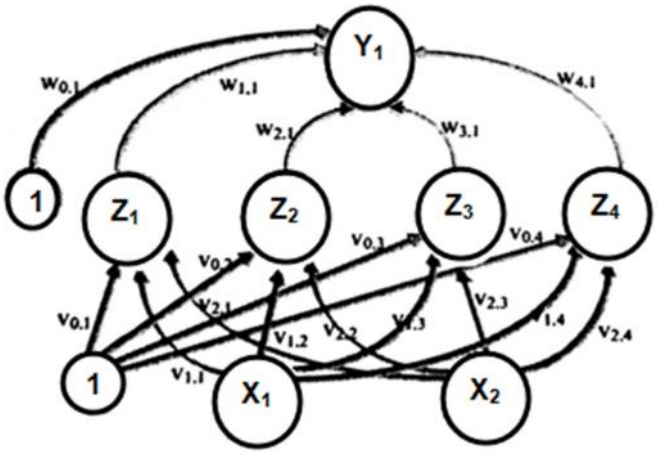
5.1 Arsitektur

Di dalam jaringan propagasi balik, setiap unit yang berada dilapisan input terhubung dengan setiap unit yang ada di lapisan tersembunyi. Hal serupa berlaku pula pada lapisan tersembunyi. Setiap unit yang ada di lapisan tersembunyi terhubung dengan dengan setiap unit yang ada di lapisan output. Lebih jelas tentang hal ini ditunjukkan pada Gambar 5.1.

Jaringan syaraf tiruan propagasi balik terdiri dari banyak lapisan (*multilayer neural networks*):

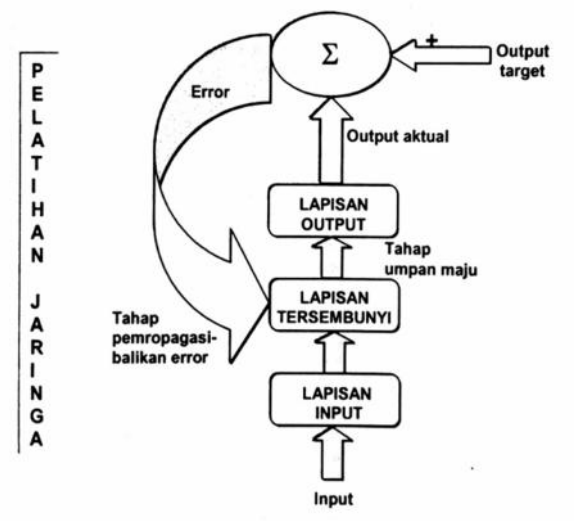
1. Lapisan input (1 buah). Lapisan input terdiri dari neuron-neuron atau unit-unit input, mulai dari unit input sampai unit input n.
2. Lapisan tersembunyi (minimal 1). Lapisan tersembunyi terdiri dari unit-unit tersembunyi mulai dari unit tersembunyi 1 sampai unit tersembunyi p.

3. Lapisan output (1 buah). Lapisan output terdiri dari unit-unit output mulai dari unit output 1 sampai unit output m, n, p, m masing-masing adalah bilangan intgrer sembarang menurut arsitektur jaringan syaraf tiruan yang dirancang. V_{0j} dan W_{0k} masing-masing adalah bias untuk unit tersembunyi ke-j dan untuk unit ke-k bias V_{0j} dan W_{0k} berperilaku seperti bobot di mana output bias ini selalu sama dengan 1. V_{ij} adalah bobot koneksi antara unit ke-I lapisan input dengan unit ke-j lapisan tersembunyi dengan unit ke-j lapisan output.



Gambar 5.1 Jaringan propagasi balik dengan satu buah lapisan tersembunyi. Perbedaan antara output aktual dengan output target dikembalikan lagi ke lapisan tersembunyi sebagai input untuk meng-update bobot, membawa keluaran jaringan ke arah semakin mendekati output terget.

5.2 Algoritma



Gambar 5.2 Alur kerja jaringan propagasi balik

Keterangan berikut ini mengacu pada Gambar 5.2. Agar dapat digunakan untuk suatu aplikasi, jaringan syaraf tiruan perlu belajar terlebih dahulu. Caranya, pada jaringan dimasukkan sekumpulan contoh pelatihan yang disebut set pelatihan. Set pelatihan ini digambarkan dengan sebuah vektor *feature* yang disebut vektor input yang diasosiasikan dengan sebuah output yang menjadi target pelatihannya. Pelatihan kemudian dilangsungkan dengan maksud membuat jaringan syaraf tiruan beradaptasi terhadap karakteristik-karakteristik dari contoh-contoh pada set pelatihan dengan cara melakukan pengubahan/peng-update-an bobot-bobot yang ada dalam jaringan.

Cara kerja jaringan propagasi balik adalah sebagai berikut: mula-mula jaringan diinisialisasi dengan bobot yang diset dengan bilangan acak. Lalu contoh-contoh pelatihan dimasukkan ke dalam jaringan. Keluaran dari jaringan berupa sebuah vektor output aktual. Selanjutnya vektor output aktual jaringan dibandingkan dengan vektor output target untuk mengetahui apakah output jaringan sudah sesuai dengan harapan (output aktual sudah sama dengan output target).

Error yang timbul akibat perbedaan antara output aktual dengan output target tersebut kemudian dihitung dan digunakan untuk meng-update bobot yang relevan dengan jalan mempropagasikan kembali error. Setiap perubahan bobot yang terjadi diharapkan dapat mengurangi besar error. Epoch (siklus setiap pola pelatihan) seperti ini dilakukan pada semua set pelatihan sampai unjuk kerja jaringan mencapai tingkat yang diinginkan atau sampai kondisi berhenti terpenuhi. Yang dimaksud dengan kondisi berhenti di sini misalnya: pelatihan akan dihentikan setelah epoch mencapai 1000 kali, atau pelatihan akan dihentikan sampai sebuah nilai ambang yang ditetapkan terlampaui. Setelah proses pelatihan selesai, barulah diterapkan algoritma aplikasi. Biasanya sebelum digunakan untuk aplikasi yang sebenarnya, pengujian unjuk kerja jaringan dilakukan dengan cara memasukkan set pengujian (set tes) ke dalam jaringan. Karena bersifat untuk menguji, set pengujian hanya berupa input saja. Dari respons jaringan dapat dinilai kemampuan memorisasi dan generalisasi jaringan dalam menebak output berdasarkan pada apa yang telah dipelajarinya selama ini.

Algoritma propagasi balik dapat dibagi ke dalam 2 bagian:

1. **Algoritma pelatihan**
   Terdiri dari 3 tahap: tahap umpan maju pola pelatihan input, tahap pemropagasibalikan error, dan tahap pengaturan bobot.
2. **Algoritma aplikasi**
   Yang digunakan hanyalah tahap umpan maju saja.

Berikut ini adalah algoritma pelatihan untuk jaringan propagasi balik dengan sebuah lapisan tersembunyi yang disusun menurut Gambar 5.1 dan 5.2.

Algoritma pelatihan

0. Inisialisasi bobot-bobot
   Tentukan angka pembelajaran (α)
   Tentukan pula nilai toleransi error atau nilai ambang (bila menggunakan nilai ambang sebagai kondisi berhenti); set maksimal epoch (bila menggunakan banyaknya epoch sebagai kondisi berhenti).

1. **While** kondisi berhenti tidak terpenuhi **do** langkah ke-2 sampai langkah ke-9.
2. Untuk setiap pasangan pola pelatihan, lakukan langkah ke-3 sampai langkah ke-8.

Tahap Umpan Maju

3. Setiap unit input x_i (dari unit ke-1 sampai unit ke-n pada lapisan input) mengirimkan sinyal input ke semua unit yang ada di lapisan atasnya (ke lapisan tersembunyi):

$$x_i$$

4. Pada setiap unit di lapisan tersembunyi z_j (dari unit ke-1 sampai unit ke-p; i=1,...,n; j=1,...,p) sinyal output lapisan tersembunyinya dihitung dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal input berbobot x_i :

$$z_j = f(v_0 + \sum x_i v_{ij} \Big)_{i=1}^{n}$$

Kemudian dikirim ke semua unit di lapisan atasnya.

5. Setiap unit di lapisan output y_k (dari unit ke-1 sampai unit ke-m; i=1,...,n; k=1,...,m) dihitung sinyal outputnya dengan meneraapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal input berbobot z_j bagi lapisan ini:

$$y_k = f(w_{0k} + \sum z_j w_{jk} \Big)_{i=1}^{p}$$

Tahap Pemropagasibalikan Error

6. Setiap unit output y_k (dari unit ke-1 sampai unit ke-m; j=1,...,p; k=1,...,m) menerima pola target t_k lalu informasi kesalahan lapisan output (δ_k) dihitung. δ_k dikirim ke lapisan di bawahnya dan digunakan untuk menghitung besar koreksi dan bias (Δw_{jk} dan Δw_{0k}) antara lapisan tersembunyi dengan lapisan output:

$$¢ = (t_k - y_k) \quad f(w_{0k} + \sum z_j w_{jk} \Big)_{j=i}^{p}$$

$$\Delta v_{jk} = \sim ¢_k z_j$$

$$\Delta v_{0k} = \sim ¢_k$$

7. pada setiap unit di lapisan tersembunyi (dari unit ke-1 sampai unit ke-p; i=1,...,n; j=1,...,p; k=1,...,m) dilakukan perhitungan informasi kesalahan lapisan tersembunyi (δ_j). δ_j kemudian digunakan untuk menghitung besar koreksi bobot dan bias (Δv_{ij} dan Δ_{0j}) antara lapisan input dan lapisan tersembunyi.

$$¢_j = (\sum ¢_k w_{jk})_{k=1}^{m} \; f(v_{0k} + \sum x_i v_{ij})_{i=1}^{n}$$

$$\Delta v_{ij} = \sim ¢_j x_i$$

$$\Delta v_{0j} = \sim ¢_j$$

Tahap Peng-pdate-an Bobot dan Bias

8. pada tahap unit output y_k (dari unit ke-1 sampai unit ke-m) dilakukan peng-update-an bias dan bobot (j=0,...,p; k=1,...,m) sehingga bias dan bobot yang baru menjadi:

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

dari unit ke-1 sampai unit ke-p di lapisan tersembunyi juga dilakukan peng-update-an pada bias dan bobotnya (i=0,...,n;j=1,...,p):

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

9. Tes kondisi berhenti.

Algoritma Aplikasi

0. Inisialisasi bobot. Bobot ini diambil dari bobot-bobot terakhir yang diperoleh dari algoritma pelatihan.
1. Untuk setiap vektor input, lakukan langkah ke-2 sampai ke-4.
2. Setiap unit x_i (dari unit ke-1 sampai unit ke-n pada lapisan input; i=1,..., n) menerima sinyal x_i ke semua unit pada lapisan di atasnya (unit-unit tersembunyi):
3. Setiap unit di lapisan tersembunyi z_j (dari unit ke-1 sampai unit ke-p; i=1,...,n; j=1,...,p) menghitung sinyal outputnya dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal input x_i. Sinyal output dari lapisan tersembunyi kemudian dikirim ke semua unit pada lapisan di atasnya:

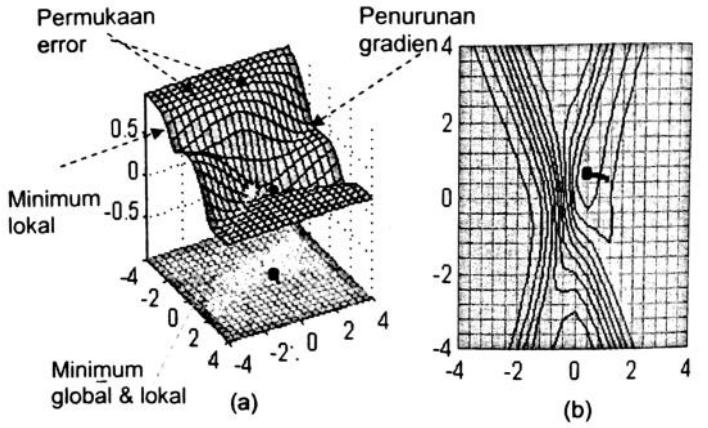
$$z_j = f(v_{0j} + \sum_{i=1}^{n} x_i v_{ij})$$

4. Setiap unit output y_k (dari unit ke-1 sampai unit ke-m; j=1,...,p; k=1,...,m) menghitung sinyal outputnya dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal input bagi lapisan ini, yaitu sinyal-sinyal input z_j dari lapisan tersembunyi:

$$y_k = f(w_{0k} + \sum_{j=1}^{p} z_j w_{jk})$$

Minimum Error Kuadrat

Gambar 5.3 menggambarkan pencarian error minimum pada permukaan error sepanjang gradien. Permukaan error kadang-kadang mengalami penurunan sampai mencapai suatu kondisi minimum namun kadang kala menaik lagi. Turunan yang paling curam (error yang paling minim, lihat Gambar 5.3 (a)) dari semua minimum-minimum lokal yang ada disebut minimum global. Kondisi minimum global merupakan kondisi yang diinginkan.



Gambar 5.3 Pencarian error minimum

(a) Permukaan error beserta minimum lokal dan minimum global
(b) Pergerakan pencarian error minimum setelah pelatihan berlangsung

Persyaratan minimasi error:

$$E = 1/2 \sum (t_k - y_k)^2$$

Pencarian error minimum pada saat pelatihan jaringan propagasi balik sedang berlangsung digambarkan secara lebih jelas pada Gambar 5.3(b).

5.3 Pilihan-pilihan dalam Pengaplikasian Metode Propagasi Balik

5.3.1 Fungsi Aktivasi

Ada beberapa pilihan fungsi aktivasi yang digunakan di dalam metode propagasi balik, seperti fungsi sigmoid biner, sigmoid bipolar, dan tengen hiperbolik. Karakteristik yang harus dimiliki fungsi aktivasi tersebut adalah kontinu, diferensiabel, dan tidak menurun secara monoton. Fungsi aktivasi diharapkan jenuh (mendekati nilai-nilai maksimum dan minimum secara asimtot).

Fungsi Sigmoid Biner

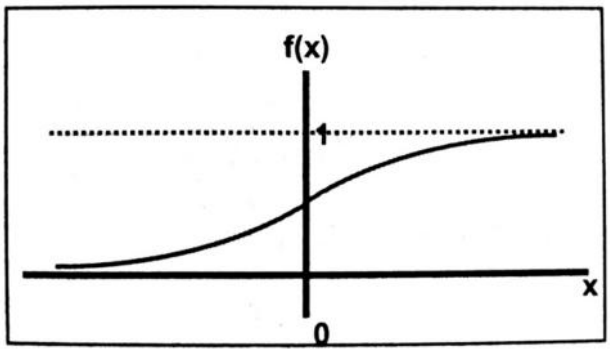
Fungsi ini merupakan fungsi yang umum digunakan. Rangenya adalah (0,1) dan didefinisikan sebagai:

$$f_1(x) = \frac{1}{1+e^2}$$

dengan aturan:

$$f'(x) = f_1(x)(1 - f_1(x))$$

fungsi sigmoid biner ini diilustrasikan pada Gambar 5.4



Gambar 5.4 Fungsi sigmoid biner dengan range (0,1)

Fungsi Sigmoid Bipolar

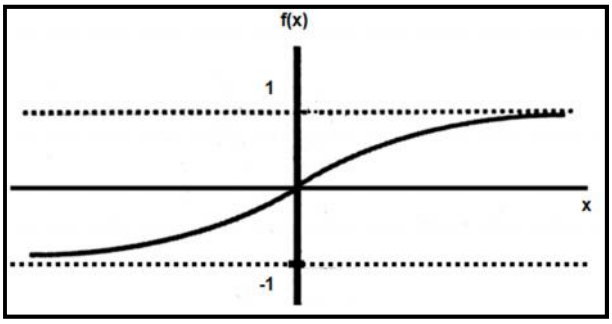
Fungsi sigmoid bipolar merupakan fungsi yang umum digunakan dan memiliki range (-1,1) yang didefinisikan sebagai:

$$f_2(x) = 2 f_1(x) - 1$$

dengan turunan

$$f_2'(x) = \frac{1}{2}(1 + f_2(x))(1 - f_2(x))$$

Fungsi sigmoid bipolar digambarkan pada Gambar 5.5



Gambar 5.5 Fungsi sigmoid bipolar range (-1,1)

Fungsi Tangen Hiperbolik

Fungsi tangen hiperbolik didefinisikan sebagai:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

$$\tanh'(x) = (1 + \tanh(x))(1 - \tanh(x))$$

5.3.2 Penginilisasian Bobot dan Bias

Inisialisasi Acak

Prosedur yang umum dilakukan adalah menginisialisasi bias dan bobot, baik dari unit input ke unit tersembunyi maupun dari unit tersembunyi ke output ke dalam sebuah interval tertentu (-Y

dan Ɣ), misalnya antara -0.4 sampai 0.4, -0.5 sampai 0.5, dan -1 sampai 1.

Inisialisasi Nguyen-Widrow

Waktu pembelajaran jaringan propagasi balik yang bobot dan biasnya ditandai dengan inisialisasi Nguyen-Widrow lebih capat dibandingkan bila diinisialisasi dengan inisialisasi acak. Pada inisialisasi Nguyen-Widrow, inisialisasi acak tetap terpakai tetapi digunakan untuk menginisialisasi bias dan bobot dari unit tersembunyi ke unit output saja. Untuk bias dan bobot dari unit-unit ke unit-unit tersembunyi digunakan bias dan bobot yang khusus diskala agar jatuh pada range tertentu. Dengan penskalan maka diharapkan kemampuan belajar dari unit-unit tersembunyi dapat meningkat.

Faktor skala Nguyen-Widrow (**β**) didefinisikan sebagai:

$$= 0.7(p)^{1/n}$$

di mana:
n = banyak unit input
p = banyak unit tersembunyi
β = faktor skala.

Prosedur inisialisasi Nguyen-Widrow

Untuk setiap unit tersembunyi dari unit ke-1 sampai ke-p:

1. Inisialisasi vektor bobot dari unit-unit input ke unit-unit tersembunyi
   (j = 1,...,p) dengan cara:
   - Menentukan bobot-bobot antara unit input ke unit tersembunyi (v_{ij}):
     v_{ij} (lama = bilangan acak antara − dan
     di mana $i = 1, \ldots, n$
   - Menghitung v_{ij}
   - Menginisialisasi kembali v_{ij} :

$$v_{ij} = \frac{\bullet v_{ij}(lama)}{v_{ij}}$$

2. Menentukan bias antara unit input ke unit tersembunyi (j = 1,...,p): v_{0j} diset dengan bilangan acak yang terletak pada skala antara – dan .

   Sebuah eksperimen menarik untuk memecahkan masalah XOR dengan jaringan propagasi balik 2-4-1 dilakukan dengan 3 cara representasi: representasi biner, bipolar, dan bipolar yang dimodifikasi, sedangkan inisialisasi bobot dan bias dilakukan masing-masing dengan cara acak dan Nguyen-Widrow. Adapun bipolar yang dimodifikasi adalah representasi bipolar yang nilai-nilai target terpilihnya tidak berada pada asimtot fungsi sigmoid dengan harapan agar konvergensi lebih cepat tercapai.

   Berikut ini jumlah epoch yang dibutuhkan masing-masing pada representasi biner, bipolar, dan bipolar yang dimodifikasi:
   - Dengan inisialisasi acak: 2891, 387, dan 264
   - Dengan inisialisasi Nguyen-Widrow: 1935, 224, dan 127.
     Dapat disimpulkan bahwa penggunaan inisialisasi Nguyen-Widrow mereduksi waktu yang dibutuhkan untuk melatih jaringan.

5.3.3 Jumlah Lapisan Tersembunyi yang Digunakan

Satu buah lapisan tersembunyi bisa dikatakan cukup memadai untuk menyelesaikan masalah sembarang fungsi pendekatan. Dengan menggunakan lebih dari satu buah lapisan tersembunyi, kadang-kadang suatu masalah lebih mudah untuk diselesaikan. Mengenai banyaknya jumlah lapisan tersembunyi yang dibutuhkan, tidak ada ketentuan khusus. Untuk itu hanya dibutuhkan sedikit modifikasi terhadap algoritma propagasi balik yang telah dibahas sebelumnya (pemodifikasian dilakukan dengan tetap berpegang pada konsep yang sama).

5.3.4 Pengupdatean Bobot dengan Momentum

Penambahan parameter momentum dalam mengupdate bobot seringkali bisa mempercepat proses pelatihan. Ini disebabkan karena momentum memaksa proses perubahan bobot terus bergerak sehingga tidak terperangkap dalam minimum-minimum lokal (lihat Gambar 5.3). untuk membuktikan hal ini dapat dilihat kembali persamaan:

$$\Delta w_{jk} = \sim ¢_k z_j$$

$$\Delta v_{ij} = \sim ¢_j x_i$$

Jika error terjadi (output aktual telah sama dengan output target) maka δ_k menjadi nol dan hal ini akan menyebabkan koreksi bobot $\Delta w_j = 0$, atau dengan kata lain peng-update-an bobot berlanjut dalam arah yang sama seperti sebelumnya.

Jika parameter momentum digunakan maka persamaan-persamaan peng-update-an bobot dengan langkah pelatihan t, dan t+1 untuk langkah pelatihan selanjutnya, mengalami modifikasi sebagai berikut:

$$\Delta w_{jk}(t+1) = \sim ¢_k z_j + « \Delta w_{jk}(t)$$

dan

$$\Delta v_{ij}(t+1) = \sim ¢_j x_1 + « \Delta v_{ij}(t)$$

dengan μ adalah parameter momentum dalam range antara 0 sampai 1.

5.4 Aplikasi

Salah satu contoh penggunaan aplikasi jaringan propagasi balik tahap demi tahap dapat dilihat pada bab 7.

C. Rangkuman

1. Jaringan syaraf tiruan propagasi balik terdiri dari banyak lapisan (*multilayer neural networks*):
   (a) Lapisan input (1 buah). Lapisan input terdiri dari neuron-neuron atau unit-unit input, mulai dari unit input sampai unit input n.
   (b) Lapisan tersembunyi (minimal 1). Lapisan tersembunyi terdiri dari unit-unit tersembunyi mulai dari unit tersembunyi 1 sampai unit tersembunyi p.
   (c) Lapisan output (1 buah). Lapisan output terdiri dari unit-unit output mulai dari unit output 1 sampai unit output m, n, p, m masing-masing adalah bilangan intgrer sembarang menurut arsitektur jaringan syaraf tiruan yang dirancang. V_{0j} dan W_{0k} masing-masing adalah bias untuk unit tersembunyi ke-j dan untuk unit ke-k bias V_{0j} dan W_{0k} berperilaku seperti bobot di mana output bias ini selalu sama dengan 1. V_{ij} adalah bobot koneksi antara unit ke-I lapisan input dengan unit ke-j lapisan tersembunyi dengan unit ke-j lapisan output.
2. Algoritma propagasi balik dapat dibagi ke dalam 2 bagian:
   (a) Algoritma pelatihan
   Terdiri dari 3 tahap: tahap umpan maju pola pelatihan input, tahap pemropagasibalikan error, dan tahap pengaturan bobot.
   (b) Algoritma aplikasi
   Yang digunakan hanyalah tahap umpan maju saja.
3. Pilihan-pilihan dalam pengaplikasian metode propagasi balik, antara lain:
   (a) Fungsi aktivasi
   (b) Penginilisasian bobot dan bias
   (c) Jumlah lapisan tersembunyi yang digunakan
   (d) Pengupdatean bobot dengan momentum

D. Tugas

Carilah referensi lain mengenai jaringan propagasi balik yang telah dibahas dan buatlah makalah!

E. Tes Formatif

1. Jelaskan pengertian jaringan propagasi balik!
2. Sebutkan contoh aplikasi jaringan propagasi balik yang anda ketahui!
3. Buatlah gambaran umum alur proses dari jaringan syaraf propagasi balik!

BAB 6

PEMBUATAN APLIKASI JARINGAN SYARAF TIRUAN

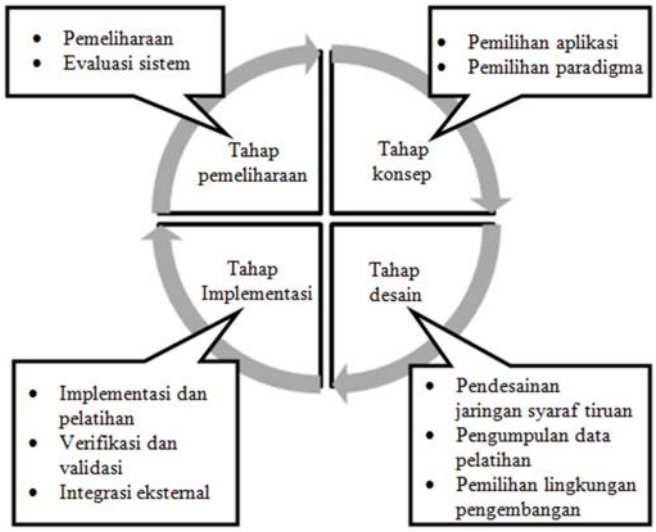
A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan siklus pembuatan aplikasi jaringan syaraf tiruan.
2. Menjelaskan mengenai tahap implementasi dalam pembuatan aplikasi jaringan syaraf tiruan.

B. Uraian Materi

6.1 Siklus Pembuatan Aplikasi Jaringan Syaraf Tiruan



Gambar 6.1 Siklus pembuatan aplikasi jaringan syaraf tiruan

Seperti halnya pembuatan perangkat lunak pada umumnya, pembuatan sebuah aplikasi jaringan syaraf tiruan juga melalui tahap-tahap atau metodologi pengembangan yang dapat dibagi ke dalam 4 tahap, yaitu tahap konsep, tahap desain, tahap implementasi, dan tahap pemeliharaan sistem. Namun sebelumnya perlu dipahami terlebih dahulu proses-proses komputing jaringan syaraf tiruan.

Masalah yang ditangani, dibawah pengawasan orang yang ahli dalam permasalahan tersebut, ditransformasikan ke dalam bentuk jaringan syaraf tiruan oleh insinyur *neurocomputing*. Langkah-langkahnya adalah sebagai berikut:

1. Mendeskripsikan permasalahan secara jelas. Pendeskripsian masalah secara jelas nantinya akan sangat membantu insinyur *neurocomputing* dalam merancang arsitektur jaringan syaraf tiruan.

2. Merepresentasikan pengetahuan yang dimiliki insinyur *neurocomputing* ke dalam bentuk set data pelatihan dan pengujian (untuk jaringan syaraf tiruan terawasi). Di sini pakar bertuujuan untuk menjamin keakurasian dari set-set data tersebut.
3. Memilih teknik representasi data yang akan digunakan.
4. Set data diubah ke dalam format presentasi yang layak, termasuk di dalamnya adalah perancangan arsitektur jaringan syaraf tiruan yang optimal untuk mewakili perpresentasian masalah. Format yang layak akan berpengaruh pada kualitas dan unjuk kerja jaringan selama menjani proses pelatihan.
5. *Coding* atau pemilihan lingkungan pengembangan. Dewasa ini telah banyak tersedia perangkat lunak pengembang jaringan syaraf tiruan. Tinggal diputuskan akan memakai peragkat lunak pengembangan yang tersedia di pasaran atau membuat program sendiri.
6. Melatih jaringan syaraf tiruan sampai sebuah tingkat akurasi yang ditetapkan tercapai.
7. Menguji jaringan syaraf tiruan dengan menggunakan set pengujian yang berisikan contoh-contoh yang hasil outputnya sudah diketahui sebelumnya.
8. Verifikasi dan validasi jaringan syaraf tiruan.

6.2 Tahap Konsep

6.2.1 Pemilihan aplikasi

Pada tahap ini masalah yang ada perlu diselidiki terlebih dahulu, apakah aplikasi untuk masalah itu memang layak menggunakan jaringan syaraf tiruan. Lazimnya aplikasi-aplikasi yang bisa diselesaikan dengan baik dengan menggunakan teknik-teknik algoritma atau aplikasi-aplikasi yang membutuhkan deduksi dan logika sekuensial dipecahkan dengan teknik lain semisal sistem pakar.

Adapun pertimbangan dalam penggunaan teknik jaringan syaraf tiruan antara lain:

a. Tidak memadainya basis pengetahuan

Hal ini mungkin disebabkan beberapa hal:

- Pakar yang tidak tersedia
  Ketidak tersediaan pakar membuat wawancara mengenai aturan-aturan pengetahuan yang berlaku tidak mungkin dilakukan. Oleh karena itu pengumpulan data penyusunan basis pengetahuan harus dilakukan dengan cara lain.
- Aturan pengetahuan yang sulit dirumuskan
  Meskipun aturan pengetahuan sulit dirumuskan namun bial data historis tersedia dalam jumlah besar maka jaringan syaraf tiruan layak dipakai.

b. Basis pengetahuan yang tidak tetap

Pengupdatean harus sering dilakukan guna menambah perbendaharaan pengetahuan dari aplikasi jaringan syaraf tiruan. Tambahan pengetahuan tinggal disisipkan ke dalam set-set data untuk dilatih ulang. Pemodifikasian pada jaringan syaraf tiruan berlangsung jauh lebih mudah dari pada pemodifikasian pada sistem pakar. Pada sistem pakar, selain aturan pengetahuan harus dimodifikasi, pendebugan program juga harus dilakukan kembali.

c. Sistem data yang intensif

Didalam sistem data yang intensif, input datanya dalam jumlah yang besar dan perlu pemrosesan secara cepat. Selain itu input-inputnya ambigu, mengandung nois, atau cenderung mengalami error.

d. Analisis regresi

Pada jaringan syaraf tiruan, masalah-masalah yang bisa diselesaikan dengan menggunakan analisis regresi diselesaikan dengan memanfaatkan data statistikal untuk mencari pola dan kecenderungannya.

e. Perangkat keras paralel

Tersedianya perangkat keras paralel jaringan syaraf tiruan yang murah merupakan keuntungan karena akan memperbaiki aplikasi jaringan syaraf tiruan, misalnya di bidang robotika.

Beberapa contoh perangkat keras yang mengimplementasikan model jaringan syaraf tiruan:

- Spert-II (*synthetic Perceptron Testbed II*) merupakan prototip mikroprosesor vektor *full-custom*. Arsitektur mikroprosesor vektor akan membuat elemen-elemen pengolahan untuk aplikasi multimedia dan atar muka manusia-mesin yang sering berisi algoritma berbentuk data paralel. Spert-II mempercepat pelatihan multiparameter jaringan syaraf tiruan untuk pengenalan suara.
- *Corprocessor board* yang khusus dirancang untuk aplikasi mesin yang bisa mengkonversi suara ke dalam tulisan (*neural phonetic typewriter*).
- Penelitian mengenai pengimplementasian jaringan syaraf tiruan pada sirkuit VLSI CMOS telah banyak dilakukan. Salah satunya adalah pengimplementasian model jaringan syaraf tiruan pada sirkuit VLSI CMOS sebagai memori asosiatif maupun sebagai pengklasifikasi pola.

6.2.2 Pemilihan Paradigma

Dalam pemilihan paradigma pembelajaran jaringan syaraf tiruan, hal-hal yang harus diperhatikan adalah sebagai berikut:

1. Ukuran jaringan
   Pada saat mendesain jaringan syaraf tiruan, kebutuhan node dan lapisan dapat diperkirakan. Untuk jaringan berukuran besar maka tentunya tidak mungkin menggunakan metode pembelajaran yang hanya mampu menampung jaringan kecil. Contohnya, teknik Hopfield menampung jaringan yang lebih kecil dari teknik propagasi balik yang mampu menampung jaringan yang lebih kecil dari teknik propagasi balik yang mampu menampung jaringan pengukuran yang lebih besar. Kapasitas pola yang bisa disimpang jaringan propagasi balik relatif lebih besar dari pada jaringan hopfield.
2. Pembawaan input dan output

Pembawaan input dan output, misalnya matriks angka cocok untuk optimisasi, diskrit cocok untuk klasifikasi dan pengenalan poa.

3. Mekanisme memori
   Mekanisme memori ada 2 macam, yaitu memori autoasosiatif dan memori heteroasosiatif.
   - Memori autoasosiatif: mengambil informasi secara parsial kemudian digunakan untuk merekonstruksi pola-pola yang sudah tersimpan sebelumnya.
   - Memori heteroasosiatif: sebuah jaringan syaraf tiruan yang didesain menghubungkan pasangan pola input dengan pola output, di mana pola input dan outputnya tidak sama (vektor output ≠ vektor input)
4. Tipe pelatihan: terawasi dan tidak terawasi
   - Pelatihan terawasi: pelatihan yang menghadirkan serangkaian vektor input pelatihan yang masing-masing berhubungan dengan vektor output yang menjadi targetnya.
   - Pelatihan tidak terawasi: jaringan syaraf tiruan yang mengorganisasi dirinya sendiri dengan membentuk vektor-vektor input yang sama tanpa menggunakan data pelatihan untuk menunjukkan vektor-vektor tersebut termasuk ke dalam kelompok yang mana.
5. Batasan waktu operasi rutin dari sistem berjalan.
   Batasan waktu berbeda-beda tergantung dari model dan ukuran jaringan serta kualitas data.

6.3 Tahap Desain

Terbagi tiga tahap, yaitu tahap pendesainan jaringan syaraf tiruan, tahap pengumpulan data, dan tahap pemilihan lingkungan pengembangan.

6.3.1 Pendesainan Jaringan Syaraf Tiruan

Ada 3 tingkat pendesainan:

1. Desain node

- Menentukan tipe input
- Melakukan pemilihan fungsi transfer

2. Desain jaringan
   - Menentukan banyaknya lapisan yang akan digunakan
   - Menentukan ukuran setiap lapisan.
   - Merancang desain output.
3. Desain pelatihan
   Menentukan parameter-parameter jaringan (angka pembelajaran, nilai ambang). Angka pembelajaran digunakan untuk mengukur ukuran langkah jaringan yang tengah didesain, sedangkan nilai ambang digunakan untuk menghentikan pelatihan .

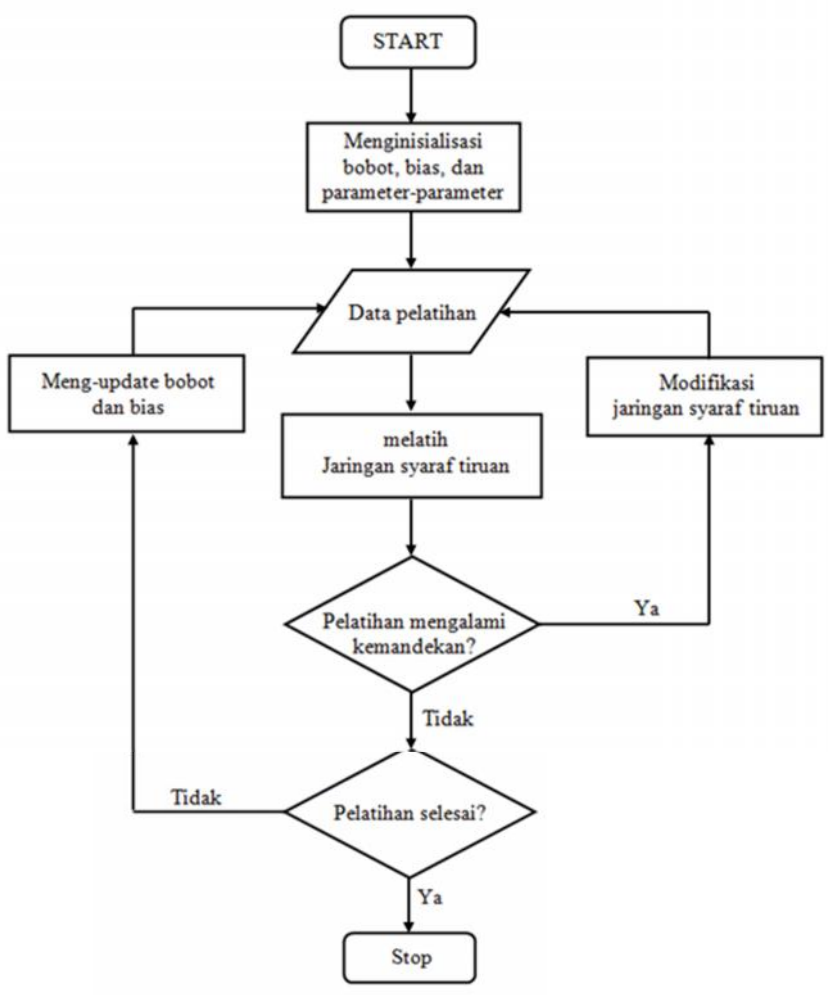
6.3.2 Pengumpulan Data

1. Mengumpulkan data yang dapat dipercaya
2. Membagi data tersebut ke dalam set pelatihan dan set tes. Set pelatihan digunakam untuk melatih jaringan, sedangkan set tes setelah pelatihan jaringan selesai, yaitu untuk menguji jaringan apakah jaringan menghasilkan output sesuai dengan yang diinginkan pada waktu input yang belum pernah dipelajari oleh jaringan dimasukkan.

6.3.3 Pemilihan Lingkungan Pengembangan

Yang perlu dipertimbangkan pada tahap ini adalah pertimbangan waktu dan biaya. Saat ini telah banyak perangkat lunak pengembang jaringan syaraf tiruan yang tersedia di pasaran dengan beragam harga dan platform masing-masing.

6.4 Tahap Implementasi



Gambar 6.2 prosedur pelatihan jaringan syaraf tiruan

6.4.1 Implementasi dan pelatihan

1. Mendesain penyimpanan data dan memanipulasi data agar data bisa berada dalam tipe dan format yang layak.
2. Melakukan prosedur pelatihan jaringan syaraf tiruan seperti pada gambar 6.2
3. Metode-metode untuk memperbaiki proses pelatihan jaringan:
   a) Membatasi jaringan
      Semakin besar jaringan maka akan semakin menjadi kompleks. Sebaiknya jaringan dibatasi maksimal 3 lapis dengan jumlah node kurang dari 300 node.
   b) Memperluas lapisan tengah
      Pelatihan jaringan yang memakan waktu terlalu lama bisa dikurangi dengan menambah node pada lapisan tengah dengan ketentuan ukuran node <300.
   c) Menggunakan momentum
      Dengan menggunakan momentum, pelatihan dapat terus berlanjut, yaitu dengan cara meng-update bobotnya dengan menambahkan pecahan bobot terakhir terhadap bobot.
   d) Memperbesar nilai toleransi perbedaan niai output terhadap nilai target yang didizinkan pada jaringan.
   e) Mengatur jaringan.
   f) Menambahkan data yang mengandung noise
      Selain menambah eefisien pelatihan juga membuat jaringan mampu menghasilkan output yang bagus meski input yang dimasukkan tidak lengkap.
   g) Memulai lagi dari awal prosedur prlatihan jaringan syaraf tiruan.

Overtraining

Hal yang perlu dihindari adalah jangan sampai melatih jaaringan syaraf secara berlebihan (*overtraining*). Contohnya seperti dengan 200 epoch keluaran jaringan sudah bagus tetapi pelatihan terus diteruskan sampai 1000 kali epoch dengan anggapan bahwa semakin sering dilatih maka keluaran jaringan akan semakin baik. Anggapan tersebut tidak benar. Jika pola-pola pelatihan yang sama dimasukkan ke jaringan berulang kalli maka bobot-bobot akan diatur semakin

mendekati output yang diinginkan. Ini sama halnya dengan memaksa jaringan syaraf untuk menghafal pola-pola (memorisasi) dan bukan mengambil maksud dari relasi yang ada (generalisasi). Jaringan syaraf tiruan akan bagus dalam memberikan keluaran untuk data pelatihan tetapi tidak memberikan unjuk kerja yang bagus ketika ia diuji dengan pola-pola yang belum pernah dipelajarinya. Tujuan dari pelatihan jaringan adalah keseimbangan antara kemampuan memorisasi dan generalisasi. Jaringan syaraf tiruan diharapkan tidak hanya baik dalam memprediksi pada set testing dan data validasi. Tool-tool komersial jaringan syaraf tiruan biasanya menyediakan pergantian anatara data set testing untuk mengecek unjuk kerja jaringan pada data testing.

Teknik penambahan dan pemangkasan jaringan

1. Dari jaringan besar ke kecil
   Arsitektur jaringan dibangun berdasarkan data, membentuk jaringan yang besar dan kemudian memangkas node-node dan koneksi-koneksi sampai menjadi jaringan terkecil yang bisa melakukan aplikasi yang diinginkan. Algoritma genetika (GA = *Genetic Alghorithm*) sering digunakan untuk mengoptimasi fungsi-fungsi menggunakan metode pencarian paralel tersebut berdasarkan teori evolusi biologis. Disini pemiihan banyaknya lapisan tersembunyi dan unit-unit tersembunyi dipandang sebagai sebuah masalah optimasi, dan GA bisa digunakan untuk menemukan arsitektur optimumnya.
   Untuk mengetahui yanng mana yang harus dipangkas, lakukan analisis pada bobot-bobot yang terkoneksi pada sebuah node, apakah benar bobot tersebut membawa ke arah prediksi yang akurat dari pola output. Jika ditemukan bobot tersebut nilainya sangat kecil maka pangkas saja node tersembunyi tersebut. Proses yang menyebabkan penurunan unjuk kerja pada set testing habis dipangkas.
2. Dari jaringan kecil ke besar
   Dimulai dengan pembangunan sebuah jaringan kecil, kemudian secara bertahap dilakukan penambahan node-node tersembunyi (dan bobot-bobot). Error prediksi jaringan dipantau dan

sepanjang unjuk kerja jaringan pada set testing meningkat sehingga unit-unit tersembunyi terus ditambahkan. Jaringan syaraf dengan korelasi bertingkat mengalokasikan sekumpulan node potensial baru yang kemudian dipersaingkan satu sam lain, lalu node yang paling besar dalam mereduksi error dalam presiksi akan dipilih sebagai node yang ditambahkan ke jaringan.

6.4.2 Verifikasi dan Validasi (V dan V)

Verifikasi adalah pembuktian bahwa sistem telah dibangun secara layak ditinjau dari segi mekanis. Validasi adalah pengecekan apakah sistem yang dibangun sesuai dengan kebutuhan yang sudah diidentifikasi sebelumnya pada awal proyek. Pendekatan V dan V yang paling umum digunakan untuk jaringan syaraf tiruan adalah pengujian *trial-and-error*.

Verifikasi

Beberapa aspek deklaratif dalam verifikasi jaringan syaraf tiruan:

1. Jaringan
   a. Apakah arsitektur jaringan syaraf tiruan berbentuk lapisan tunggal atau multilapis?
   b. Apakah aliran informasi jaringan adalah umpan maju atau berulang?
   c. Apakah pola koneksinya terhubung penuh atau terhubung sebagian?
2. Unit
   a. Bagaimana koneksi inputnya?
   b. Bagaimana koneksi outputnya?
   c. Mengenai nonlinearitas, apakah hard limit atau sigmoid atau tangen hiperbola?
3. Perilaku
   a. Bagaimana inisialisasi bobotnya?
   b. Bagaimana perhitungan aktivasinya?
   c. Bagaimana pengupdatean bobotnya?

Hal lain yang perlu diverifikasi adalah apakah persamaan-persamaan matematika dri teknik jaringan syaraf tiruan yang dipilih sudah diimplementasikan dengan benar.

Validasi

Untuk pemvalidasian jaringan syaraf tiruan, terdapat dua kali validasi yaitu pada saat pelatihan dan setelah pelatihan berakhir. Pada saat pelatihan, jaringan syaraf tiruan dipandang sebagai sebuah sistem pembelajaran. Setelah pelatihan, jaringan syaraf tiruan dipandang sebagai sebuah sistem kerja (*aplication system*).

Aspek-aspek validasi jaringan syaraf tiruan sebagai sebuah sistem pembelajaran:

a. Aspek untuk kerja
   Apakah kerja suatu sistem sebagai pemecah masalah menghasilkan solusi yang memuaskan dalam menghadapi berbagai kondisi baik data normal, data terkena noise, data tidak lengkap, atau data yang tidak relevan?
   Untuk menjawab pertanyaan tersebut kemampuan generalisasi jaringan syaraf tiruan harus teruji.
b. Aspek keumuman (generalis)
   Semakin baik jika bisa diaplikasikan untuk memecahkan banyak masalah.
c. Aspek stabilitas
   Apakah pengetahuan yang pernah dipelajari oleh jaringan syaraf tiruan tetap dipelihara.
d. Aspek efesiensi
   Aspek efesiensi merupakan kriteria yang lemah. Di sini dibicarakan mengenai kompleksitas algoritma pembelajaran dan kecepatan pembelajaran. Walaupun jaringan sysaraf tiruan belajar dengan lambat tetapi akurat yang ditawarkan lebih tinggi maka jaringan syaraf tiruan tersebut tetap dapat diterima.
e. Aspek kualitas
   Yang dilihat dari aspek kualitas adalah seberapa menarik atau seberapa besar menfaat dari sistem.
f. Aspek konsistensi
   Apakah sebuah sistem pembelajaran bisa membangkitkan siste dengan unjuk kerja yang konsisten berdasarkan data yang sama?
g. Aspek penambahan pembelajaran

Aspek ini penting untuk sistem pembelajaran yang terus menerima input dari lingkungan dunia nyata.

h. Aspek sensivitas
   Unjuk kerja sebuah sistem pembelajaran bersifat sensitif terhadap perubahan parameter sistem ataupun pada perubahan yang terjadi pada data pelatihan.

Aspek-aspek validasi jaringan syaraf tiruan sebagai sebuah sistem kerja:

a. Keakurasian
   Keakurasian adalah ketepatan dalam menjawab permasalahan
b. Kekuatan (*rebustness*)
   Sistem memiliki unjuk kerja yang memadai disegala keadaan.
c. Efisiensi
   Seberapa cepat sistem pada jawaban yang benar dan seberapa besar memori yang dibutuhkan.
d. Konsistensi
   Sistem yang buruk hanya menunjukkan penanganan yang bagus pada kasus-kasus tertentu.
e. Kemampuan adaptasi dan kemampuan perluasan (*adaptability and extensibility*)
   Sistem akan sangat terbatas penggunaannya jika tidak bisa dimodifikasi atau diperluas.
f. Keandalan dan toleransi kesalahan (*reability and fault tolerence*)
   Seberapa jauh penurunan unjuk kerja jaringan bila sebagian dari sistem mengalami kegagalan.
g. Sensivitas
   sistem harus cukup sensitif mendeteksi perubahan pada data input (walapun kecil) tetapi tetap menghasilkan outoput yang tepat. Selain itu yang harus dihindari adalah sistem yang terlalu sensitif terhadap noise sehingga sistem menghasilkan konklusi yang salah.

Teknik Validasi

Beberapa teknik validasi untuk memperkirakan teknik kesalahan:

1. *Hold out*
   Data dibagi ke dalam 2 set terpisah. Yang satu dipergunakan untuk pelatihan (*set training*) dan yang satunya lagi digunakan untuk pengujian (*set testing*). Perlu kasus pengujian dalam jumlah yang cukup untuk menjamin pengestimasian yang bisa dipercaya. Metode ini cocok untuk contoh-contoh kasus dalam jumalah besar.

   Contoh pembagian data yang adil untuk set training dan set testing untuk kasus penentuan penerimaan lamaran kartu kredit menggunakan jaringan syaraf tiruan.

   Dipunyai basis data yang terdiri dari 31000 contoh pelamar dengan kategori: risiko rendah, risiko tak dapat ditentukan, dan risiko tinggi. Misalnya 10 persen dianggap sudah mewakili. Set testing ini kemudian diambil secara sekuensial menyeluruh dengan cara mengambil setiap kelipatan 10 dari data. Hasilnya adalah sebagai berikut:

Kategori	Data awal	Set testing (10%)	Set training	Contoh tak terpakai
Risiko rendah	17284	1739	6179	9366
Risiko tak dapat ditentukan	7523	755	6179	589
Risiko tinggi	6851	672	6179	0
Total:	**31658**	**3166**	**18537**	**9955**

   Pada contoh di atas, untuk memilih set training, carilah kuantitas data terkecil, yaitu “risiko tinggi” (sebanyak 6851 contoh) kemudian kurangi dengan set tetingnya (672) diperoleh set training untuk risiko tinggi sebesar 6179. Kategori lainnya kemudian mengikuti kuantitas terkecil ini. Dalam melakukan pelatihan jaringan syaraf, set training dari setiap kategoti diambil

secara acak untuk menghindari sembarang bias yang mungkin ada disebabkan pengurutan yang biasa dikenakan pada basis data. Biasanya dalam pelatihan, setiap ketegori dimasukkan secara bergantian, berselang-seling (coontoh: risiko rendah – risiko tak dapat ditentukan – risiko tinggi – risiko rendah – risiko tak dapat ditentukan – risiko tinggi dan seterusnya berulang). Maksud dari pnyajian secara berselang-seling ini adalah agar jaringan tidak melupakan contoh-contoh yang pernah dipelajari atau sebaliknya terlalu ingat kepada pola-pola terakhir yang pernah dipelajari. Ingat bahwa tujuan pembelajaran jaringan syaraf tiruan adalah keseimbangan antara kemampuan memorisasi dan kemmampuan generalisasi.

2. *Leave one out*
   Mengulang sebanyak n kali untuk contoh-contoh kasus sejumlah n dengan setiap kali mengeluarkan satu kasus untuk pengujian. Angka perkiraan kesalahan adalah angka kesalahan pengujian rata-rata n kali percobaan. Metode ini cocok untuk contoh-contoh kasus dalam jumlah yang kecil (sedikit).
   Contoh:
   Terdapat data berisikan contoh-contoh kasus 1,2,3,4,5,6,7,8. Eksperimen dilakukan dengan cara sebagai berikut:

Eksperimen ke-	Pelatihan	Pengujian
1	1,2,3,4,5,6,7	8
2	1,2,3,4,5,6,8	7
3	1,2,3,4,5,7,8	6
4	1,2,3,4,6,7,8	5
5	1,2,3,5,6,7,8	4
6	1,2,4,5,6,7,8	3
7	1,3,4,5,6,7,8	2
8	2,3,4,5,6,7,8	1

Angka kesalahan pengujian misalnya 0.00, 1.00, 0.00, 1.00, 1.00, 0.00, 1.00 sehingga angka eerror rata-ratanya adalah 0.5. kisaran angka error adalah antara 1 dan 0 sehingga semakin kecil nilainya semakin tangguh jaringan tersebut.

3. *Cross-validation*
   Teknik ini juga disebut dengan *leave some* atau *leave one out*. *K-fold cross validation* cocok untuk data baik dalam jumlah besar maupun kecil.
   Contoh:
   Terdapat data berisikan contoh-contoh kasus 1, 2, 3, 4, 5, 6, 7, 8. Eksperimen bisa dilakukan antara lain dengan cara sebagai berikut (*four-fold-validation*):

Eksperimen ke-	Pelatihan	Pengujian
1	1,2,3,4,5,6	7,8
2	1,2,3,4,7,8	5,6
3	1,2,5,6,7,8	3,4
4	3,4,5,6,7,8	1,2

Atau dengan *two-fold cross validation*:

Eksperimen ke-	Pelatihan	Pengujian
1	1,2,3,4	5,6,7,8
2	5,6,7,8	1,2,3,4

Validasi terhadap sistem berjalan diterapkan melalui uji lapangan atau uji turing. Uji lapangan di mana sistem diterapkan di lapangan dengan dipantau oleh pemakai. Uji turinng yaitu membandingkan hasil yang diperoleh sistem dengan hasil yang diperoleh oleh pakar di mana sistem valid jika manusia tidak bisa mengetahui hasil yang dikeluarkan oleh sistem.

6.4.3 Integrasi eksternal

Bila jaringan syaraf tiruan merupakan bagian dari sistem yang lebih besar maka dibutuhkan antarmuka yang bagus terhadap sistem informasi lainnya, juga terhadap perangkat input/output serta terhadap pemakai simulator. Misalnya untuk manipulasi input/output mungkin saja diguunakan sinyal digitizer.

6.5 Tahap pemeliharaan

Untuk memelihara keakuratan data agar tidak menurun atau untuk memnuhi kebutuhan sistem yang meningkat sebaiknya dilakukan pemeliharaan sistem dengan melatih kembali jaringan (misalnya dengan menyegarkan jaringan syaraf tiruan dengan data yang baru) dan mengevaluasi ulang output yang dihasilkan.

Perubahan besar-besaran mungkin juga dilakukan bila meyangkut perubahan antar muka jaringan syaraf tiruan. Hal ini mengakibatkan pembangunan sistem harus dimulai kembali dari tahap desain.

C. Rangkuman

1. Siklus pembuatan aplikasi jaringan syaraf tiruan terdiri dari 4 tahap, yaitu:
   a. Tahap konsep
   b. Tahap desain
   c. Tahap implementasi
   d. Tahap pemeliharaan
2. Tahap konsep dalam siklus pembuatan aplikasi jaringan syaraf tiruan meliputi beberapa bagian, yaitu:
   a. Pemilihan aplikasi
   b. Pemilihan paradigma
3. Tahap desain meliputi:
   a. Pendesainan jaringan syaraf tiruan
   b. Pengumpulan data
   c. Pemilihan lingkungan pengembangan
4. Tahap implementasi meliputi:
   a. Implementasi dan pelatihan
   b. Verifikasi dan validasi
   c. Integrasi eksternal
5. Pada pemilihan aplikasi, pertimbangan dalam penggunaan teknik jaringan syaraf tiruan, meliputi:
   a. Tidak memadainya basis pengetahuan
   b. Basis pengetahuan yang tidak tetap
   c. Sistem data yang intensif
   d. Analisis regresi
   e. Perangkat keras paralel

D. Tugas

1. Carilah referensi lain mengenai pembuatan aplikasi jaringan syaraf tiruan!
2. Buatlah kesimpulan mengenai siklus pembuatan aplikasi jaringan syaraf tiruan pada bab 8 ini!

E. Tes formatif

1. Sebutkan dan jelaskan beberapa contoh perangkat keras yang mengimplementasikan model jaringan syaraf tiruan!
2. Hal-hal apa saja yang harus diperhatikan dalam pemilihan paradigma?
3. Sebutkan 3 aspek deklaratif dalam verifikasi jaringan syaraf tiruan!

BAB 7

APLIKASI PENGENALAN KARAKTER ALFANUMERIK MENGGUNAKAN METODE PROPAGASI BALIK

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan mengenai tahap konsep pada aplikasi pengenalan karakter alfanumerik menggunakan metode propagasi balik.
2. Menjelaskan mengenai tahap desain pada aplikasi pengenalan karakter alfanumerik menggunakan metode propagasi balik.
3. Menjelaskan mengenai tahap implementasi pada aplikasi pengenalan karakter alfanumerik menggunakan metode propagasi balik.
4. Mendiskusikan tahapan implementasi pada aplikasi pengenalan karakter alfanumerik menggunakan propagasi balik

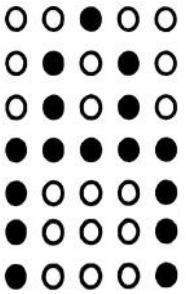
B. Uraian Materi

7.1 Tahap konsep

7.1.1 Pemilihan aplikasi

Salah satu contoh aplikasi pengenalan pola yang cukup kompleks adalah pengenalan karakter allfanumerik (['A'...'Z'] dan ['0'...'9']).

Diasumsikan ingin dibuat sebuah desain program komputer yang bertugas mengenali karakter alfanumerik dengan jalan menerjemahkan sebuah matriks berukuran 5x7 yang berisikan bilangan-bilangan biner yang menggambarkan citra pixel (*picture element*) pemetaan-bit (*bit-mapped pixed image*) dari sebuah karakter alfanumerik ke dalam bentuk kode ASCII 8 bit. Masalah ini digambarkan seperti pada Gambar 7.1.



$= 0010001010010101111111000110001100001_2$
$= 114AFC631_{16}$
$= 65_{10}$ ASCII

Gambar 7.1 setiap citra karakter dipetakan ke dalam kode ASCII yang ditunjuknya

Tidak tersedia fungsi matematika yang jelas yang bisa menghasilkan translasi yang diinginkan membuat masalah penerjemahan karakter alfanumerik ini menjadi cukup memusingkan karena untuk menghasilkan translasi yang salah satu caranya adalah dengan melakukan korelasi pixel demi pixel dibutuhkan waktu yang relatif terlalu lama. Solusi lain yang ditawarkan adaalah dengan menggunakan sebuah tabel lookup berupa sebuah array linear dengan struktur sebagai berikut:

record AELEMENT =

```
        pattern : long integrer;
        ascii   : byte;
    end record;
```

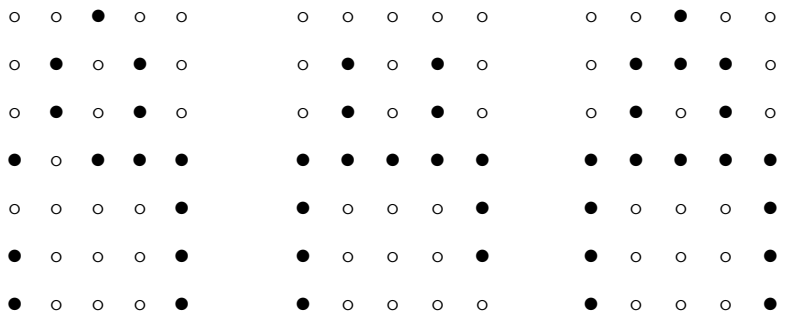
terdapat dua hal yang perlu mendapatkan perhatian:

a. $00100010100101011111100011000110001_2$ yang merupakan persamaan numerik dari kode pola-bit dihasilkan dengan jalan memindahkan 7 baris matriks ke dalam sebuah baris dengan hasilnya berupa sebuah angka biner 35 bit.
b. Kode ASCII beserta asosiasi karakternya. Algoritma yang digunakan untuk proses konversi ke ASCII adalah sebagai berikut:

```
function TRANSLATE (INPUT : long integrer;
                    LUT : ^AELEMENT[] return ascii;
{konversi matriks-pixel ke dalam bentuk karakter ASCII}

var TABLE : ^AELEMENT[];
found     : boolean;
I     : integrer;

begin
    TABLE = LUT;        {menempatkan ke table translasi)
    found = false;  {translasi belum ditemukan}

    for i=1 to length (TABLE) do {untuk semua item pada TABLE}
        if TABLE[i].patern=INPUT
        then Found=True; Exit;
                        {translasi ditemukan, keluar dari loop}
    end;

    if Found
    Then return TABLE[i].ascii   {mengembalikan ascii}
```

```
        Else return()
End;
```

Terlihat bahwa pendekatan tabel lookup ini cukup cepat dan mudah. Sayangnya, di dalam sistem nyata ada situasi-situasi yang tidak dapat ditangani dengan metode ini karena sangat dimungkinkan sebuah citra yang dibaca ternyata mengandung noise atau kurang lengkap. Adanya noise atau citra yang kurang lengkap mengakibatkan algoritma lookup akan mengembalikan kode ASCII yang salah sebab di dalam proses pencocokan antara pola input dan pola asosiasi target tidak menghasilkan karakter yang dimaksudkan.



Gambar 7.2 Contoh citra karakter A yang tidak lengkap ((a) dan (b)) atau mengandung noise (c)

Anggaplah masalah citra yang mengandung noise atau kurang lengkap dapat diselesaikan dengan menambah sejumlah program (yang berarti penambahan waktu yang dihabiskan CPU) ke dalam algoritma tabel lookup. Penambahan program ini akan memperbaiki kemampuan komputer untuk menebak pada karakter mana yang seharusnya terletak citra yang mengandung noise. Pada bit tunggal adalah cukup mudah untuk menemukan dan kemudian mengoreksi error. Namun demikian akan menjadi sulit bila error terjadi bukan hanya pada bit tunggal, melainkan pada banyak bit. Selain itu, pengeliminasian noise dari pola input untuk kemudian diterjemahkan ke dalam ASCII akan menghabiskan banyak CPU.

Solusi untuk masalah pengenalan karakter alfanumerik ini adalah dengan menggunakan jaringan syaraf tiruan. Dengan memanfaatkan sifat paralel jaringan syaraf tiruan maka waktu yang dibutuhkan oleh sebuah prosesor sekuensial untuk melakukan pemetaan dapat dikurangi.

Dengan melihat pertimbangan-pertimbangan di atas maka penggunaan jaringan syaraf tiruan untuk pengenalan karakter alfanumerik ini dipilih berdasarkan pertimbangan 1 dan 3. Adapun alasan mengapa basis pengetahuannya dikatakan tidak memadai adalah karena aturan-aturannya sangat suit untuk dirumuskan. Juga pengenalan karakter alfanumerik ini melibatkan data yang mengandung noise cukup besar (perlu sistem data intensif).

7.1.2 Pemilihan Paradigma

Dalam pemilihan paradigma, hal-hal yang harus diperhatikan antara lain adalah ukuran yang dirancang, pembawaan input dan output, tipe pelatihan, dan waktu operasi rutin dari sistem berjalan. Mengenai kemampuan jaringan syaraf tiruan dapat dilihat pada Tabel 7.1.

Tabel 7.1 Kemampuan Jaringan Syaraf Tiruan

Paradigma	**Model Pelatihan**	**Waktu Pelatihan**	**Waktu Eksekusi**
Propagasi balik	Terawasi	Lambat	Cepat
ART2	Tidak terawasi	Cepat	Cepat
Kohonen	Tidak terawasi	Sedang	Cepat
Hopfield	Terawasi	Cepat	Sedang
Boltzmann	Terawasi	Lambat	Lambat

Metode propagasi balik dipilih oleh penulis sebagai solusi kasus pengenalan karakter alfanumerik dengan pertimbangan bahwa pembawaan dari input dan outputnya dalam bentuk diskrit (biner). Tipe pelatihannya adalah pelatihan terawasi.

7.2 Tahap Desain

7.2.1 Pendesainan Jaringan Syaraf Tiruan

Ada 3 tingkat dalam pendesainan sebuah sistem jaringan syaraf tiruan, yaitu tingkat node, tingkat jaringan, dan tingkat pelatihan. Masing-masing tingkatan tersebut akan diterangkan berikut ini.

Tingkat node

Melihat karakteristik dari kakter alfanumerik yang merupakan feature maka tipe input yang cocok adalah yang bertipe biner. Demikian pula dengan tipe outputnya. Fungsi transfer yang akan digunakan adalah fungsi transfer sigmoid biner dengan alasan bahwa fungsi ini yang paling umum digunakan untuk metode propagasi balik. Pemilihan ini dengan memperhatikan beberapa karakteristik penting untuk jaringan propagasi balik, yaitu kontinu, diferensiabel, tidak menurun secara monoton, dan turunannya mudah melakukan perhitungan (hal ini untuk keefisienan perhitungan).

Fungsi transfer sigmoid biner (range antara 0 dan 1) didefinisikan sebagai:

$$f_1(x) = \frac{1}{1+e^{-x}}$$

dengan turunannya

$$f_1'(x) = f_1(x).[1 - f_1(x)]$$

Tingkat Jaringan

Pada tingkat jaringan dilakukan penentuan banyaknya lapisan. Seperti telah dinyatakan sebelumnya bahwa jaringan propagasi balik merupakan jaringan multilayer dalam pengertian bahwa jaringan dapat memiliki lapisan tersembunyi yang berjumlah lebih dari satu. Namun demikian, dalam jaringan syaraf tiruan untuk aplikasi pengenalan karakter alfanumerik ini hanya akan digunakan lapisan tersembunyi sebanyak sebuah saja, sehingga

secara keseluruhan jaringan propagasi balik untuk pengenalan karakter alfanumerik ini hanya terdiri dari 3 lapisan, yaitu sebuah lapisan input, sebuah lapisan tersembunyi, dan sebuah lapisan output.

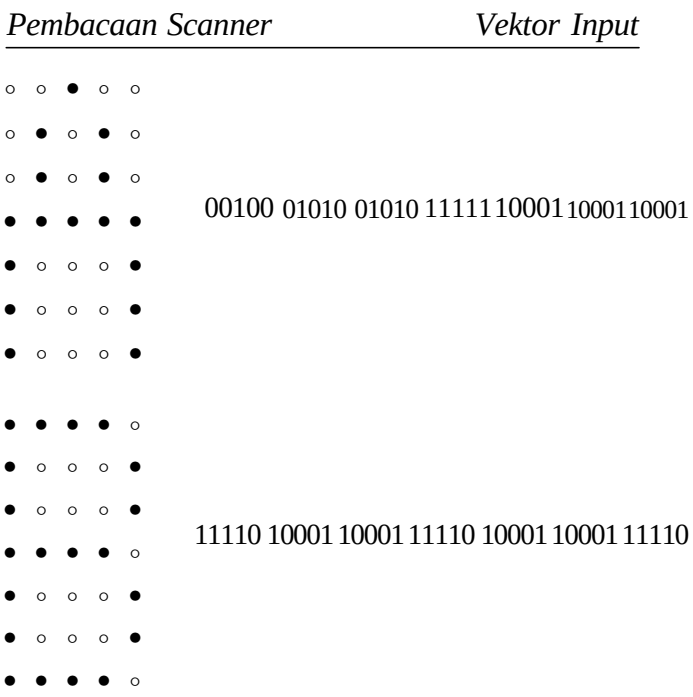
Jaringan propagasi balik tersebut direncanakan hanya akan menerima input biner sehingga data harus diatur sebagai sekumpulan angka (vektor). Elemn-elemen yang dikelompokkan bersama ke dalam sebuah vektor ini menggambarkan feature data dan karena disusun bersama-sama maka akan mempermudah jaringan untuk mengenali feature tersebut.

Tingkat Pelatihan

Rancangan jumlah node pada jaringan syaraf tiruan untuk pengenalan karakter alfanumerik:

1. Lapisan input sebanyak 35 node
2. Lapisan tersembunyi sebanyak 35 node
3. Lapisan output sebanyak 6 node.

Penentuan 35 node pada lapisan input didasarkan pada pembacaan yang dilakukan oleh scanner. Sebagaimana telah dijelaskan sebelumnya, karakter alfanumerik yang dibaca oleh scanner diterjemahkan ke dalam bentuk matriks berukuran 5 x 7 (atau = 35) yang melambangkan pixel-pixel data ini dimasukkan ke dalam jaringan syaraf tiruan dalam bentuk biner. Angka 1 menunjukkan pixel berisi citra, angka 0 menunjukkan kosong. Untuk lebih jelasnya, lihat gambar 6.3.



Gambar 7.3 Citra karakter A dan B yang dipetakan sebagai input jaringan

Mengenai jumlah node untuk lapisan output adalah sebanyak 6 node dengan pertimbangan bahwa pelambangan 36 karakter alfanumerik dalam bentuk biner akan menghabiskan tempat sebanyak 6 digit.

$$36_{10} = 100100_{2}$$
$$6\ digit$$

Pada pengolahan data output, ketigapuluhenam karakter alfanumerik tersebut diurutkan sesuai dengan urutan binernya. Sebagai contoh dapat dilihat pada Tabel 7.2

Tabel 7.2 Output dan karakter yang diwakilinya

Output	Karakter
000001	A
000010	B
000011	C
000100	D
dst.	dst.

sementara dari Tabel 7.1 (kemampuan jaringan syaraf tiruan) dapat dilihat bahwa jaringan syaraf tiruan yang menggunakan metode propagasi balik memiliki waktu pelatihan yang lambat. Dengan mengingat bahwa tujuan penerapan jaringan propagasi balik adalah untuk mendapatkan kesetimbangan antara respons yang benar terhadap pola pelatihan (kemampuan memorisasi) dan respons yang baik terhadap pola-pola input baru (kemampuan menggeneralisasi) maka tidak terlalu penting untuk melanjutkan pelatihan sampai total error kuadratnya mencapai nilai minimum. Oleh sebab itu nilai error yang dapat diterima oleh jaringan (nilai toleransi error atau disebut juga dengan istilah "limit") ditentukan terlebih dahulu dan nilai ini digunakan untuk menghentikan pelatihan bila error jaringan lebih kecil atau sama dengan nilai limit yang telah ditentukan tersebut. Pada aplikasi untuk pengenalan karakter alfanumerik ini nilai limit yang direncanakan akan digunakan adalah sebesar 0,002. Nilai limit sengaja dipilih ang bernilai kecil untuk tetap menjaga keakuratan hasil yang diperoleh.

7.2.2 Pengumpulan Data Pelatihan

Langkah-langkah pengumpulan data untuk membangun aplikasi pengenal karakter alfanumerik adalah sebagai berikut:

a. Membuat rancangan data (input dan output) yang akan digunakan sebagai latihan dan testing.

   Karena di dalam dunia nyata citra karakter yang dibaca sangat mungkin mengandung derau (*noise*) atau tidak lengkap,

maka jaringan propagasi balik yang dirancang harus dilatih dengan banyak contoh yang juga mengandung noise atau tidak lengkap. Diharapkan dengan contoh-contoh tersebut maka jaringan dapat menggeneralisasinya sehigga bila dimasa depan dimasukkan input baru yang berbeda yang citranya juga mengandung error maka jaringan tetap mampu mengenalinya sebagai sebuah karakter tertentu.

Pada penelitian ini belum keseluruhan karakter alfanumerik yang dicobakan, hanya terbatas pada karakter 'A', 'B', dan 'C' saja, namun telah dipikirkan pengembangan jaringan ke arah itu (sengja disediakan tempat untuk node output sebanyak 6 node untuk kemungkinan pengembangan).

b. Memisahkan data ke dalam 2 bagian, yaitu set pelatihan (set training) dan set testing

Citra-citra karakter yang digunakan untuk pelatihan dan testing diubah ke dalam bentuk biner. Citra-citra tersebut mempunyai ketidaklengkapan citra sampai sebanyak 2 pixel dengan pertimbangan jika lebih dari itu citra mungkin sangat susah dikenali (lihat Tabel 7.3).

Tabel 7.3 Set pelatihan (set training) yang digunakan untuk melatih jaringan

Vektor Input (35 digit)	**Vektor Output**	**Target**
12345 67890 67890 12345 67890 12345	123456	Karakter
00100 01010 01010 11111 10001 10001 10001	000001	A
11110 10001 10001 11110 10001 10001 11110	000010	B
01110 10001 10000 10000 10000 10001 01110	000011	C
00000 01010 01010 11111 10001 10001 10001	000001	A
01110 10001 10001 11110 10001 10001 11110	000010	B
00110 10001 10000 10000 10000 10001 01110	000011	C
00100 00010 01010 11111 10001 10001 10001	000001	A
11010 10001 10001 11110 10001 10001 11110	000010	B
01010 10001 10000 10000 10000 10001 01110	000011	C

00100 01000 01010 11111 10001 10001 10001	000001	A
11010 10001 10001 11110 10001 10001 11110	000010	B
01100 10001 10000 10000 10000 10001 01110	000011	C
00100 01010 00010 11111 10001 10001 10001	000001	A
11100 10001 10001 11110 10001 10001 11110	000010	B
01110 00001 10000 10000 10000 10001 01110	000011	C
00100 01010 01000 11111 10001 10001 10001	000001	A
11110 00001 10001 11110 10001 10001 11110	000010	B
01110 10000 10000 10000 10000 10001 01110	000011	C
00100 01010 01010 01111 10001 10001 10001	000001	A
11110 10000 10001 11110 10001 10001 11110	000010	B
01110 10001 00000 10000 10000 10001 01110	000011	C
00100 01010 01010 10111 10001 10001 10001	000001	A
11110 10001 00001 11110 10001 10001 11110	000010	B
01110 10001 10000 00000 10000 10001 01110	000011	C
00100 01010 01010 11011 10001 10001 10001	000001	A
11110 10001 10000 11110 10001 10001 11110	000010	B
01110 10001 10000 10000 00000 10001 01110	000011	C
00100 01010 01010 11101 10001 10001 10001	000001	A
11110 10001 10001 01110 10001 10001 11110	000010	B
01110 10001 10000 10000 10000 00001 01110	000011	C
00100 01010 01010 11110 10001 10001 10001	000001	A
11110 10001 10001 10110 10001 10001 11110	000010	B
01110 10001 10000 10000 10000 10000 01110	000011	C
00100 01010 01010 11111 00001 10001 10001	000001	A
11110 10001 10001 11010 10001 10001 11110	000010	B
01110 10001 10000 10000 10000 10001 00110	000011	C
00100 01010 01010 11111 10000 10001 10001	000001	A
11110 10001 10001 11100 10001 10001 11110	000010	B
01110 10001 10000 10000 10000 10001 01010	000011	C
00100 01010 01010 11111 10001 00001 10001	000001	A
11110 10001 10001 11110 00001 10001 11110	000010	B
01110 10001 10000 10000 10000 10001 01100	000011	C
00100 01010 01010 11111 10001 10000 10001	000001	A
11110 10001 10001 11110 10000 10001 11110	000010	B
00010 10001 10000 10000 10000 10001 01110	000011	C

00100 01010 01010 11111 10001 10001 00001 11110 10001 10001 11110 10001 00001 11110 00110 00001 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00100 01010 01010 11111 10001 10001 00000 11110 10001 10001 11110 10001 10000 11110 00110 10000 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 00010 01010 11111 10001 10001 10001 11110 10001 10001 11110 10001 10001 01110 00110 10001 00000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 01000 01010 11111 10001 10001 10001 11110 10001 10001 11110 10001 10001 10110 00110 10001 10000 00000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 00010 11111 10001 10001 10001 11110 10001 10001 11110 10001 10001 11010 00110 10001 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 01000 11111 10001 10001 10001 11110 10001 10001 11110 10001 10001 11100 00110 10001 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 01010 01111 10001 10001 10001 00110 10001 10001 11110 10001 10001 01110 00110 10001 10000 10000 10000 10000 01110	000001 000010 000011	A B C
00000 01010 01010 10111 10001 10001 10001 01010 10001 10001 11110 10001 10001 01110 00110 10001 10000 10000 10000 10001 00110	000001 000010 000011	A B C
00000 01010 01010 11011 10001 10001 10001 00110 10001 10001 11110 10001 10001 01110 00110 10001 10000 10000 10000 10001 01010	000001 000010 000011	A B C
00000 01010 01010 11101 10001 10001 10001 01110 00001 10001 11110 10001 10001 11110 00110 10001 10000 10000 10000 10001 01100	000001 000010 000011	A B C
00000 01010 01010 11110 10001 10001 10001 01110 10001 10001 11110 10001 10001 11110 01000 10001 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 01010 11111 00001 10001 10001 01110 10001 00001 11110 10001 10001 11110 01010 00001 10000 10000 10000 10001 01110	000001 000010 000011	A B C

00000 01010 01010 11111 10000 10001 10001 01110 10001 10000 11110 10001 10001 11110 01010 10000 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 01010 11111 10001 00001 10001 01110 10001 10001 01110 10001 10001 11110 01010 10001 00000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 01010 11111 10001 10001 11110 01110 10001 10001 10110 10001 10001 11110 01010 10001 10000 00000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 01010 11111 10001 10001 00001 01110 10001 10001 11010 10001 10001 11110 01010 10001 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00000 01010 01010 11111 10001 10001 10000 01110 10001 10001 11100 10001 10001 11110 01010 10001 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00100 00000 01010 11111 10001 10001 10001 01110 10001 10001 11110 00001 10001 11110 01010 10001 10000 10000 10000 10000 01110	000001 000010 000011	A B C
00100 00010 00010 11111 10001 10001 10001 01110 10001 10001 11101 10000 10001 11110 01010 10001 10000 10000 10000 10001 00110	000001 000010 000011	A B
00100 00010 01000 11111 10001 10001 10001 01110 10001 10001 11110 10001 00001 11110 01010 10001 10000 10000 10000 10001 01010	000001 000010 000011	A B C
00100 00010 01010 01111 10001 10001 10001 01110 10001 10001 11110 10001 10000 11110 01010 10001 10000 10000 10000 10001 01100	000001 000010 000011	A B C
00100 00010 01010 10111 10001 10001 10001 01110 10001 10001 11110 10001 10001 01110 01100 00001 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00100 00010 01010 11011 10001 10001 10001 01110 10001 10001 11110 10001 10001 10110 01100 10000 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00100 00010 01010 11101 10001 10001 10001 11110 10001 10001 11110 10001 10001 11010 01100 10001 00000 10000 10000 10001 01110	000001 000010 000011	A B C

00100 00010 01010 11110 10001 10001 10001	000001	A
01110 10001 10001 11110 10001 10001 11100	000010	B
01100 10001 10000 00000 10000 10001 01110	000011	C
00100 00010 01010 11111 00001 10001 10001	000001	A
10010 10001 10001 11110 10001 10001 11110	000010	B
01100 10001 10000 10000 10000 10001 01110	000011	C
00100 00010 01010 11111 10000 10001 10001	000001	A
10100 10001 10001 11110 10001 10001 11110	000010	B
01100 10001 10000 10000 10000 10001 01110	000011	C
00100 00010 01010 11111 10001 00001 10001	000001	A
10110 00001 10001 11110 10001 10001 11110	000010	B
01100 10001 10000 10000 10000 10000 01110	000011	C
00100 00010 01010 11111 10001 10000 10001	000001	A
10110 10000 10001 11110 10001 10001 11110	000010	B
01100 10001 10000 10000 10000 10001 00110	000011	B
00100 00010 01010 11111 10001 10001 10001	000001	A
10110 10001 00001 11110 10001 10001 11110	000010	B
01100 10001 10000 10000 10000 10001 01010	000011	C
00100 01000 00010 11111 10001 10001 10001	000001	A
10110 10001 10000 11110 10001 10001 11110	000010	B
01100 10001 10000 10000 10000 10001 01100	000011	C
00100 01000 00010 11111 10001 10001 10001	000001	A
10110 10001 10001 01110 10001 10001 11110	000010	B
01110 00000 10000 10000 10000 10001 01110	000011	C
00100 01000 01000 11111 10001 10001 10001	000001	A
10110 10001 10001 10110 10001 10001 11110	000010	B
01110 00001 00000 10000 10000 10001 01110	000011	C
00100 01000 01010 01111 10001 10001 10001	000001	A
10110 10001 10001 11010 10001 10001 11110	000010	B
01110 00001 10000 00000 10000 10001 01110	000011	C
00100 01000 01010 10111 10001 10001 10001	000001	A
10110 10001 10001 11100 10001 10001 11110	000010	B
01110 00001 10000 10000 00000 10001 01110	000011	C
00100 01000 01010 11011 10001 10001 10001	000001	A
10110 10001 10001 11110 00001 10001 11110	000010	B
01110 00001 10000 10000 10000 00001 01110	000011	C

00100 01000 01010 11101 10001 10001 10001 10110 10001 10001 11110 10000 10001 11110 01110 00001 10000 10000 10000 10000 01110	000001 000010 000011	A B C
00100 01000 01010 11110 10001 10001 10001 10110 10001 10001 11110 10001 10001 11110 01110 00001 10000 10000 10000 10001 00110	000001 000010 000011	A B C
00100 01000 01010 11111 00001 10001 10001 10110 10001 10001 11110 10001 10000 11110 01110 00001 10000 10000 10000 10001 01010	000001 000010 000011	A B C
00100 01000 01010 11111 10000 10001 10001 10110 10001 10001 11110 10001 10001 01110 01110 00001 10000 10000 10000 10001 01100	000001 000010 000011	A B C
00100 01000 01010 11111 10001 00001 10001 10110 10001 10001 11110 10001 10001 10110 01110 10000 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00100 01000 01010 11111 10001 10000 10001 10110 10001 10001 11110 10001 10001 11010 01110 10000 10000 00000 10000 10001 01110	000001 000010 000011	A B C
00100 01000 01010 11111 10001 10001 00001 10110 10001 10001 11110 10001 10001 11100 01110 10000 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00100 01000 01010 11111 10001 10001 10000 11000 10001 10001 11110 10001 10001 11110 01110 10000 10000 10000 10000 10001 01110	000001 000010 000011	A B C
00100 01010 00000 11111 10001 10001 10001 11010 00001 10001 11101 10001 10001 11110 01110 10000 10000 10000 10000 10000 01110	000001 000010 000011	A B C
00100 01010 00010 11111 10001 10001 10001 11010 10000 10001 11110 10001 10001 11110 01110 10000 10000 10000 10000 10001 00110	000001 000010 000011	A B C

Sementara untuk set testing yang digunakan untuk menguji kemampuan generalisasi jaringan dapat dilihat pada Tabel 7.4.

Vektor Input (35 digit)	**Vektor Output**	**Target**
12345 67890 67890 12345 67890 12345	123456	Karakter
00100 01010 00010 10111 10001 10001 10001	000001	A
11010 10001 00001 11110 10001 10001 11110	000010	B
01110 10000 10000 10000 10000 10001 01010	000011	C
00100 01010 00010 11011 10001 10001 10001	000001	A
11010 10001 10000 11110 10001 10001 11110	000010	B
01110 10000 10000 10000 10000 10001 01100	000011	C
00100 01010 00010 11011 10001 10001 10001	000001	A
11010 10001 10001 01110 10001 10001 11110	000010	B
01110 10001 00000 00000 10000 10001 01110	000011	C
00100 01010 00010 11110 10001 10001 10001	000001	A
11010 10001 10001 10110 10001 10001 11110	000010	B
01110 10001 00000 10000 00000 10001 01110	000011	C
00100 01010 00010 11111 00001 10001 10001	000001	A
11010 10001 10001 11010 10001 10001 11110	000010	B
01110 10001 00000 10000 10000 10001 01110	000011	C
00100 01010 00010 11111 10000 10001 10001	000001	A
11010 10001 10001 11100 10001 10001 11110	000010	B
01110 10001 00000 10000 10000 10000 01110	000011	C
00100 01010 00010 11111 10001 00001 10001	000001	A
11010 10001 10001 11110 00001 10001 11110	000010	B
01110 10001 00000 10000 10000 10001 00110	000011	C
00100 01010 00010 11111 10001 10000 10001	000001	A
11010 10001 10001 11110 10000 10001 11110	000010	B
01110 10001 00000 10000 10000 10001 01010	000011	C
00100 01010 00010 11111 10001 10001 00001	000001	A
11010 10001 10001 11110 10001 00001 11110	000010	B
01110 10001 00000 10000 10000 10001 01100	000011	C

00100 01010 00010 11111 10001 10001 10000 11010 10001 10001 11110 10001 10000 11110 01110 10001 10000 00000 00000 10001 01110	000001 000010 000011	A B C
00100 01010 01000 01111 10001 10001 10001 11010 01010 01001 11110 10001 10001 01110 01110 10001 10000 00000 10000 00001 01110	000001 000010 000011	A B C
00100 01010 01000 10111 10001 10001 10001 11010 10001 10001 10001 11110 10001 10110 01110 10001 10000 00000 10000 10000 01110	000001 000010 000011	A B C
00100 01010 01000 11011 10001 10001 10001 11010 10001 10001 11110 10001 10001 11010 01110 10001 10000 00000 10000 10001 00110	000001 000010 000011	A B C
00100 01010 01000 11101 10001 10001 10001 11010 10001 10001 11110 10001 10001 11100 01110 10001 10000 00000 10000 10001 01010	000001 000010 000011	A B C
00100 01010 01000 11110 10001 10001 10001 11100 00001 10001 11110 10001 10001 11110 01110 10001 10000 00000 10000 10001 01100	000001 000010 000011	A B C
00100 01010 01000 11111 00001 10001 10001 11100 10000 10001 11111 10001 10001 11110 01110 10001 10000 10000 00000 00001 01110	000001 000010 000011	A B C
00100 01010 01000 11111 10000 10001 10001 11100 10001 00001 11110 10001 10001 11110 01110 10001 10000 00000 00000 10000 01110	000001 000010 000011	A B C
00100 01010 01000 11111 10001 00001 10001 11100 10001 10000 11110 10001 10001 11110 01110 10001 10000 10000 00000 10001 00110	000001 000010 000011	A B C
00100 01010 01000 11111 10001 10000 10001 11100 10001 10001 01110 10001 10001 11110 01110 10001 10000 10000 00000 10001 01010	000001 000010 000011	A B C
00100 01010 01000 11111 10001 10001 00001 11100 10001 10001 10110 10001 10001 11110 01110 10001 10000 10000 00000 10001 01100	000001 000010 000011	A B C
00100 01010 01000 11111 10001 10001 10000 11100 10001 10001 11010 10001 10001 11110 01110 10001 10000 10000 10000 00000 01110	000001 000010 000011	A B C

00100 01010 01010 00111 10001 10001 10001	000001	A
11100 10001 10001 11100 10001 10001 11110	000010	B
01110 10001 10000 10000 10000 10001 00110	000011	C
00100 01010 01010 01011 10001 10001 10001	000001	A
11100 10001 10001 11110 00001 10001 11110	000010	B
01110 10001 10000 10000 10000 00001 01010	000011	C
00100 01010 01010 01101 10001 10001 10001	000001	A
11100 10001 10001 11110 10000 10001 11110	000010	B
01110 10001 10000 10000 10000 00001 01100	000011	C
00100 01010 01010 01110 10001 10001 10001	000001	A
11100 10001 10001 11110 10001 00001 11110	000010	B
01110 10001 10000 10000 10000 10000 00110	000011	C
00100 01010 01010 01111 00001 10001 10001	000001	A
11100 10001 10001 11110 10001 10000 11110	000010	B
01110 10001 10000 10000 10000 10000 01010	000011	C
00100 01010 01010 01111 10000 10001 10001	000001	A
11100 10001 10001 11110 10001 10001 01110	000010	B
01110 10001 10000 10000 10000 10000 01100	000011	C
00100 01010 01010 01111 10001 00001 10001	000001	A
11100 10001 10001 11110 10001 10001 10110	000010	B
01110 10001 10000 10000 10000 10001 00010	000011	C
00100 01010 01010 01111 10001 10000 10001	000001	A
11100 10001 10001 11110 10001 10001 11010	000010	B
01110 10001 10000 10000 10000 10001 00100	000011	C
00100 01010 01010 01111 10001 10001 00001	000001	A
11100 10001 10001 11110 10001 10001 11100	000010	B
01110 10001 10000 10000 10000 10001 01000	000011	C

Pada pembelajaran set pelatihan, karakter-karakter yang berbeda disajikan ke dalam jaringan secara berselang-seling (misalnya A, lalu B, lalu C, kemudian kembali ke A, dan seterusnya). Jika batas error yang dapat diterima belum dicapai maka pelatihan akan terus berlangsung (sangat mungkin terjadi banyak epoch), dan jaringan akan dilatih mulai dari contoh kasus pelatihan yang pertama lagi, begitu seterusnya sampai kondisi berhenti terpenuhi.

c. Mengonfirmasikan keandalan

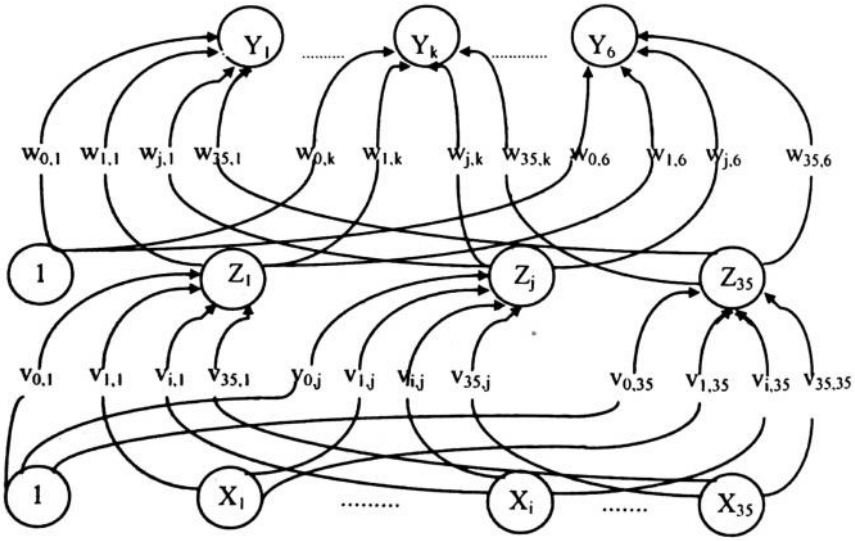
Jumlah set pelatihan adalah sebanyak 183 contoh kasus, sedangkan set testingnya sebanyak 90 kasus. Jumlah set pelatihan (183 contoh kasus) dianggap cukup mampu melatih jaringan untuk dapat mengenali karakter-karakter yang telah dipelajarinya.

4.2.3 Pemilihan Lingkungan Pengembangan

Berikut ini diuraikan langkah-langkah pembangunan simulator jaringan propagasi balik untuk pengenalan karakter alfanumerik.

Arsitektur jaringan

Pembangunan simulator jaringan syaraf tiruan dimulai dari perancangan arsitektur jaringan seperti terlihat pada Gambar 7.4.



Gambar 7.4 Rancangan arsitektur jaringan syaraf propagasi balik untuk pengenalan alfanumerik

Dari gambar 7.4 terlihat:
1). Unit input = 3 6 node
2). Unit tersembunyi = 36 node → HiddenSize = 36
3). Unit output = 6 node → OutSize = 6

Hal lain yang perlu diperhatikan di dalam perancangna jaringan propagasi balik untuk pengenalan karakter alfanumerik ini adalah:

1). Ambang pembelajaran (limit) = 0,002
2). Angka pembelajaran (alpha) = 0,2
3) jumlah contoh kasus pada set pelatihan (MaximumX) = 183

Sementara untuk bobot-bobot yang terlibat adalah:

a. Bobot-bobot antara lapisan input dengan lapisan tersembunyi (v_{ij})
   v_{ij} = 36 x 35 = 1260 buah
b. Bobot-bobot antara lapisan tersembunyi dengan lapisan output (w_{jk})
   W_{jk} = 36 x 6 = 216 buah.

Struktur Data

Berdasarkan pemahaman tentang cara jaringan propagasi balik beroperasi, maka struktur top-level dari simulator jaringan propagasi balik (BPN = BackPropogation Network) dapat didefinisikan sebagai berikut:

```
    BPN = record
  inp_layer : TLayer_In   ;  { lapisan input }
  hid_layer : TLayer_Hid  ;  { lapisan tersembunyi}
  out_layer : TLayer_Out  ;  { lapisan output )
end;
```

struktur lapisan input didefinisikan sebagai:

```
    TLayer_In = record
          input : Pdouble36;
          bobot : Pdouble1260;
    end;
```

untuk lapisan tersembunyi, strukturnya adalah:

```
    TLayer_Hid = record
```

```
        input : Pdouble36;
        bobot : Pdouble1260;
        output : Pdouble35;
        delta_k : Pdouble6;
        delta_j : Pdouble35;
    end;
```

sedangkan untuk lapisan output, strukturnya adalah:

```
    TLayer_Out = record
        input : Pdouble35;
        output : Pdouble6;
        bobot : Pdouble216;
        target : Pdouble6;
        delta_k : Pdouble6;
        delta_wjk : Pdouble216;
    end;
```

dengan

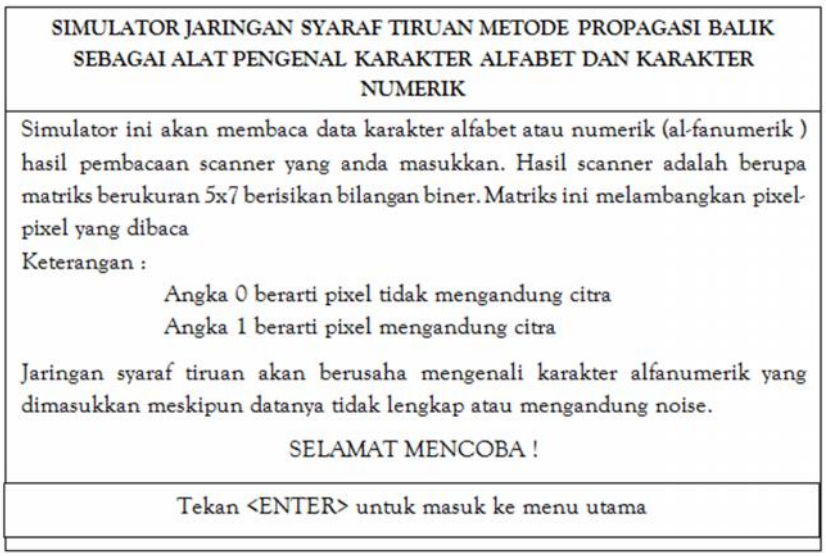
```
    Pdouble6 = ^double6;
    Pdouble35 = ^double35;
    Pdouble36 = ^double36;
    Pdouble216 = ^double216;
    Pdouble1260 = ^double1260;
```

di mana

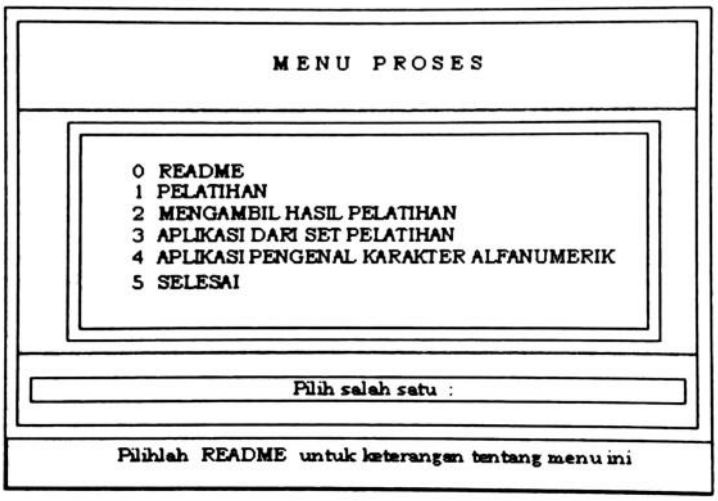
```
    double6 = array[1...6] of double;
    double35 = array[1...35] of double;
    double36 = array[0...35] of double;
    double216 = array[0...35, 1...6] f double;
    double1260 = array[0...35, 1...35] of double;
```

Rancangan Simulator Jaringan Propagasi Balik

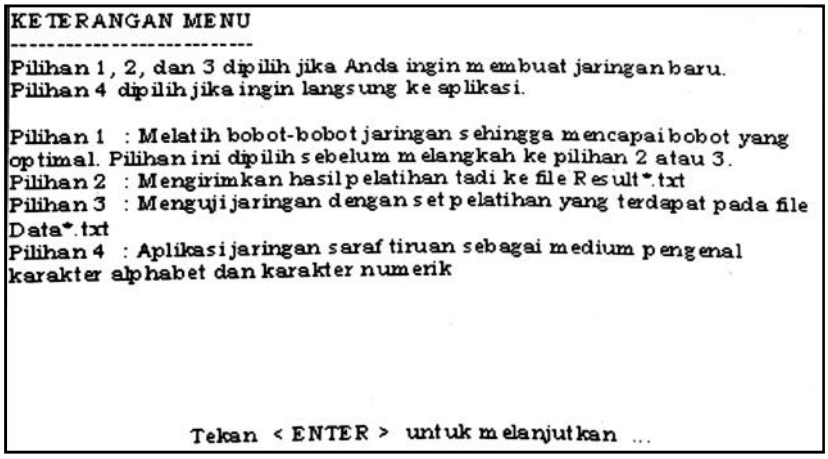
Rancangan tampilan layar simulator sebagai berikut



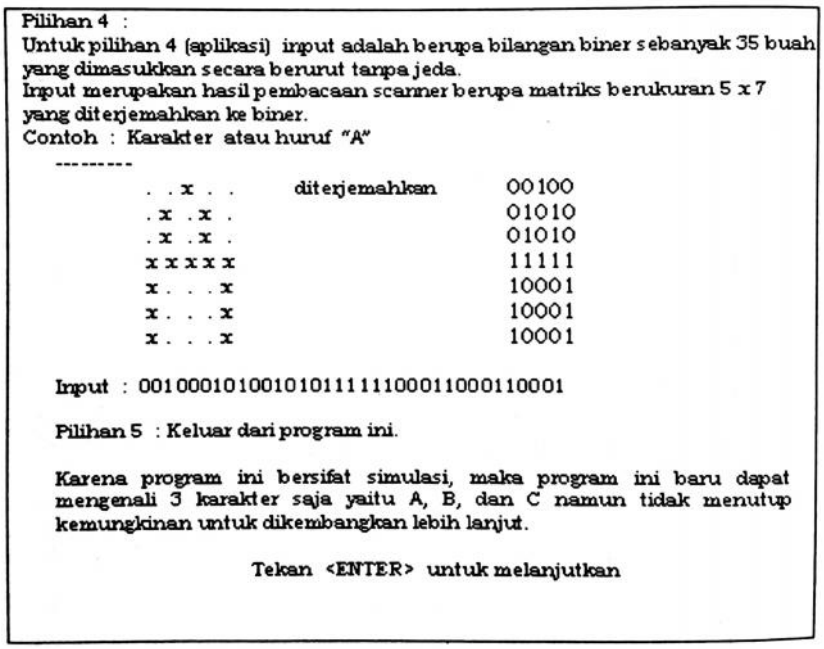
Gambar 7.5 Tampilan pembuka



Gambar 7.6 Tampilan menu



Gambar 7.7 **Panduan menu**



Gambar 6.8 Panduan menu (lanjutan)

Enam pilihan menu lainnya yang ditawarkan oleh simulator ini adalah:

0. Readme
   Berisi panduan menu, semacam *help*.
1. Pelatihan
   Digunakan untuk melatih jaringan, menggunakan algoritma pelatihan propagasi balik (lihat bagian B pada Bab 5).
2. Menggunakan hasil pelatihan
   Bobot-bobot hasil pelatihan dikirim ke sebuah file (result sampai dengan result4) sehingga bisa dilihat bobot final hasil pelatihannya.
3. Aplikasi dari set pelatihan
   Jaringan propagasi balik diuji kemampuan memorisasinya dengan pilihan ini. Sebagai kasusnya adalah isi dari set pelatihan pelatihan (semua kasus telah dipelajari sebelumnya oleh jaringan). Hasil pengujian dari set training ini berbentuk file teks.
4. Aplikasi pengenal karakter alfanumerik
   Kemampuan generalisasi jaringan diuji dengan pilihan ini. Untuk menguji jaringan digunakan kasus-kasus yang belum pernah dipelajari oleh jaringan. Kasus-kasus ini terdapat pada set tes. Sebelum mengambil pilihan ini harus mengambil 2 dulu untuk pengetesan bobot awal pada rutin untuk aplikasi pengenal alfanumerik (yaitu dengan meniginisialisasinya dengan bobot final yang diperoleh selama pelatihan). Pilihan ini juga digunakan untuk menguji kemampuan jaringan syaraf tiruan untuk mengenali karakter alfanumerik dengan input sembarang.
5. Keluar
   Untuk keluar dari program.

7.3 Tahap implementasi

7.3.1 Implementasi dan Pelatihan

Tujuan dari tahap ini adalah untuk menghasilkan sebuah sistem berjalan yang akurat dan konsisten serta mampu mengeksploitasi keunggulan-keunggulan jaringan syaraf tiruan. Guna mencapai hal ini perlu dipahami proses-proses yang terjadi di dalam jaringan propagasi balik.

Perlu diperhatikan bahwa:

i. n adalah pencacah unit-unit dengan nilai tertinggi n = 35
ii. p adalah pencacah unit-unit tersembunyi dengan nilai tertinggi p = 35
iii. m adalah pencacah unit-unit output dengan nilai tertinggi m = 6.

Proses-proses yang terjadi di dalam sebuah jaringan propagasi balik:

a) Inisialisasi Bobot
   Seluruh bobot di set ke dalam bilangan acak yang kecil.
   1. Seluruh bobot antara lapisan input dengan lapisan tersembunyi disebut sebagai v_{ij} sejumlah 1260 buah dengan I=1,..., n dan j=1,...,p.
   2. Bobot-bobot bias antara lapisan input dengan lapisan tersembunyi disebut v_{ij} sejumlah 35 buah dengan j=1,...,p.
   3. Bobot-bobot antara lapisan tersembunyi dengan lapisan output disebut sebagai w_{jk} sejumlah 216 buah dengan j=1,...,p dan k=1,...,m.
   4. Bobot-bobot bias antara lapisan tersembunyi dengan lapisan output disebut v_{ok} sejumlah 6 buah dengan k=1,...,m.

Untuk menghindari minimum lokal keseluruhan bobot di atas diinisialisasi dengan menggunakan bilangan acak antara -0,5 dan 0,5. Bobot-bobot inisial yang digunakan untuk mengeset bobot awal pelatihan selengkapnya adalah sebagai Tabel berikut 7.5 berikut.

Tabel 7.5 Bobot-bobot inisial antara lapisan input dan lapisan tersembunyi

v / i	[i,01]	[i,02]	[i,03]	[i,04]	[i,05]	[i,06]	[i,07]	[i,08]	[i,09]	[i,10]
00	-0.123	-0.345	+0.432	-0.231	+0.167	+0.176	+0.178	+0.25	+0.261	+0.162
01	-0.111	-0.242	+0.153	+0.488	+0.343	-0.476	+0.19	-0.222	+0.3	+0.177
02	+0.243	+0.368	+0.171	-0.196	+0.441	+0.118	+0.399	-0.198	-0.377	+0.494
03	+0.154	+0.244	-0.1	+0.121	+0.196	+0.492	+0.235	+0.403	-0.193	-0.239
04	+0.489	-0.187	-0.482	+0.493	+0.4	+0.196	-0.378	+0.238	+0.401	-0.202
05	+0.344	+0.223	+0.263	-0.197	+0.438	+0.121	-0.194	-0.379	+0.236	+0.194
06	+0.477	+0.117	-0.357	+0.122	+0.379	-0.12	+0.401	+0.197	-0.379	+0.124
07	+0.191	+0.425	-0.442	+0.137	-0.119	+0.494	+0.237	+0.407	-0.195	-0.317
08	-0.223	-0.183	-0.173	+0.495	+0.232	+0.198	+0.38	+0.241	+0..403	+0.202
09	+0.304	-0.309	+0.301	+0.201	-0.316	+0.123	+0.196	+0.201	-0.238	+0.302
10	+0.178	-0.214	-0.082	+0.126	+0.192	+0.134	+0.406	+0.499	+0.383	-0.31
11	+0.09	+0.091	+0.215	+0.324	-0.491	0.498	+0.24	+0.203	+0.197	+0.302
12	+0.435	-0.307	-0.309	+0.193	+0.341	+0.119	-0.2	-0.399	+0.493	+0.234
13	+0.181	+0.171	-0.187	+0.198	+0.127	+0.25	+0.492	-0.235	-0.128	+0.496
14	+0.223	-0.119	+0.425	+0.294	+0.249	+0.317	+0.38	+0.378	-0.203	-0.174
15	-0.334	+0.482	+0.117	+0.203	+0.323	+0.401	-0.242	+0.194	+0.499	+0.184
16	+0.062	-0.263	-0.223	+0.495	+0.119	+0.196	+0.494	+0.402	-0.201	+0.224
17	+0.3	+0.367	-0.334	+0.243	+0.499	+0.123	-0.202	+0.237	+0.241	-0.121
18	+0.172	+0.243	+0.223	-0.201	+0.492	+0.325	+2.293	-0.378	+0.407	+0.4
19	-0.441	-0.186	+0.181	+0.407	+0.119	+0.378	-0.327	+0.196	+0.19	+0.123
20	+0.356	+0.062	+0.435	-0.197	+0.493	+0.194	-0.125	+0.406	+0.238	-0.358
21	+0.262	+0.222	-0.089	+0.194	+0.119	-0.118	+0.492	+0.196	+0.121	-0.12
22	+0.481	+0.116	-0.438	+0.291	-0.197	+0.126	+0.196	-0.293	+0.202	+0.494
23	-0.118	+0.424	+0.406	-0.196	+0.38	+0.237	+0.402	+0.194	-0.378	+0.235
24	+0.17	-0.182	+0.121	+0.2	+0.492	-0.119	+0.199	-0.124	+0.135	+0.489
25	-0.306	-0.308	+0.202	+0.498	-0.2	+0.24	+0.406	+0.196	-0.378	-0.237
26	+0.09	+0.218	+0.439	-0.318	+0.438	+0.317	-0.493	+0.2	+0.199	+0.127
27	+0.213	+0.081	-0.366	+0.207	+0.326	-0.186	+0.197	+0.495	+0.224	+0.224
28	-0.308	+0.3	+0.329	+0.198	+0.487	+0.216	+0.195	-0.206	+0.484	+0.213
29	-0.182	+0.172	+0.111	+0.32	+0.189	+0.377	+0.218	+0.317	-0.178	+0.166

30	+0.424	-0.441	+0.135	-0.253	+0.417	+0.337	-0.118	+0.206	+0.192	+0.217
31	+0.116	+0.356	-0.188	+0.319	-0.11	+0.426	+0.346	+0.127	+0.245	-0.225
32	+0.222	+0.262	+0.367	+0.423	-0.343	+0.124	+0.242	-0.344	-0.246	+0.214
33	-0.186	+0.481	-0.488	+0.167	+0.463	+0.099	-0.387	+0.357	+0.413	+0.323
34	+0.243	-0.118	+0.226	-0.497	+0.199	-0.188	+0.328	+0.209	+0.368	+0.424
35	+0.367	+0.17	+0.196	+0.185	-0.342	-0.122	+0.194	+0.121	-0.197	+0.233

(Lanjutan)

v i	[i,11]	[i,12]	[i,13]	[i,14]	[i,15]	[i,16]	[i,17]	[i,18]	[i,19]	[i,20]
00	-0.253	+0.199	-0.312	-0.411	+0.025	+0.17	+0.087	-0.432	-0.2	-0.31
01	-0.089	+0.434	+0.18	-0.222	-0.333	+0.061	-0.299	+0.171	-0.44	0.355
02	+0.198	-0.366	+0.239	+0.407	+0.201	+0.333	+0.435	+0.153	+0.186	-403
03	+0.21	-0.111	+0.144	+0.144	+0.431	+0.201	+0.062	-0.323	+0.173	+0.223
04	+0.302	+0.305	-0.414	-0.414	+0.12	+0.431	-0.356	+0.325	+0.41	-0.491
05	+0.31	-0.227	+0.303	+0.303	-0.203	+0.12	+0.236	+0.366	-0.246	+0.197
06	+0.202	+0.377	+0.402	+0.402	-0.238	+0.203	+0.345	+0.242	+0.198	-0.2
07	+0.347	+0.178	+0.136	+0.138	+0.12	+0.238	+0.181	+0.121	-0.184	+0.239
08	-0.124	+0.091	-0.238	-0.238	+0.408	+0.12	-0.229	+0.241	+0.488	+0.496
09	+0.197	+0.174	+0.197	+0.197	+0.493	-0.408	+0.239	+0.237	+0.194	+0.128
10	+0.202	-0.369	+0.404	+0.404	-0.194	+0.493	+0.123	-0.202	+0.242	+0.235
11	-0.239	-0.203	+0.132	+0.132	+0.415	+0.194	+0.19	+0.123	+0.401	+0.492
12	+0.303	+0.063	-0.224	-0.224	+0.496	+0.415	-0.196	+0.499	+0.323	-0.25
13	+0.239	+0.2	+0.491	+0.491	+0.403	+0.4%	+0.327	+0.242	-0.203	+0.127
14	+0.38	+0.302	-0.219	-0.219	+0.415	+0.403	+0.378	-0.194	+0.117	+0.198
15	-0.198	-0.246	+0.173	+0.173	+0.186	+0.415	+0.119	+0.243	+0.482	+0.183
16	+0.408	+0.188	+0.342	+0.342	+0.17	+0.4%	-0.407	+0.334	+0.334	+0.171
17	+0.242	+0.366	+0.322	+0.322	-0.153	-0.403	+0.181	+0.367	+0.415	-0.81
18	-0.202	+0.411	-0.362	-0.362	+0.142	+0.415	+0.186	+0.3	-0.219	+0.496
19	-0.239	+0.229	+0.345	+0.345	-0.236	+0.186	+0.441	-0.17	+0.318	+0.224
20	+0.198	+0.318	+0.399	+0.399	+0.317	+0.17	-0.142	+0.342	+0.302	+0.264
21	+0.494	-0.2	+0.125	+0.323	-0.192	+0.153	+0.362	+0.128	+0.38	+0.063
22	+0.242	-0.383	+0.493	+0.2	+0.24	+0.142	+0.482	-0.188	+0.174	-0.303
23	-0.399	+0.195	+0.123	+0.194	-0.196	+0.236	+0.411	+0.408	-0.203	+0.234
24	+0.12	+0.291	-0.318	+0.402	+0.197	-0.317	+0.202	+0.224	+0.378	+0.493

25	+0.402	+0.194	+0.378	+0.235	+0.399	+0.399	+0.4	+0.201	+0.38	+0.399
26	+0.198	-0.292	+0.439	-0.193	+0.324	+0.302	-0.407	+0.402	-0.317	+0.2
27	-0.126	+0.345	+0.425	+0.369	+0.21	+0.318	-0.378	-0.494	+0.249	+0.119
28	+0.233	+0.263	-0.253	+0.311	-0.1	+0.198	+0.293	-0.196	+0.294	-0.341
29	+0.104	+0.422	-0.089	+0.414	+0.324	+0.352	+0.325	+0.199	-0.425	+0.193
30	-0.187	-0.176	+0.316	-0.217	+0.376	-0.238	+0.492	+0.495	-0.119	+0.309
31	+0.496	+0.198	+0.187	+0.327	+0.208	+0.406	-0.201	-0.223	+0.307	+0.187
32	+0.485	+0287	-0.207	+0.196	+0.217	-0.125	+0.223	+0.263	-0.435	+0.489
33	-0.134	+0.252	+0.416	+0.336	-0.117	+0.194	+0.242	+0.062	+0.214	-0.123
34	-0.344	+0.125	+0.243	-0.223	-0.494	+0.493	-0.172	-0.414	+0.178	+0.122
35	+0.128	-0.493	+0.198	+0.383	-0.238	-0.197	+0.416	+0.135	+0.223	+0.119

(lanjutan)

v / i	[i,21]	[i,22]	[i,23]	[i,24]	[i,25]	[i,26]	[i,27]	[i,28]	[i,29]	[i,30]
00	+0.342	+0.155	+0.43	+0.019	+0.086	-0.143	+0.267	+0.136	+0.048	+0.366
01	+0.261	+0.48	-0.117	+0.169	-0.305	+0.08	+0.212	-0.307	-0.181	+0.423
02	+0.415	+0.493	+0.12	-0.203	+0.431	+0.333	-0.495	+0.235	+0.456	+0.138
03	-0.132	+0.197	+0.138	+0.303	+0.144	+0.111	+0.407	+0.194	+0.31	-0.379
04	+0.306	+0.122	+0.132	+0.121	+0.197	+0.242	+0.201	-0.11	+0.227	+0.124
05	+0.203	+0.174	-0.178	+0.227	-0.185	+0.153	+0.184	+0.446	+0.121	+0.202
06	+0.239	+0.197	+0.317	-0.31	+0.196	+0.488	+0.244	+0.145	-0.303	+0.378
07	+0.302	-0.312	+0.347	+0.194	+0.242	+0.343	+0.119	+0.431	+0.203	-0.415
08	-0.197	-0.238	+0.195	+0.236	+0.366	+0.476	-0.121	+0.489	+0.477	+0.402
09	+0.203	+0.201	-0.407	+0.379	+0.104	+0.19	+0.196	+0.187	-0.117	+0.238
10	+0.302	+0.196	+0.237	-0.194	+0.422	+0.222	+0.492	+0.482	+0.357	+0.475
11	+0.24	+0.123	+0.494	+0.121	-0.344	+0.3	+0.235	+0.493	+0.122	-0.191
12	+0.498	-0.316	+0.119	-0.438	+0.166	+0.177	-0.408	-0.399	+0.379	+0.425
13	+0.491	+0.201	+0.137	+0.197	+0.463	+0.089	+0.193	+0.196	-0.12	+0.442
14	-0.324	+0.301	-0.442	+0.263	+0.206	+0.434	+0.239	-0.378	+0.401	+0.137
15	-0.219	+0.309	+0.245	+0.223	+0.1	-0.18	+0.201	+0.238	+0.197	-0.119
16	+0.019	-0.304	+0.191	+0.344	+0.357	+0.222	-0.111	+0.401	+0.379	+0.494
17	+0.09	+0.408	+0.431	-0.12	+0.187	+0.333	+0.444	+0.202	+0.236	+0.237
18	+0.194	+0.238	+0.144	+0.414	-0.263	+0.243	+0.144	+0.302	+0.194	+0.407
19	+0.404	+0.495	+0.444	+0.131	+0.117	-0.368	+0.431	-0.305	+0.31	+0.195

20	+0.236	-0.091	+0.111	+0.305	+0.11	+0.171	+0.154	+0.131	+0.227	-0.317
21	-0.369	+0.124	-0.201	+0.302	+0.487	+0.196	+0.244	+0.414	-0.121	+0.347
22	+0.202	+0.202	+0.239	-0.202	+0.425	+0.441	+0.119	-0.12	+0.303	+0.178
23	+0.31	+0.402	+0.193	+0.401	-0.223	+0.118	-0.121	+0.489	-0.203	+0.132
24	-0.383	+0.241	+0.403	+0.238	+0.423	-0.399	+0.196	+0.187	+0.344	+0.138
25	+0.499	-0.38	+0.235	+0.378	+0.186	+0.198	-0.441	+0.482	+0.223	-0.12
26	+0.406	+0.198	+0.492	-0.196	+0.19	-0.377	+0.492	+0.493	+0.263	-0.223
27	-0.134	+0.232	+0.196	+0.4	+0.217	-0.494	+0.235	-0.407	+0.197	+0.183
28	+0.192	+0.495	-0.4	+0.493	-0.417	+0.198	+0.403	+0.19	-0.438	+0.173
29	+0.126	+0.173	-0.493	+0.482	+0.183	+0.366	-0.193	+0,373	+0.121	+0.495
30	+0.082	-0.183	+0.482	-0.187	+0.19	-0.239	+0.239	+0.233	+0.194	+0.232
31	+0.498	-0.198	+0.233	-0.498	+0.38	+0.499	-0.439	+0.119	-0.122	+0.191
32	+0.194	+0.24	-0.439	+0.493	-0.198	+0.24	+0.318	-0.438	+0.121	+0.133
33	-0.399	+0.121	+0.196	-0.324	+0.238	+0.121	-0.38	+0.197	+0.495	+0.382
34	+0.401	-0.378	+0.492	+0.237	+0.495	-0.378	+0.24	+0.494	+0.201	-0.441
35	-0.121	+0.29	-0.402	-0.199	+0.378	+0.197	+0.401	-0.498	+0.126	+0.405

(lanjutan)

v / i	[i,31]	[i,32]	[i,33]	[i,34]	[i,35]
00	+0.402	-0.35	+0.189	-0.375	+0.222
01	-0.115	+0.221	-0.185	+0.242	+0.366
02	+0.147	+0.09	+0.1%	-0.124	+0.207
03	0.198	-0.204	+0.494	+0.343	-0.326
04	+0.38	-0.301	+0.223	-0.423	+0.186
05	-0.241	+0.276	-0.243	+0.367	+0.197
06	+0.403	+0.231	+0.125	+0.208	-0.495
07	+0.202	-0.38	+0.344	+0.327	+0.224
08	+0.124	+0.471	-0.424	+0.187	+0.244
09	+0.098	-0.302	+0.368	-0.198	+0.126
10	-0.485	+0.387	-0.209	+0.4%	+0.345
11	+0.228	+0.189	+0.328	+0.225	-0.425
12	+0.408	+0.371	+0.188	-0.245	+0.369
13	+0.304	-0.366	+0.199	+0.127	-0.21
14	+0.209	+0.402	+0.497	-0.346	+0.329

15	+0.301	+0.17	-0.226	+0.426	+0.198
16	+0.201	+0.432	+0.117	+0.11	+0.487
17	-0.301	+0.25	+0.336	+0.319	+0.216
18	+0.134	-0.199	+0.416	+0.188	-0.195
19	+0.169	+0.167	+0.252	-0.376	+0.206
20	+0.083	+0.375	-0.134	+0.217	+0.484
21	+0.142	-0.433	+0.323	+0.316	-0.213
22	+0.304	+0.19	+0.413	-0.176	+0.233
23	-0.259	+0.266	+0.357	+0.187	+0.263
24	+0.22	-0.164	+0.387	+0.217	-0.253
25	+0.116	+0.407	-0.099	+0.192	+0.311
26	+0.376	+0.394	+0.463	+0.206	+0.1
27	+0.138	-0.491	+0.1667	+0.118	-0.111
28	+0.254	+0.251	+0.488	-0.337	+0.32
29	-0.362	+0.088	+0.217	+0.417	+0.189
30	+0.026	-0.187	-0.196	+0.253	+0.377
31	+0.2	+0.235	+0.207	-0.135	+0.218
32	+0.118	-0.121	+0.485	+0.324	-0.317
33	+0.493	-0.399	+0.214	+0.414	+0.178
34	+0.194	+0.195	-0.246	+0.086	+0.166
35	-0.318	+0.4	+0.344	-0.422	+0.104

w / j	[j,1]	[j,2]	[j,3]	[j,4]	[j,5]	[j,6]
i=0	0.324	-0.492	0.12	-0.201	0.194	-0.497
i=1	0.44	-0.117	0.376	0.436	0.398	0.194
i=2	-0.195	0.491	0.192	0.316	-0.234	0.119
i=3	0.399	0.195	-0.4	0.437	0.377	-0.118
i=4	0.437	0.12	0.235	-0.317	0.193	0.492
i=5	0.378	0.119	-0.378	0.438	0.401	0.196
i=6	0.118	0.493	0.194	0.318	-0.236	0.121
i=7	0.231	0.197	-0.402	-0.439	0.379	0.12
i=8	0.315	0.122	0.237	0.319	0.195	-0.494
i=9	0.191	-0.133	0.382	0.441	0.405	0.2

i=10	-0.49	0.497	0.197	0.321	-0.239	0.125
i=11	0.34	0.118	-0.492	0.499	0.199	0.323
i=12	0.126	-0.249	0.127	0.341	-0.491	0.192
i=13	0.248	0.316	0.232	-0.119	0.379	0.438
i=14	-0.322	0.4	0.196	0.441	0.241	-0.291
i=15	0.198	0.195	0.12	-0.419	0.493	0.197
i=16	0.498	-0.122	0.121	0.495	0.201	-0.126
i=17	0.491	0.324	0.193	0.439	-0.292	0.198
i=18	0.118	0.377	-0.195	0.437	0.326	0.293
i=19	0.492	-0.193	0.12	0.317	0.124	0.202
i=20	-0.24	0.401	0.119	0.438	-0.197	0.492
i=21	-0.196	0.236	0.493	0.318	0.402	0.242
i=22	0.121	0.378	0.197	-0.439	0.318	-0.38
i=23	0.12	0.195	0.122	0.399	-0.291	0.492
i=24	0.494	0.403	0.121	0.235	0.12	-0.2
i=25	0.198	0.238	-0.495	0.378	0.499	0.24
i=26	0.123	0.383	0.201	-0.194	0.135	0.406
i=27	0.134	0.198	-0.126	0.402	0.124	0.196
i=28	0.498	-0.493	0.324	0.237	0.199	0.38
i=29	-0.119	0.128	0.193	-0.379	0.119	-237
i=30	0.29	0.233	0.439	0.196	-0.492	0.402
i=31	0.317	0.197	-0.292	0.406	0.2	0.194
i=32	0.401	-0.121	0.198	0.24	0.121	0.378
i=33	-0.196	0.194	-0.127	0.2	0.196	-0.235
i=34	0.123	0.122	-0.199	0.498	0.194	0.399
i=35	0.325	-0.342	0.493	-0.202	0.123	0.195

(lanjutan)

Untuk nilai ambang (limit) diset juga ke dalam bilangan yang kecil, yaitu sebesar 0.002 dengann harapan agar keakuratan hasil dapat dipertanggungjawabkan. Sedangkan untuk nilai alpha yang digunakan adalah sebesar 0,2.

b) Penentuan kondisi berhenti

Pelatihan jaringan akan berhenti jika informasi error jaringan <= limit.

c) Set pelatihan sejumlah 183 kasus (pasangan vektor input dan vektor output) akan menjalani tahap-tahap sebagai berikut:

➢ Umpan maju

Pada tahap ini setiap unit input (x_i, i=1,...,35) menerima sinyal input x_i dan menyiarkan sinyal ini ke semua unit (node) yang ada pada lapisan di atasnya, yaitu unit-unit tersembunyi.

Unit-unit tersembunyi (z_j, j=1,...,35) menjumlahkan sinyal –sinyal input berbobotnya dengan menggunakan formula berikut:

$$z\_in_i = v_{0i} + \sum_{i=1}^{35} x_i \, v_{ij}$$

Kemudian menerapkan fungsi aktivasi untuk menghitung sinyal outputnya:

$$z_i = f(z\_in_j)$$

$$= \frac{1}{1+e^{-z\_in_j}}$$

Sinyal ini dikirim ke semua unit di lapisan atasnya (ke unit-unit output). Setiap unit output (Y_k, k=1,...,6) menjumlahkan sinyal-sinyal input berbobotnya dengan formula:

$$y\_in_k = w_{0k} + \sum_{j=1}^{35} z_j \, w_{jk}$$

Dan menerapkan kembali fungsi aktivasi untuk menghitung sinyal outputnya,

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

$$= 1/(1+e^{-y\_in_k}$$

- Pemropagasibalikan error

  Unit-unit output (Y_k, k = 1,...,6) menerima masing-masing sebuah pola target (t_k) yang berhubungan dengan pola latihan input, kemudian persamaan informasi errornya (δ_k) dihitung,

$$\begin{aligned} ¢_k &= (t_k - y_k) f'(y\_in_k) \\ &= (t_k - y_k).(1 - y_k) \end{aligned}$$

  Demikian pula persamaan koreksi bobotnya yang kelak dipergunakan untuk mengupdate w_{jk},

$$\Delta w_{jk} = \sim ¢_k z_j$$

  Menghitung persamaan biasnya dengan formula seperti berikut:

$$\Delta w_{0k} = \sim ¢_k,$$

  Lalu menyiarkan δ_k ke lapisan di bawahnya (δ_k dipropagasibalikan). Persamaan koreksi bias ini nantinya dipergunakan untuk mengu-update w_{0k}.

  Setiap unit tersembunyi (z_j, j=1,...,35) menjumlahkan input-input deltanya (dari unit-unit pada lapisan diatasnya),

$$¢\_in_j = \sum_{k=1}^{6} ¢_k w_{jk}$$

  Mengalikannya dengan turunan dari fungsi aktivasinya untuk menghitung persamaan informasi errornya dengan menggunakan formula :

$$¢_j = ¢\_in.(z\_in_j).(1 - z\_in_j)$$

  menghitung persamaan koreksi bobotnya yang akan digunakan untuk mengupdate v_{ij} ,

$$\Delta_{ij} = \sim ¢_j x_i$$

  Dan menghitung persamaan koreksi biasnya yang akan digunakan untuk mengupdate v_{0j} :

$$\Delta_{0j} = \sim ¢_j$$

- Peng-update-an bobot dan bias

  Tiap-tiap unit output (Y_k, k = 1,...,) akan mengupdate bias dan bobotnya (j=0,...,35) :

$$w_{jk}(baru) = w_{jk}(lama) + w_{jk}$$

  Setiap unit bersembunyi (z_j, j=1,...,35) mengupdate biasnya dan juga bobotnya (i=0,...,35) :

$$v_{ij}(baru) = v_{ij}(lama) + v_{ij}$$

  Langkah nomor 3 akan terus diulangi sampai kondisi berhenti dipenuhi sehingga mungkin akan memerlukan banyak sekali iterasi seperti ini sebelum proses pembelajaran (pelatihan) berhenti.

  Sementara itu pada pengujian jaringan maupun aplikasi jaringan sebagai medium pengenal karakter alfanumerik yang diterapkan hanya tahap umpan maju (algoritma aplikasi) seperti berikut ini:

  1) Setelah pelatihan selesai bobot-bobot yang diperoleh digunakan untuk menginisialisasi bobot pada pengujian atau aplikasi.
  2) Untuk setiap vektor input (misalnya set tes), yang terjadi adalah:
     - Memasukkan unit input x_i .
     - Melakukan perhitungan node-node (unit-unit) tersembunyi:

$$z\_in_j = v_{0j} + \sum_{i=j}^{35} x_i . v_{ij}$$

       Dengan (j = 1,...,35).
     - Menghitung sinyal outputnya (j = 1,...,35):

$$z_i = f(z\_in_j)$$

$$= 1/(1+e^{-z\_in_j}$$

Sinyal ini lalu dikirim ke semua unit di lapisan atasnya (ke unit-unit output).

- Setiap unit output (Y_k, k=1,...,6) menjumlahkan sinyal-sinyal input berbobotnya:

$$y\_in_k = w_{0k} + \sum_{j=1}^{6} z_j .w_{jk}$$

- Menerapkan fungsi aktivasi untuk menghitung sinyal output,

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

$$= 1/(1+e^{-y\_in_k})$$

Set data yang meliputi 273 pasangan vektor input dan vektor output (untuk pengenalan 3 karakter : 'A', 'B', 'C') dibagi ke dalam 2 bagian, yaitu set pelatihan dan set tes. Set pelatihan terdiri dari 183 pasangan vektor input dann vektor output dengan karakter 'A', 'B', dan 'C' masing-masing teridiri dari 61 pasangan. Pelatihan berjalan memadai dengan hasil akhir keseluruhan set pelatihan (183 kasus) dapat dikenali, baik yang mengandung noise ataupun yang tidak.

7.3.2 Verifikasi dan Validasi

Jaringan syaraf tiruan dibagi dengan menggunakan 2 set data, yaitu set pelatihan dan set tes. Seperti telah dikemukakan sebelumnya, set pelatihan terdiri dari 183 pasangan vektor input dan vektor output dengan karakter 'A', 'B', dan 'C', masing-masing 61 pasangan. Set data tes terdiri dari 90 pasangan vektor input dan vektor output dengan karakter 'A', 'B', dan 'C', masing-masing 30 pasangan. Set pelatihan digunakan untuk menguji kemampuan memorisasi jaringan. Set tes digunakan untuk menguji kemampuan generalisasi jaringan.

Ketika jaringan diuji dengan menggunakan set pelatihan, yang diuji adalah ingatan jaringan, sebab kasus-kasus yang dimasukkan telah ia pelajari sebelumnya. Sebaliknya, pada saat

jaringan diuji dengan menggunakan set tes, apa yang dimasukkan ke dalam jaringan adalah kasus-kasus yang belum pernah dipelajari oleh jaringan. Dari pengalaman selama pelatihan diharapkan jaringan akan mampu menggenerealisasikan kasus yang ia hadapi dan kemudian menarik kesimpulan yang cenderung ke output tertentu.

Hasil percobaan memperlihatkan bahwa jaringan syaraf tiruan yang dirancang ternyata dapat mengenali dengan baik sekali kesembilan puluh alfanumerik (berhasil mengenali 100 persen) baik karakter 'A', karakter 'B', maupun karakter 'C' yang datanya mengandung noise.

C. Rangkuman

1. Pada aplikasi pengenalan karakter alfanumerik menggunakan metode propagasi balik, terdapat beberapa konsep yang harus dilalui antara lain:
   a. Tahap konsep yang terdiri dari:
      1) Pemilihan aplikasi
      2) Pemilihan paradigma
   b. Tahap desain yang terdiri dari:
      1) Pendesainan jaringan syaraf tiruan
      2) Pengumpulan data pelatihan
      3) Pemilihan lingkungan pengembangan
   c. Tahap implementasi
      1) Implementasi dan pelatihan
      2) Verifikasi dan validasi
2. Ada 3 tingkat dalam pendesainan sebuah sistem jaringan syaraf tiruan, diantaranya:
   a. Tingkat node
   b. Tingkat jaringan
   c. Tingkat pelatihan
3. Langkah-langkah dalam pengumpulan data untuk membangun aplikasi pengenalan karakter alfanumerik adalah:
   a. Membuat rancangan data yang akan digunakan sebagai latihan dan testing
   b. Memisahkan data ke dalam 2 bagian, yaitu set pelatihan (set training) dan set testing

D. Tugas

1. Buatlah kesimpulan mengenai materi yang telah dibahas!
2. Carilah referensi lain mengenai materi pengenalan karakter alfanumerik menggunakan propagasi balik!

E. Tes Formatif

1. Apa yang anda ketahui tentang aplikasi pengenalan karakter alfanumerik?
2. Sebutkan 3 lapisan yang terdapat pada jaringan syaraf tiruan dan jelaskan prinsip kerja dari ketiga lapisan tersebut!
3. Berikanlah satu contoh kemungkinan penerapan pengenalan karakter alfanumerik!

BAB 8

PENELITIAN-PENELITIAN DAN APLIKASI JARINGAN SYARAF TIRUAN

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan beberapa penelitian yang menggunakan jaringan syaraf tiruan.
2. Menjelaskan beberapa aplikasi jaringan syaraf tiruan.

B. Uraian Materi

8.1 Penelitian dengan Bantuan Jaringan Syaraf Tiruan

Berikut ini dibahas beberapa penelitian yang menggunakaan jaringan syaraf tiruan:

a. *Hypernet* : penerapan jaringan syaraf tiruan – sistem pakar untuk pendiagnosisan dan perawatan tekanan darah tinggi,
b. Pengenalan pola tulisan tangan : menggunakan jaringan syaraf tiruan dan logika samar, dan
c. Mesin ketik suara (neural Phonetic typewriter) yang menerjemahkan suara ke dalam bentuk tulisan.

8.1.1 *Hypernet*

Hypernet adalah perpaduan antara sistem pakar dengan jaringan syaraf tiruan yang digunakan untuk pendiagnosisan dan perawatan penyakit darah tinggi. *Hypernet* tidak dapat digolongkan sebagai sistem pakar klasik karena pengetahuan tidak dikodekan dalam bentuk simbolik (berbentuk frame-frame atau rule-rule) tetapi pengetahuan tersebut didistriusikan ke seluruh jaringan. Pelatihan atau pembelajaran jaringan ini dapat digolongkan sebagai basis pengetahuan yang mudah dimodifikasi dengan bantuan program basis data sederhana. Hypernet dikembangkan oleh beberapa pakar italia, diantaranya Riccardo Poli, Stefano Cagnoni (*University of florence*), Riccardi Livi (*University Center Of Clinical Chronobiology*), dan kawan-kawan.

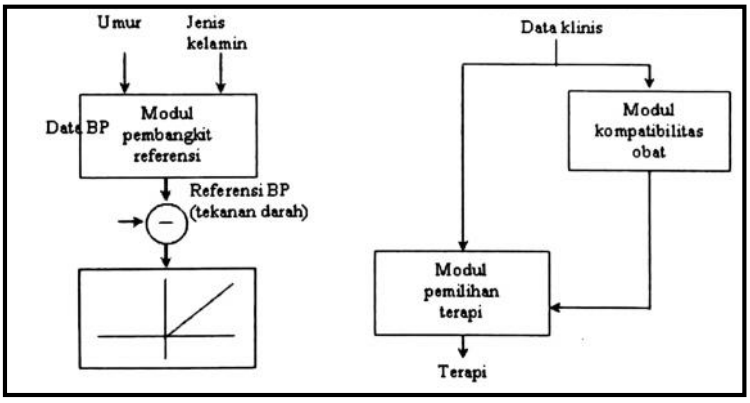
Mayoritas keputusan-keputusan klinis, kesimpulan-kesimpulan yang kompeks, dan juga pengetahuan pathofisiologi, diambil berdasarkan pengalaman ahli medis. Namun demiikian tidak semua pengalaman dapat dibuat ke dalam set relasi yang kecil sehingga hal ini membuat pendekatan algoritmik kurang leluasa untuk digunakan. Dilain pihak jaringan syaraf tiruan memiliki properti-properti seperti kemampuan belajar berdasarkan pengalaman, memiliki toleransi terhadap kesalahan (jaringan tidak sensitif terhadap noise pada pola input), degredasi kinerja yang terjadi perlahan-lahan (bial noise yang diberikan pada pola input

sangat besar) dan meningkatnya sinyal, yang mampu mengatasi permasalahan di atas. Hal ini yang melatar belakangi penggunaan jaringan syaraf tiruan pada aplikasi ini.

Artsitektur sistem

Gambar 8.1 menggambarkan struktur logika *hypernet* yang mensimulasikan pengambilan keputusan seorang dokter yang dibagi dalam 3 buah modul utama, yaitu modul pembangkit referensi (RGM), modul kompabilitas obat (DCM), dan modul pemilihan terapi (TSM). Metode jaringan syaraf tiruan yang digunakan pada setiap modul adalah metode propagasi balik.

Input : data amnestik subjek-subjek (pasien/orang sehat) dan serangkaian tekanan darah diastolik dalam skala waktu 24 jam. **Output** : 4 buah array yang berisikan 24 item, yang masing-masing nilainya menunjukkan dosis per jam obat antihipertensi yang paling umum digunakan.



Gambar 8.1 Struktur logika hypernet

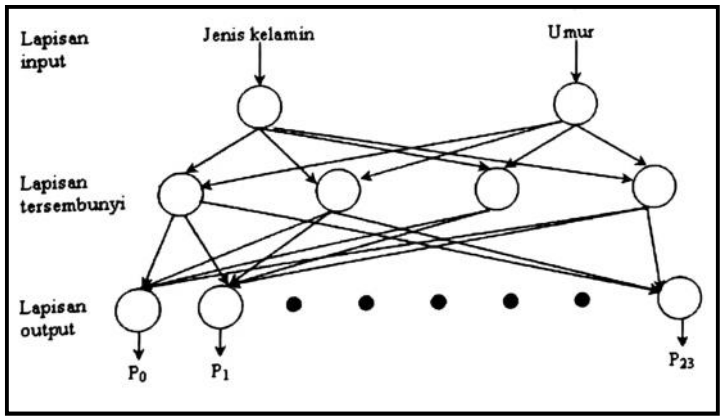
✓ RGM

RGM digambarkan seperti pada Gambar 8.2. tujuan dari modul ini adalah untuk memanggil pengetahuan medis mengenai konsep normalitas dari rekaman tekanan darah.

- Lapisan input : terdiri dari 2 buah unit input (melambungkan jenis kelamin dan umur)

- Lapisan tersembunyi : terdiri dari 4 buah unit tersembunyi.
- Lapisan output : terdiri dari 24 unit output (P_0 ... P_{23}) melambangkan serangkaian tekanann darah selama 24 jam.

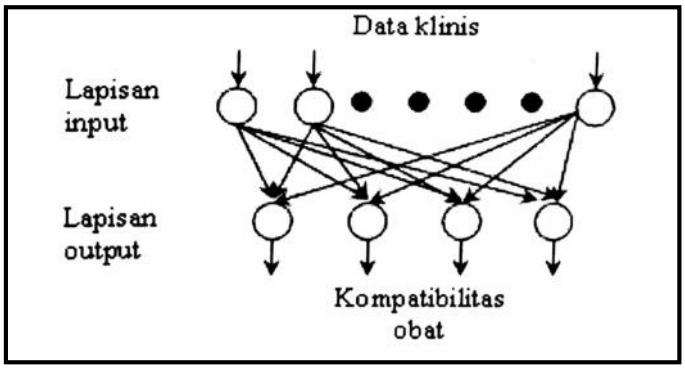
Pada jaringan RGM dilakukan perbandingan antara serangkaian tekanan darah yang dimiliki oleh orang yang normal dengan tekanan darah pasien (kedua subjek tersebut memiliki jenis kelamin dan umur yang sama).



Gambar 8.2. Arsitektur modul pembangkit referensi (reference Generating Module = RGM

✓ DCM

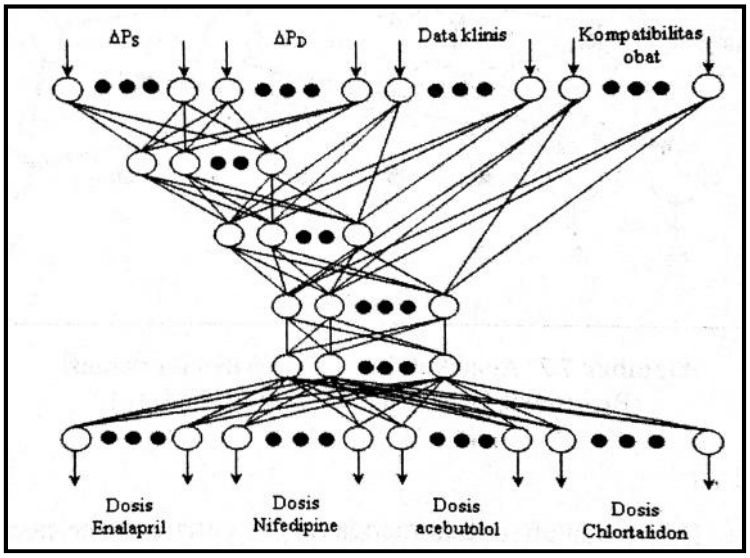
Tujuan DCM adalah untuk menganalisis entri-entri terpenting dari laporan kinis pasien dan untuk menentukan derajat kecocokan si pasien dengan masing-masing obat yang dipertimbangkan akan diberikan kepadanya. DCM berupa jaringan syaraf sederhana yang terdiri dari 2 lapisan berisi 17 input (masukan laporan klinis pasien) dan 4 buah unit output (lihat Gambar 7.3)



Gambar 8.3 Arsitektur modul kompabilitas obat (Drug Compability Module)

✓ TSM

TSM bertujuan memilih kombinasi obat yang paling sesuai untuk pasien, dan obat yang harus diberikan untuk setiap obat terpilih (beberapa dosis setiap kali pemberian). TSM terdiri dari 6 lapis jaringan syaraf tiruan (lihat Gambar 8.4) berikut:



Gambar 8.4 Arsitektur module pemilihan terapi (Therapy-Selecting-Module = TSM)

i. Lapisan input; berupa derajat kompabilitas obat, perbedaan tekanan darah sistolik (ΔP_s), perbedaan tekanan darah diastolik (ΔP_D) , dan 5 data anamnesis yang bisa mempengaruhi diagnosis.
ii. Lapisan tersembunyi terdiri dari 4 lapisan tersembunyi:
    - Lapisan tersembunyi pertama berisi 12 unit yang input-inputnya adalah ΔP_s dan ΔP_D. hal ini merupakan penggambaran terbesar kelebihan tekanan darah
    - Lapisan tersembunyi kedua berisi 12 unit tambahan yang terhubung balik ke lapisan sebelumnya maupun ke lima buah unit yang menggambarkan data anamnesis.
    - Lapisan tersembunyi ketiga terdiri dari 12 unit yang terhubung baik ke lapisan sebelumnya maupun ke 4 buah unit yang menggambarkan perkiraan kompatibilitas data oleh DCM.
    - Lapisan tersembunyi keempat yang menggambarkan keseluruhan data.
iii. Lapisan output : berisi dosis 4 macam obat (enalpril, nifedipine, acebutolol, dan chlortalidon).

Jalannya percobaan

Jaringan syaraf tiruan pada *hypernet* ini didefenisikan dengan menggunakan bahasa deskriptif simbolik sederhana. Pada saat file definisi jaringan telah telah siap, program C oleh pengembang *hypernet* dinamakan sebagai kompiler jaringan syaraf tiruan menerjemahkan primitif-primitif bahasa tersebut ke dalam struktur data yang cocok, lalu pelatihan jaringan dimulai (NNC mempproses sekitar 20.000 bobot perdetik PC 80386 20-MHz). setelah pelatihan, NNC mengeksekusi set tes ke dalam bentuk primitif-primitif bahasa, dan kemudian menuliskan file definisi jaringan yang telah di-update tersebut pada disk.

Data yang digunakan untuk melatih dan menguji unjuk kerja jaringan terdiri dari serangkaian waktu tekanan darah dari sekitar

300 subjek yang dinyatakan sehat oleh Italian Multicentric Study on Blood Pressure Variability, dan juga dari 85 subjek yang diduga mengidap penyakit tekanan darah tinggi.

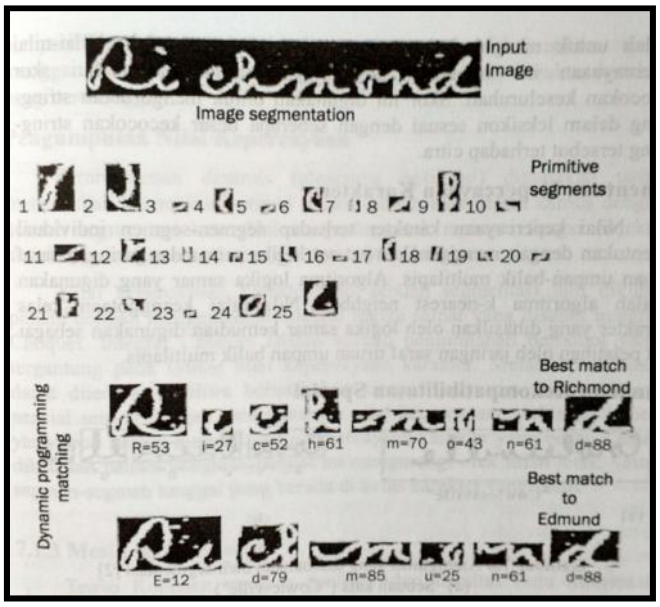
Tabel 8.1 hasil evaluasi diagnosis *hypernet* oleh spesialis

	Benar	Salah
Dirawat	11	1
Tidak dirawat	22	1
Total	33	2

Hasil yang diperoleh: hypernet berhasil mendiagnosis dengan benar 33 subjek dari 35 subjek (94%), menunjukkan 92% sensivitas dan 96% terjadi spesivisitas (tidak perlu dirawat, tetapi membutuhkan obat khusus).

8.1.2 Metode Jaringan Syaraf Tiruan – Logika Samar dalam Pengenalan Tulisan Tangan

Pengenalan tulisan tangan melibatkan percobaan citra digital sebuah kata tulisan tangan sampai ke pencocokan kamusnya. Pengenalan tulisan tangan dikatakan sukses bila ia bisa mencapai interpretasi terbaik. Jadi tidak hanya sekedar berhasil mengisolasi dan mengenali karakter-karakter individual. Untuk pencapaian interpretasi terbaik, informasi mengenai konteks dari kata tersebut harus diketahui. Paul D. Gader, James M. Keller, Raghu Krishnapuram, Jung-hsien Chiang, dan Magdi A. mohammed dari University of Missouri, AS, menggunakan jaringan syaraf tiruan dan logika samar untuk memecahkan masalah pengenalan tulisan tangan tersebut. Jaringan syaraf tiruan digunakan untuk memproses informasi yang tidak tepat karena jaringan syaraf tiruan mampu melakukan pendekatan batasan keputusan yang rumit dengan baik. Metode logika samar digunakan untuk mewakili derajat kebenaran.



Gambar 8.5 Pendekatan segmentasi pada pengenalan tulisan tangan

Program pengenalan kata tulisan tangan yang berhasil kembangkan menerima input berupa sebuah citra digital dari sebuah kata dan sebuah kamus. Struktur program ini ditunjukkan seperti pada gambar 8.5

Pada gambar Gambar 8.5 tersebut kata *Richmond* dibagi ke dalam 25 primitif lalu dilakukan proses pencocokan untuk menemukan cara terbaik menyusun primitif-primitif tersebut agar bisa sesuai dengan salah satu kata yang ada dalam kamus dengan menggunakan algoritma optimasi. Program kemudian menetapkan nilai kebenaran antara 0 sampai 100 terhadap setiap segmen yang menunjukkan derajat kmiripan sebuah segmen terhadap sebuah karakter.

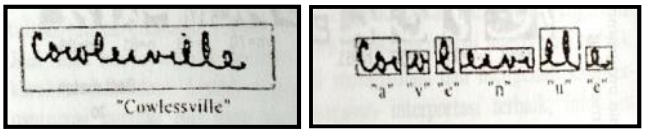
Segmen yang menggambarkan citra-citra nonkarakter, misalnya *Edmund* pada Gambar 8.5 diatas harus diberikan nilai kepercayaan yang rendah untuk menghindari penginterpretasian

yang salah. Nilai-nilai kepercayaan ini kemudian digabungkan untuk mengetahui skor kecocokan keseluruhan. Skor ini digunakan untuk mengurutkan string-string dalam lekisong sesuai dengan seberapa besar kecocokan string-string tersebut terhadap citra.

- ✓ Penentuan kepercayaan karakter

Nilai kepercayaan karakter terhadap segmen-segmen individual ditentukan dengan menggunakan set-set logika samar dan jaringan syaraf tiruan umpan-balik multilapis. Algoritma logika samar yang digunakan adalah algoritma *k-nearest neighbor*. Nilai-nilai keanggotaan kelas karakter yang dihasilkan oleh logika samar kemudian digunakan sebagai set pelatihan oleh jaringan syaraf tiruan umpan balik multilapis.

- ✓ Jaminan kekompabilitasan spasial



Gambar 8.6 Ambiguitas yang disebabkan oleh segmentasi (a) Sebuah kata ("cowlesville). (b) Segmentasi kata yang tidak tepat meyebabkan pencocokan yang buruk meskipun potongan-potongan citra menunjukkan karakter-karakter yang bisa diterima

Segmen-segmen individu bisa terlihat mirip seperti karakter hanya berbeda dalam ukuran dan lokasi spasial semisal segmen ke-5 dan segmen ke-6 dalam Gambar 8.6. untuk itu segmen ke-5 dan ke-6 tersebut harus diberikan nilai keanggotaan kelas karakter yang tinggi pada kelas "u" dan kelas "e". sedangkan untuk mengukur kompabilitas lokasi spasial relatif dari pasangan segmen-segmen yang bertetangga (dengan interpretasi sebagai pasangan karakter) digunakan jaringan syaraf tiruan umpan-balik multilapis lagi.

Hasil penelitian menunjukkan pada saat keanggotaan dalam kelas-kelas kompabilitas diterapkan untuk mengaugmentasikan nilai karakter logika samar, unjuk kerja pengenalan kata menunjukkan peningkatan dari 79,5% menjadi 85,3%. Adapun yang dimaksud dengan kelas-kelas kompabilitas, yaitu kelas-kelas pola yang didefinisikan menggambarkan konfigurasi yang berbeda-beda dari karakter-karakter yang bertetangga.

- ✓ Pengumpulan nilai kepercayaan

Pemrograman dinamis (algoritma optimasi) digunakan untuk menemukan kecocokan optimal antara sebuah kata dalam kamus dengan sebuah segmentasi kata. Hal ini perlu untuk mencari fungsi objektif standar, yaitu rata-rata nilai kepercayaan karakter. Nilai kepercayaan karakter dari sebuah segmentasi lalu dikumpulkan dengan menggunakan integral logika samar yang disebut integral Choquet. Integral Choquet adalah sebuah penjumlahan berbobot yang tergantung pada urutan nilai kepercayaan karakter. Melalui percobaan dapat ditentukan bahwa bobot-bobot menjadi menurun secara eksponsial seiring dengan meningkatnya nilai kepercayaan, sehingga bobot yang kecil berarti memiliki nilai kepercayaan karakter yang besar. Dapat dikatakan bahwa iintegral Choquet ini mengurangi efek salah letak, yaitu segmen-segmen tunggal yang berada di kelas karakter yang salah.

- ✓ Mesin ketik suara

Teuvo Kohonen, seorang professor dari fakultas ilmu informasi Helsinki University off Technology (Finlandia), mengembangkan mesin ketik suara yang mampu mengonversikan suara ke dalam bentuk tulisan. Masalah yang paling signifikan dalam pembuatan sistem ini adalah masalah pengenalan suara (*speech recognition*).

Pengenalan suara merupakan masalah yang cukup rumit dan melibatkan banyak hal, mulai dari pendekatan fonem-fonem sampai kepada pemahaman pesan yang disampaikan. Contoh-contoh masalah yang harus ditangani dalam pengenalan suara diantaranya:

a. Ketika mendengar sebuah ucapan yang asing di telinga, maka secara tidak sadar manusia langsung menguji dan mengulangi kembali persepsi dari berbagai pengalaman berisi konteks-konteks yang telah akrab didengarnya. Jadi apa yang telah dipercayai telah didengar berupa potongan-potongan informasi yang diterima kemudian akan mengalami rekonstruksi di dalam pikiran manusia. Pertanyaan yang timbul adalah: bagaimana caranya membuat sebuah sistem (mesin ketik suara) yang memiliki kemampuan merekonstruksi.
b. Distribusi sampel-sampel spektral mengalami overlap meskipun suara berasal dari pembicara yang sama dan diucapkan dengan jelas.
c. Fonem-fonem yang sama yang diucapkan oleh orang berbeda kadang-kadang juga bisa membingungkan.
d. Bagaimana membuat sebuah sistem pengenalan suara tingkat tinggi yang bisa menginterpetasikan isi semantik dari pengucapan/ pembicaraan.

Karena otak mampu berfungsi untuk mengenali suara, para peneliti beranggapan bahwa jaringan syaraf tiruan juga harus dapat berbuat hal yang sama. Pada penelitian ini metode jaringan syaraf tiruan yang digunakan adalah metode pemetaan yang mengorganisasi sendiri (self-organizing map) yang lebih dikenal dengan, sesuai nama penemunya, kohonen-SOM. Metode ini diluar pembahasan buku ini.

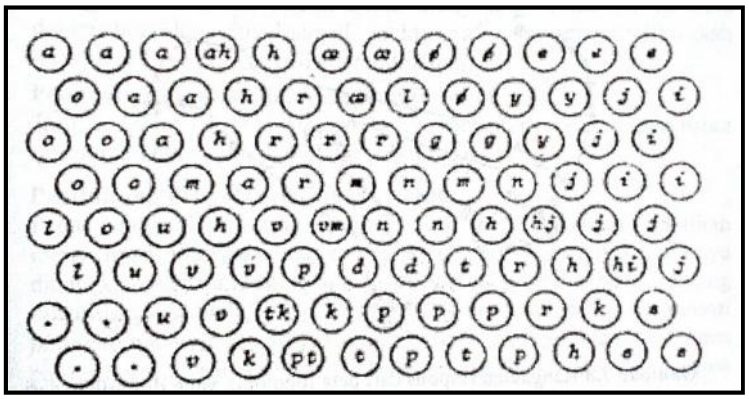
Mesin ketik menunjukkan potensi jaringan syaraf tiruan dalam membangun pengenalan suara *speaker-independent* pada sebuah

komputer. Lebih jauh lagi, mesin ketik suara juga menunjukkan bagaimana teknologi jaringan syaraf tiruan bisa digabungkan dengan pengolahan sinyal tradisional dan teknik-teknik standar kecerdasan buatan untuk memecahkan suatu masalah. Mesin ketik suara dapat menerjemahkan suara ke dalam bentuk teks tertulis dengan kamus yang tidak terbatas dan secara real time dengan keakurasian antara 92 hingga 97 persen. Alasan mengapa disebut mesin ketik karena alat ini hanya sekedar menerjemahkan, tidak dimaksudkan untuk mengerti makna pembicaraan. Mesin ketik suara hanya mendukung bahasa Finlandia dan bahasa Jepang.

Sebagai input sebuah array berupa node-node 2 dimensi dilatih dengan menggunakan 15 komponen analisis spektral dari kata-kata yang diucapkan dan diambil sebagai sampel setiap 9,83 milidetik. Vektor-vektor input ini dihasilkan dari serangkaian langkah pemrosesan awal suara, meliputi:

1. Penggunaan mikrofon penghilang noise (*noise-canceling-microphone*)
2. Preamplifier dengan sebuah kapaaasitor bersaklar, low-pass filter 5,3-kHz
3. Konverter 12 bit analog ke digital dengan angka sampling 13,02 kHz
4. Fast Fourier Transform 256-point, dihituung setiap 9,83 milidetik menggunakan sebuah jendela Hamming 256-point.
5. Logaritmisasi dan pemfilteran power spektrum menggunakan low pass filter *fourth order elliptic*.
6. Pengelompokan saluran-saluran spektrum ke dalam 15 komponen vektor pola-nyata.
7. Pengurangan rata-rata dari semua komponen, dan
8. Normalisasi 15 komponen vektor input yang dihasilkan ke dalam penjang konstan.

Dengan menggunakan algoritma pengklusteran kohonen, node-node dalam sebuah array 2 dimensi diijinkan mengorganisasi dirinya sendiri dalam berespons dengan vektor input. Setelah pelatihan dilakukan, peta yang dihasilkan disesuaikan dengan menggunakan spektrum fonem-fonem sebagai vektor inputnya. Kebanyakan node berespon terhadap sebuah fnem tunggal seperti ditunjukkan pada Gambar 8.7.

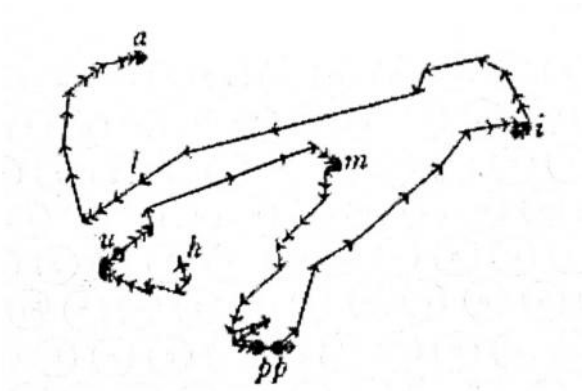


Gambar 8.7 Peta fonotopik neuron dan fenom. Neuron dilambangkan sebagai lingkaran dengan label didalamnya adalah fonem yang bersesuaian memberi respon terbaik. Fonem yang berlabel ganda menunjukkan bahwa node beersesuaian dengan lebih dari satu fonem. Fonem-fonem seperti k, p dan t sukar dibedakan oleh jaringan syaraf tiruan dan paling mudah mengalami penyalahgolongan.

Ketika sebuah kata diucapkan, kata tersebut disampel lalu dianalisis dan dikirimkan ke jaringan sebagai serankaian vektor input. Begitu node-node dalam jaringan berespons, sebuah jalur muncul pada peta yang bersesuaian dengan rangkaian pola input (lihat gambar 8.8 sebagai contoh "humppila"). Pada Gambar 8.8, tanda panah sesuai dengan waktu sampling 9,83 milidetik. Jalur ini merupakan transkripsi fonetik dari kata, yang kemudian dibandingkan dengan kata-kata yang sudah dikenali atau digunakan sebagai input ke sebuah sistem berbasis aturan. Analisis dilakukan secara cepat dan

efisien menggunakan berbagai teknik, termasuk jaringan syaraf tiruan memori asosiatif.

Secara singkat dapat dikatakan, begitu kata-kata diucapkan ke mikrofon maka transkripsinya akan muncul dilayar komputer. Dengan demikian mesin ketik suara ini akan sangat membantu dalam pekerjaan kantor modern. Pelatihan jaringan syaraf tiruan seorang pembicara individual hanya membutuhkan pendiktean sekitar 100 buah kata dengan memakan waktu sekitar 10 menit pada sebuah PC.



Gambar 8.8 Rangkaian respon dari peta fonotopik yang dihasilkan dari pengucapan kata Finladia "humpfilla"

8.2 Aplikasi Jaringan Syaraf Tiruan

berikut ini beberapa aplikasi jaringan syaraf tiruan:

8.2.1 Detector virus komputer

Jaringan syaraf tiruan digunakan untuk mendeteksi dan mengoreksi virus komputer. Contohnya, program anti virus IBM yang selain mendeteksi juga mengeradikasi virus-virus baru secara otomatis. Virus baru yang ditemukan kemudian digunakan sebagai set pelatihan untuk versi antivirus berikutnya yang jauh lebih cerdas. Jaringan syaraf tiruan yang digunakan adalah propagasi balik.

8.2.2 Pendeteksian kanker kulit

Jaringan syaraf tiruan digunakan untuk mendiagnosis malignant melanoma. Citra tumor dipisahkan antara malignant atau benign. Ada 3 kategori tumor benign yang dibedakan dari malignant melanoma. Citra digital dari tumor diklasifikasikan dengan menggunakan propagasi balik.

8.2.3 Pengidentifikasian pola-pola data pasar saham

Jaringan syaraf tiruan digunakan untuk memproses informasi dari basis data yang besar untuk mencari pola dan kecenderungan. Hasil pengolahan ini digunakan dalam kebutuhan investasi. Contohnya, apakah membeli atau menjual saham, atau apakah membeli kemudian menahannya dulu.

8.2.4 Pendeteksi bom

Jaringan syaraf tiruan dilatih untuk mengenali bentuk-bentuk bom dalam tampilan-tampilan sinyal spektograf yang mengindikasikan muatan nitrogen tiinggi pada barang-barang.

8.2.5 Pengontrol gerakan dan penglihatan robot

Jaringan syaraf tiruan digunakan untuk koordinasi mata-tangan robot untuk memegang objek melalui pelatihan robot.

8.2.6 Pendukung pengolahan bahasa alami

Pada pengenalan suara, jaringan syaraf tiruan melalui pelatihan menyimpan informasi bagian-bagian pembicaraan untuk nantinya dicocokkan secara cepat dengan pola-pola input. Sebuah sistem yang dikembangkan oleh Teuvo Kohonen, yaitu mesin tik suara (*neural phonetic typewriter*) yang memiliki akurasi paling tinggi dan kamus paling besar. Jaringan syaraf tiruan yang digunakan adalah *kononen self-orgnizing map*.

Sistem yang disebut NETtalk yang dikembangkan oleh Sejnowski dan Rosenbarg berupa sebuah jaringan syaraf tiruan tiga lapis yang mensintesis suara dari teks. Langkah pelatihannya mirip dengan tahap perkembangan anak yang belajar berbicara.

8.2.7 Pendukung DSS (=*Descision Support System*)

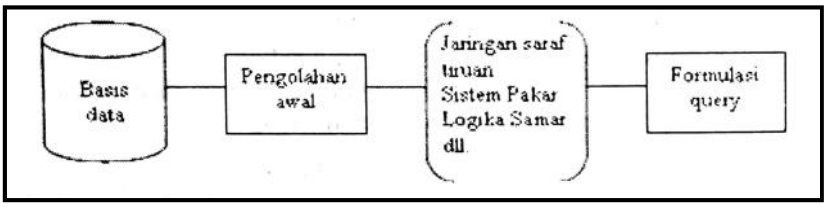
- Optimasi
  Jaringan syaraf tiruan dapat digunakan untuk menemukan solusi optimal dari masalah-masalah yang melibatkan banyak parameter, misalnya masalah TSP (*Traveling Salesperson Problem*).
- Alokasi sumber daya
  Alokasi ditemukan berdasarkan data historis.

8.2.8 Sistem hibrid

Sistem yang melakukan analisis statistikal standar pada kumpulan data yang telah diseleksi oleh jaringan syaraf tiruan.

8.2.9 Basis data cerdas (*intelligence database*)

Tujuan dari sistem basis data cerdasa adalah untuk menangani informasi dan pembuatan keputusan dengan cara yang lebih mirip dengan cara manusia. Jaringan syaraf tiruan berperan penting dalam penemuan pola-pola data, menemukan pendekatan yang cocok, dan perkiraan terbaik. Jaringan syaraf tiruan juga memfasilitasi query tidak pasti.



Gambar 8.9 Arsitektur sebuah sistem basis

Arsitektur untuk sistem basis data cerdas seperti pada Gambar 8.9. fitur cerdas dari sistem ini berasal dari bermacam-macam teknologi, seperti sistem pakar, orientasi objek, logika samar, dan jaringan syaraf tiruan.

C. Rangkuman

1. *Hypernet* adalah perpaduan antara sistem pakar dengan jaringan syaraf tiruan yang digunakan untuk pendiagnosisan dan perawatan penyakit darah tinggi.
2. Arsitektur sistem dari hypernet terdiri dari:
   a. RGM
   b. DCM
   c. TSM
3. DCM berfungsi untuk menganalisis entri-entri terpenting dari laporan kinis pasien dan untuk menentukan derajat kecocokan si pasien dengan masing-masing obat yang dipertimbangkan akan diberikan kepadanya.
4. TSM bertujuan memilih kombinasi obat yang paling sesuai untuk pasien, dan obat yang harus diberikan untuk setiap obat terpilih (beberapa dosis setiap kali pemberian).
5. Jenis-jenis aplikasi jaringan syaraf tiruan antara lain:
   a. Detector virus komputer
   b. Pendeteksian kanker kulit
   c. Pengidentifikasian pola-pola data pasar saham
   d. Pendeteksi bom
   e. Pengontrol gerakan dan penglihatan robot
   f. Pendukung pengolahan bahasa alami
   g. Pendukung DSS (=*Descision Support System*)
   h. Sistem hibrid

i. Basis data cerdas (*intelligence database*)

D. Tugas

1. Carilah referensi lain mengenai penelitian-penelitian dan aplikasi jaringan syaraf tiruan!
2. Buatlah kesimpulan mengenai penelitian-penelitian dan aplikasi jaringan syaraf tiruan pada bab 8 ini!

E. Tes Formatif

1. Jelaskan pengertian dari hypernet!
2. Sebutkan beberapa aplikasi jaringan syaraf tiruan yang anda ketahui!
3. Jelaskan tujuan dari DCM!

BAB 9

JARINGAN SYARAF TIRUAN DAN SRATEGI INTEGRASI DENGAN EXPERT SYSTEM

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan keunggulan dan kelemahan jaringan syaraf tiruan.
2. Menjelaskan mengenai strategi integrasi jaringan syaraf tiruan – *expert system*

B. Uraian materi

9.1 Keunggulan Dan Kelemahan Jaringan Syaraf Tiruan

Sebagai alat pemecah masalah, jaringan syaraf tiruan memiliki keunggulan-keunggulan dan kelemahan-kelemahan. Beberapa keunggulan yang dimiiki oleh jaringan syaraf tiruan diantaranya:

1. Mampu memecahkan masalah yang sukar disimulasikan dengan menggunakan teknik analitikal logikal seperti pada sistem pakar dan teknologi software standar.
   Sebagai contoh, jaringan syaraf tiruan memiliki kemampuan menganalisis data (dalam jumlah besar) di mana aturannya tidak diketahui. Sebagai ganti dari aturan yang tidak diketahui tersebut, data yang ada tadi digunakan/diolah oleh jaringan syaraf tiruan untuk membangun pola.
2. Mampu memahami data yang dimasukkan meskipun data tersebut tidak lengkap (*incomplete data*) atau data yang terkena gangguan (*noisy data*).
3. Jaringan syaraf tiruan memiliki kelebihan yang sulit diciptakan dengan pendekatan simbolik/logikal dari teknik tradisional actificial intelligence, yaitu bahwa jaringan syaraf tiruan mampu belajar dari pengalaman.
4. Hemat biaya dan lebih nyaman bial dibandingkan dengan harus menulis program seperti software standar.
   Pada jaringan syaraf tiruan, yang perlu dilakukan adalah tinggal melatih jaringan untuk belajar dengan cara memasukkan set data berisi sekumpulan kasus ke dalam jaringan.
5. Jaringan syaraf tiruan terbuka untuk digabungkan dengan teknologi lain untuk menghasilkan sistem hibrida yang memiliki kemampuan memecahkan masalah dengan lebih baik lagi. Misalnya jaringan syaraf tiruan dengan *expert system*, dengan logika samar (*fuzzy*), dengan algoritma genetika, atau diintegrasikan dengan database.

Beberapa kelemahan yang dimiliki oleh jaringan syaraf tiruan adalah:

1. Jaringan syaraf tiruan kurang sesuai untuk aritmatika dan pengolahan data.
2. Jaringan syaraf tiruan masih membutuhkan campur tangan pakar untuk memasukkan pengetahuan dan menguji data.
3. Belum ditemukan cara terbaik untuk mempresentasikan data input, memilih arsitektur, serta jumlah node, dan jumlah lapisan. Cara yang digunakan hingga saat ini masih dengan cara coba-coba (*trial-and-error*).

   Untuk mempresentasikan data input dan output dalam jaringan syaraf tiruan, semua harus diubah ke dalam bentuk nilai antara 0 dan 1. Hal ini tentu saja membutuhkan pra-pemrosesan manipulasi data input yang membawa dampak perlu tambahan waktu, kekuatan CPU dan konsumsi ruang hardisk. Biasanya tool-tool jaringan syaraf tiruan menyediakan histogram untuk mengamati nilai-nilai kategorikal dan bisa secara otomatis mengubah nilai-nilai numerik ke dalam kisaran 0 sampai 1.
4. Jaringan syaraf tiruan kurang dapat menjelaskan hasil. Ini kritik terbesar yang sering dilontarkan tentang jaringan syaraf tiruan. Untuk domain aplikasi diman aturan-aturan penjelasan merupakan hal yang penting, misalnya seperti penolakan aplikasi pinjaman di bank, jaringan syaraf tiruan mungkin bukan tool tepat untuk menjelaskan. Akan lebih tepat jika menggunakan *expert system* lengkap dengan penjelasan penarikan inferensi. Sebaliknya jika apikasi lebih membutuhkan hasil, bukan pada pemahaman penarikan inferensi seperti pada prediksi pola-pola dasar saham, maka jaringan syaraf tiruan adalah tool yang tepat.

Dengan tidak mengesampingkan kelemahan-kelemahan di atas, jaringan syaraf tiruan secara umum dapat dikatakan baik dalam menangani masalah-masalah pada bidang-bidang berikut:

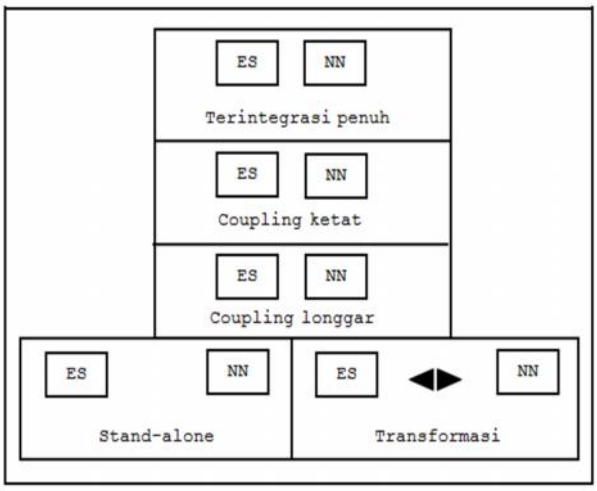
- Bidang yang melibatkan pengklasifikasian fitur geometrik atau fitur fisik. Contohnya:
  1. Transmisi data
  2. *Speech generation*

3. Pengenalan suara
4. Pembelajaran robot
5. Pengenalan tulisan.

- Pengklasifikasian pola data (penginterpretasian data dalam jumlah besar dan dari bermacam sumber). Contohnya:
  1. Analisis keuangan
  2. Diagnosis medikal
  3. Evaluasi permohonan pinjaman
  4. Peramalan banyak kursi pesawat yang akan laku
  5. Evaluasi karyawan dan pekerja yang cocok bagi karyawan tersebut.
  6. Diagnostik mesin jet dan roket.
- Bidang yang melibatkan masalah optimasi (contohnya traveling salesman problem).

9.2 Strategi Integrasi Jaringan Syaraf Tiruan – *Expert System*

Seperti telah diungkapkan pada bagian sebelumnya, salah satu kelebihan jaringan syaraf tiruan adalah ia bisa digabungkan dengan teknologi lain untuk menghasilkan sistem hibrida yang memiliki kemampuan pemecahan masalah yang lebih baik lagi. Model pengintegrasian jaringan syaraf tiruan dapat dilihat pada Gambar 9.1



Gambar 9.1 Model integrasi expert system (ES) dan jaringan syaraf tiruan (NN)

9.2.1 Model *Stand-Alone*

Sesuai namanya, pada model *stand-alone*, baik *expert system* maupun jaringan syaraf tiruan masing-masing berdiri sendiri, tidak ada ketergantungan di antara keduanya.

Tujuan penggunaan model stand-alone:

a. Sebagai sistem paralel ada 2 kemampuan yang ditawarkan oleh model *stand-alone*:
   - Kemampuan yang berasal dari jaringan syaraf tiruan, yaitu kemampuan generalisasi dan kemampuan adaptasi.
   - Kemampuan yang berasal dari *expert system*, yaitu langkah dedukasi dan fasilitas penjelasan.

b. Untuk memverifikasi aplikasi yang telah dibuat sebelumnya. Diberikan suatu permasalahan yang sama untuk dipecahkan secara terpisah, masing-masing oleh *expert system* dan jaringan syaraf tiruan. Hasil dari keduanya lalu dibandingkan sehingga dari keduanya bisa diketahui apakah sistem yang dibangun telah melaksanakan tugasnya dengan benar.

c. Untuk membangun prototype dalam waktu singkat sementara aplikasi yang sebenarnya yang lebih makan waktu tengah didesain.

Contoh: model *stand-alone expert* system dan jaringan syaraf tiruan pada perbaikan komputer

Keterangan:
Baik expert system maupun jaringan syaraf tiruan masing-masing dibangun untuk memecahkan masalah pengelompokan yang sama, yaitu masalah pengelompokan diagnostik gejala-gejala kerusakan komputer.

Cara kerja:
Ketika sebuah komputer tidak berfungsi sebagaimana mestinya, gejala-gejala dimasukkan ke dalam komponen (*expert system* dan jaringan syaraf tiruan). Keduanya kemudian mengeluarkan sebuah solusi. Kedua solusi dibandingkan, dan jika terdapat ketidaksamaan hasil dari keduanya, pemakai sistem boleh memilikih salah satu dari solusi-solusi itu untuk diimplementasikan.

Kelebihan model *stand-alone*:
- Lebih mudah dibangun karena tidak memerlukan antarmuka antara *expert system* dengan jaringan syaraf tiruan.
- Tinggal memakai paket peranngkat lunak komersial yang tersedia.

Kekurangan model *stand-alone*:
- Tidak dapat saling mendukung karena benar-benar independen satu sama lain
- Membutuhkan pemeliharaan ganda. Kedua komponen diupdate secara bersamaan untuk menghindari kebingungan.

9.2.2 Model transformasional

Serupa dengan model *stand-alone*, bedanya hanya bahwa sistem dimulai dengan salah satu tipe sistem (misalnya jaringan syaraf tiruan) dan sistem diakhiri dengan tipe lainnya (misalnya *expert system*).

Dengan bentuk model transformasi. *Expert system* yang ditransformasi ke jaringan syaraf tiruan dan jaringan syaraf tiruan yang ditransformasi ke *expert system*.

Contoh:
model transformasi untuk keputusan pemasaran

Keterangan:
analisis data dan adanya pengetahuan pendahuluan adalah ciri-ciri tipe model transformasi ini. Jaringan syaraf tiruan ditransformasi ke expert system untuk dokumentasi pengetahuan dan verifikasi, langkah-langkah pemikiran, dan kebutuhan fasilitas penjelasan.

Pada contoh ini, jaringan syaraf tiruan dibangun untuk mengidentifikasi trend dan hubungannya dengan data penjualan. Jaringan ini digunakan sebagai basis *expert system* dalam mengalokasikan sumber daya periklanan.

Cara kerja:
Jaringan syaraf tiruan digunakan untuk mengadaptasi masalah data intensif yang kompleks secara cepat untuk keperluan generalisasi dan untuk memfilter error pada data. Expert system ditargetkan sebagai sistem pemesanan dengan alasan dokumentasi dan vverifikasi pengetahuan yang digunakan pada pembuatan keputusan, dan karena penelitian pemasaran biasanya menginginkan sesuatu ditetapkan dengan alasan-alasan yang dapat dibenarkan.

Kelebihan model transformasi
- Dapat dibangun dengan cepat
- Pemeliharaan hanya pada satu sistem saja

Kekurangan model transformasi:
- Tidak terdapat alat yang otomatis secara penuh untuk mentransformasi jaringan syaraf tiruan ke *expert system* atau sebaliknya dari *expert system* ke jaringan syaraf tiruan.

- Modifikasi penting yang dilakukan terhadap suatu sistem mungkin akan membutuhkan suatu usaha pembangunan baru yang akan membawa ke sebuah transformasi yang berbeda.

9.2.3 Model *coupling* longgar

Sistem terdiri dari jaringan syaraf tiruan dan *expert system* yang terpisah di mana jaringan syaraf tiruan dan *expert system* berkomunikasi lewat file-file data. Beberapa variasi model *coupling* longgar adalah *prosesor*, *postprosesor*, *koprosesor* dan atarmuka pemakai (*user interface*).

Kelebihan model *coupling* longgar:
- Mudah dibangun
- Beban pemrograman berkurang karena bisa memakai perangkat lunak *expert system* dan jaringan syaraf tiruan komersial.
- Desain sistem dan proses implementasi lebih sederhana.
- Waktu pemeliharaan berkurang karena kesederhanaan mekanisme antarmuka file data.

Kekurangan model coupling longgar:
- Memakai waktu operasi yang lebih lama karena antar muka.
- Sering terjadi redundansi dalam membangun komponen-komponen jaringan syaraf tiruan dan *expert system* secara terpisah.
- Diperlukan biaya komunikasi yang tinggi untuk model coupling longgar ini.

Contoh-contoh model coupling longgar:
- Pada preprosesor, jaringan syaraf tiruan berfungsi sebagai front end yang mengondisikan data sebelum melewatkannya ke *expert system*. Sebagai contoh, jaringan syaraf tiruan melakukan penyatuan data, memindahkan error, mengidentifikasi objek, dan mengenali pola-pola yang informasinya kemudian oleh komponen *expert system* digunakan untuk memecahkan masalah, misalnya di bidang klasifikasi dan identifikasi.

- Pada postprosesing terjadi hal kebalikan dari preprosesing. *Expert system* melakukan persiapan data dan manipulasi, mengklasifikasikan input-input dan membuat keputusan. Output ini kemudian diteruskan lewat sebuah file data ke komponen jaringan syaraf tiruan yang kemudian melakukan peramalan, analisis data, memonitor, dan menjabarkan error.

9.2.4 Model *coupling* ketat

Model *coupling* ketat serupa dengan model *coupling* longgar. Bedanya, jaringan syaraf tiruan dan *expert system* meneruskan informasi lewat struktur datanya yang bersifat menetap (*memory resident*). Untuk preprosesor, postprosesor yang menggunakan model coupling ketat biasanya ebih cepat daripada yang menggunakan *coupling* longgar.

Kelebihan model *coupling* ketat:
- Mengurangi *overhead* komunikasi dan memperbaiki waktu unjuk kerja (bila dibandingkan dengan *coupling* longgar).
- Beberapa paket komersial tersedia, tinggal menggunakan.
- Fleksibilitas desain dan integrasi yang kokoh.

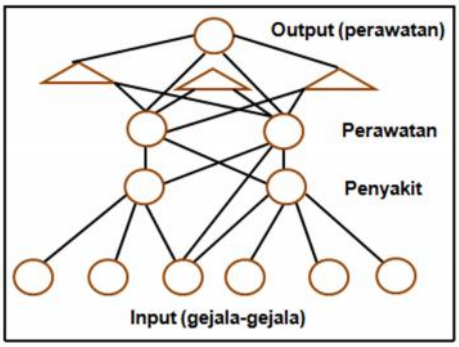
Kekurangan model *coupling* ketat:
- Kompleksitas pembangunan dan pemeliharaan meningkat karena menggunakan antar muka data internal.
- Sering terjadi redundansi pada pengumpulan data dan pegolahan.
- Verifikasi dan validasi yang lebih sulit.

Beberapa variasi lain dari coupling ketat:
- *Blackboard*: struktur data berbagi informasi yang memfalitasi pemecahan masalah interaktif lewat sistem basis pengetahuan. Contoh aplikasi: pengenalan pola kompleks dan pendukung keputusan tingkat lanjut.
- *Cooperative systems*: serupa dengan koprosesing model coupling longgar tetapi lebih interktif karena kemudahan lewatnya data. Contoh aplikasi antara lain: monitoring dan kontrol.

- *Embedded system*: modul-modu dari satu teknik membantu pengendalian fungsi yang lainnya. Komponen *embedded* jaringan syaraf tiruan digunakan untuk memfokuskan penarikan inferensi, pemandu pencarian, dan melakukan pencocokan pola. Komponen *expert system* digunakan untuk menginterpretasikan hasil yang diperoleh oleh jaringan syaraf tiruan, menyediakan konektivitas antarjaringan, dan untuk menyediakan fasilitas penjelasan. Contoh aplikasi *embedded system*: robotika dan klasifikasi.

9.2.5 Model terintegrasi penuh

Pada model terintegrasi penuh, *expert system* dan jaringan syaraf tiruan merupakan komponen dari data yang sama. Komponen jaringan syaraf tiruan merepresentasikan basis pengetahuan yang dimilikinya sebagai bobot-bobot koneksi. Jadi sistem didesain dengan bobot-bobot yang menggambarkan cabang-cabang logika dari basis aturan sehingga cara mendapatkan kesimpulan dapat dijelaskan. Lihat jaringan *Koneksionis Gallant* sebagai contohnya (Gambar 9.2)



Gambar 9.2 Arsitektur expert system koneksioniss Gallant

Node-neode input menggambarkan gejala-gejala. Nilai input +1, -1, atau 0 masing-masing mengindikasikan gejala itu ditemukan, tidak ditemukan, atau tidak teruji. Data pelatihan terdiri dari gejala-gejala dengan diagnosis yang sudah diketahui, digunakan untuk mencari bobot-bobot di antara node-node yang memberikan unjuk

kerja yang diingini. Node-node intermediat tambahan membuat sistem menjadi lebih akurat dan kokoh dalam menyarankan perawatan atas penyakit yang didiagnosis. Mesin inferensi digunakan untuk interpretasi hasil secara lebih jauh. Aspek *expert system* juga menyediakan penjelasan mengenai hasil-hasil yang diperoleh. Keuntungannya adalah kemampuan memakai file-file data pelatihan untuk mengubah perilaku sistem tanpa harus mengetahui atau menulis ulang aturan-aturan dalam basis pengetahuan.

Kelebihan model terintegrasi penuh:

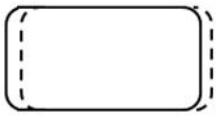
- Tegrasi yang kokoh
- Perbaikan untuk kerja
- Kemmampuan pemecahan masalah yang lebih baik

Kekurangan model integrasi penuh:

- Adanya kompleksitas dalam penspesifikasian, pendesainan, dan pengembangan sistem.
- Kurangnya tool-tol yang memfasilitasi model integrasi penuh.
- Sulit memverifikasi, dan memelihara sistem yang terintegrasi penuh.

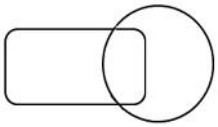
Strategi integrasi atau sinergisme antara jaringan syaraf tiruan dengan *expert System* juga dapat dilihat dari perspektif yang berbeda, yaitu dengan melihatnya dari segi fungsionall dan hubungan struktural. Keima arsitektur integrasi tersebut adalah sebagai berikut:

Overlap penuh



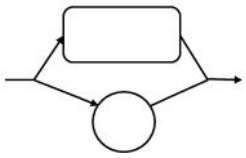
Memiliki sifat ganda: jaringan syaraf tiruan dan *expert system*. Bisa direpresentasikan kepada pemakai dalam bentuk *expert system* atau jaringan syaraf tiruan.

Overlap sebagian



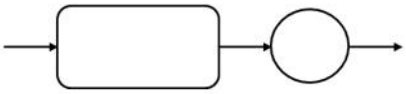
Sistem merupakan persilangan/hibrida dari *expert system* dan jaringan syaraf tiruan. Kedua kompoen berbagi beberapa bagian tetapi bukan keseluruhan variabel internal atau struktur data.

Paralel



Expert system dan jaringan syaraf tiruan bekerja sebagai paralel dalam memecahkan suatu masalah. Keduanya bisa berbentuk sistem yang stand-alone dan saling berkomunikasi melalui file data. Sebagaii contoh pada sistem daignosis medikal, jaringan syaraf tiruan menganalisis citra dan sinyal, ementara expert system mengin-terpretasi gejala-gejal kliinis. Hasil dari keduanya kemudia dikombinasikan.

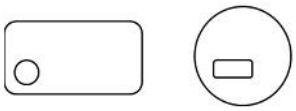
Sekuensial



Expert system dan jaringan syaraf tiruan beroperasi secara berurutan dalam memecahkan masalah. Keduanya juga bisa berbentuk sistem yang stand-alone. Output dari satu komponen diteruskan ke komponen lain untuk pemrosesan lebih lanjut.

Contohnya, jaringan syaraf tiruan yang digunakan sebagai front-rnd (*prepro-sesor*) untuk memfilter *noise* yang kemudian oleh *expert system* dilanjutkan dengan mentransformasikan sinyal ke dalam simbol.

Embedded



Expert system embedded di dalam jaringan syaraf tiruan atau sebaliknya. X embedded di dalam Y maksudnya X (tamu) menjadi elemen dari Y (*host*). Jadi fitur eksternal dari sistem ditentukan hanya oleh komponen *hoost.*

Contoh jaringan syaraf tiruan yang terembedded dalam expert sistem pada sistem pemahaman bahasa, jaringan syaraf tiruan digunakan untuk menganalisis sinyal tingkat rendah dan perilaku yang bergantung pada waktu.

C. Rangkuman

Beberapa keunggulan dari jaringan syaraf tiruan, antara lain:

1. Mampu memecahkan masalah yang sukar disimulasikan dengan menggunakan teknik analitikal logikal seperti pada sistem pakar dan teknologi software standar.
2. Mampu memahami data yang dimasukkan meskipun data tersebut tidak lengkap (*incomplete data*) atau data yang terkena gangguan (*noisy data*).
3. Jaringan syaraf tiruan memiliki kelebihan yang sulit diciptakan dengan pendekatan simbolik/logikal dari teknik tradisional actificial intelligence, yaitu bahwa jaringan syaraf tiruan mampu belajar dari pengalaman.
4. Hemat biaya dan lebih nyaman bial dibandingkan dengan harus menulis program seperti software standar.
5. Jaringan syaraf tiruan terbuka untuk digabungkan dengan teknologi lain untuk menghasilkan sistem hibrida yang memiliki kemampuan memecahkan masalah dengan lebih baik lagi. Misalnya jaringan syaraf tiruan dengan *expert system*, dengan logika samar (*fuzzy*), dengan algoritma genetika, atau diintegrasikan dengan database.

Beberapa kelemahan yang dimiliki oleh jaringan syaraf tiruan adalah:

1. Jaringan syaraf tiruan kurang sesuai untuk aritmatika dan pengolahan data.
2. Jaringan syaraf tiruan masih membutuhkan campur tangan pakar untuk memasukkan pengetahuan dan menguji data.
3. Belum ditemukan cara terbaik untuk mempresentasikan data input, memilih arsitektur, serta jumlah node, dan jumlah lapisan. Cara yang digunakan hingga saat ini masih dengan cara coba-coba (*trial-and-error*).
4. Jaringan sysaraf tiruan kurang dapat menjelaskan hasil.

Strategi integrasi jaringan syaraf tiruan – *Expert system*, antara lain:

1. Model *stand-alone*
2. Model transformasional
3. Model *coupling* longgar
4. Model *coupling* ketat
5. Model terintegrasi penuh

C. Tugas

Buatlah kesimpulan mengenai jaringan syaraf tiruan dan srategi integrasi dengan expert system yang telah dibahas pada bab 9 ini.

D. Tes formatif

1. Apa yang anda ketahui mengenai expert sistem?
2. Sebutkan beberapa kelebihan dan kekurangan model terintegrasi penuh yang anda ketahui!
3. Apa yang dimaksud dengan stand alone?

BAB 10

LOGIKA FUZZY

A. Tujuan Pembelajaran

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan mengenai variabel linguistik.
2. Menjelaskan mengenai logika fuzzy.
3. Menjelaskan beberapa aturan dalam penarikan kesimpulan berdasarkan pendekatan (fuzzy reasorning).

B. Uraian Materi

10.1 Variabel Linguistik

Variabel linguistik merupakan konsep penting pada logika fuzzy dan memegang peranan penting pada beberapa aplikasi, khususnya sistem pakar dan sistem kendali fuzzy. Pada dasarnya, variabel linguistik adalah variabel yang nilainya berupa kata-kata atau kalimat pada suatu bahasa natural maupun artifisial. Sebagai contohnya, kecepatan adalah variabel linguistik jika kecepatan tersebut memakai nilai seperti lambat, cepat, sangat cepat, dan lain sebagainya. Konsep variabel linguistik diperkenalkan oleh Zadeh sebagai cara untuk memperkirakan sifat suatu fenomena yang cukup sulit atau kompleks untuk dinyatakan secara kuantitatif. Sebelum membahas definisi variabel linguistik, akan dibahas terlebih dahulu variabel fuzzy.

Variabel fuzzy dinyatakan dengan tiga pasangan terurut (X,U,R(X)) yang mana X adalah nama variabel, U adalah semesta pembicaraan, dan R(X) adalah himpunan fuzzy subset dari U yang merepresentasikan suatu nilai fuzzy yang dinyatakan oleh X.

Contoh 10.1

Didefinisikan suatu variabel dengan X = "tua", semesta pembicaraan U = {10,20,...80} yang mewakili nilai umur dan

$R(x) = \frac{0.1}{20} + \frac{0.2}{30} + \frac{0.4}{40} + \frac{0.5}{50} + \frac{0.8}{60} + \frac{1}{70} + \frac{1}{80}$ merupakan nilai fuzzy dari "tua".

Variabel linguistik adalah variabel yang tingkatannya lebih tinggi daripada variabel fuzzy. Variabel linguistik mempergunakan variabel fuzzy sebagai nilainya. Variabel linguistik direpresentasikan dengan sebuah *quintuple* (x, T(x), U, G, M) yang mana:

- X merupakan nama variabel
- T(x) merupakan himpunan x yakni himpunan semua nama dari nilai linguistik x yang mana setiap nilainya didefinisikan sebagai variabel fuzzy di U.

- G merupakan aturan sintaksis untuk membangkitkan nama dari nilai x
- M merupakan aturan semantik untuk menghubngkan setiap nilai x dengan artinya.

Contoh 10.2

Jika kecepatan diartikan sebagai variabel inguistik dengan U = [0.100], maka x = "kecepetan", himpunan istilah dari x dapat berupa:

T (kecepatan) = {sangat lambat, lambat, sedang, cepat,...}

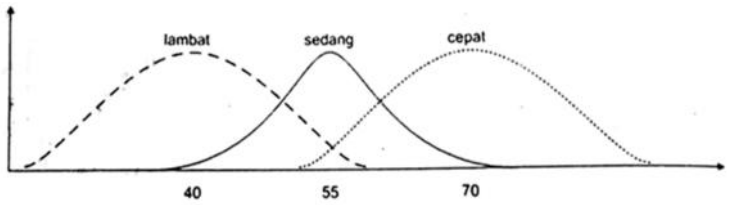
Dalam hal ini aturan sintaks untuk membangkitkan nama (label) dari elemen-elemen T(kecepatan) sangatlah intuitif. Aturan semantik M dapat didefinisikan sebagai berikut:

M(lambat) = himpunan fuzzy untuk "kecepatan di bawah 49 kilometer per jam (km/jam)" dengan fungsi keanggotaan μ_{lambat}

M(sedang) = himpunan fuzzy untuk "kecepatan sekitar 55 kilometer per jam (km/jam)" dengan fungsi keanggotaan μ_{sedang}

M(cepat) = himpunan fuzzy untuk "kecepatan di atas 70 kilometer per jam (km/jam)" dengan fungsi keanggotaan μ_{cepat}

Istilah-istilah tersebut dapat ditunjukkan dengan himpunan fuzzy yang memiliki fungsi keanggotaan seperti ditunjukkan pada gambar 10.1.



Gambar 10.1 Variabel linguistik "kecepatan" dengan tiga himpunan fuzzy sebagai nilainya

Hedge atau *modifier* linguistik adalah operator untuk memodifikasi arti suatu operan, atau lebih mudahnya, dari sebuah himpunan fuzzy A untuk membuat sebuah himpunan fuzzy baru h(A). sebagai contoh, pada himpunan fuzzy "sangat muda","sangat" merupakan *modifier* linguistik. Operator yang sering digunakan pada *modifier* linguistik adalah antara lain:

$$concentration = CON(A)$$

$$\mu_{CON(A)}(u) = (\mu_A(u))^2 \qquad (10.1)$$

$$dilation = DIL(A)$$

$$\mu_{DIL(A)}(u) = (\mu_A(u))^{1/2} \qquad (10.2)$$

10.2 Logika Fuzzy

Logika adalah dasar pemikiran. Logika klasik berkaitan dengan proposisi yang bisa bernilai benar (dengan nilai logika 1) atau salah (dengan nilai logika 0). Proposisi adalah kalimat yang dinyatakan dalam suatu bahasa dan dapat diekspresikan dalam bentuk umum:

x adalah P (10.3)

yang mana x adalah simbol dari suatu objek dan P merupakan predikat yang mencirikan sifat dari subjek tersebut. Contohnya: pensil adalah alat tulis.

Setiap proposisi A mempunyai lawannya yang disebut negasi dan dinotasikan sebagai $\bar{A}$. Proposisi dan negasinya memiliki nilai kebenaran yang berlawanan. Pada bagian ini akan dibahas dua hal penting dalam logika, yaitu operasi logika dan inferensi.

Operasi logika adalah fungsi dari dua proposisi dan didefinisikan dengan tabel kebenaran. Misalkan terdapat dua preposisi A dan B, yang mana masing-masing bisa bernilai benar dan salah. Empat operasi dasar logika adalah konjungsi ($\wedge$), disjungsi ($\vee$), implikasi (=>), dan ekuivalensi atau bidireksional ($\Leftrightarrow$) dan diinterpretasikan sebagai "A AND B" , "A OR B" , "IF A THEN B" ,

dan "A IF AND ONLY IF B". keempat operasi logika tersebut dirangkum pada tabel 10.1

Tabel 10.1 Nilai kebenaran operasi logika

Pernyataan A	Pernyataan B	Operasi Konjungsi (A∧B)	Operasi Disjungsi (A∨B)	Operasi Implikasi (A=>B)	Operasi Konjungsi (A⇔B)
Benar	Benar	Benar	Benar	Benar	Benar
Salah	Benar	Salah	Benar	Salah	Salah
Benar	Salah	Salah	Benar	Benar	Salah
Salah	Salah	Salah	Salah	Benar	benar

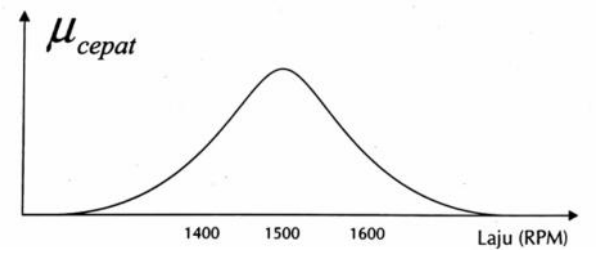
Hal yang juga penting dalam prosedur sistem logika adalah pemikiran yang mana dilakukan melalui aturan-aturan inferensi. Beberapa aturan inferensi yang penting adalah:

(A ∧ (A=>B)) => B (modus ponen)
(B ∧ (A=>B)) => $\bar{A}$ (modus tollens)
(A => B) ∧ (B => C)) => (A => C) (hipotesis silogisme) (10.4)

Hasil proposisi pada persamaan di atas selalu bernilai benar apapun nilai kebenaran A dan B. Hal itu dinamakan tautologi. Penggunaan persamaan di atas pada pemikiran cukup jelas, sebagai contoh, modus ponen diinterpretasikan sebagai: jika A benar, maka proposisi B benar. Modus ponen berkaitan dengan inferensi data arah maju. Perluasan pada logika fuzzy, modus ponen sering digunakan di sistem kendali fuzzy. Sedangkan modus tollens berkaitan dengan inferensi data arah baik yang biasanya digunakan di sistem pakar.

Logika fuzzy merupakan perluasan logika klasik dan teori himpunan. Nilai kebenarannya berupa variabel linguistik "benar". Tidak seperti pada logika klasik yang memiliki dua logika yaitu benar (1) dan salah (0), maka nilai kebenaran sebuah pernyataan atau proposisi pada logika fuzzy berada pada range interval [0,1]. Contoh sebuah proposisi "laju putaran motor listrik adalah cepat" dapat

dinyatakan dengan nilai kebenaran berupa himpunan fuzzy seperti ditunjukkan pada Gambar 10.2



Gambar 10.2 Himpunan fuzzy "cepat" untuk laju motor listrik

Jadi nilai kebenaran dari sebuah profesi "X adalah A" atau sederhananya disebut nilai kebenaran A, yang dinotasikan dengan v(A), yaitu sebuah titik pada interval [0,1] yang disebut sebagai nilai kebenaran numerik ata merupakan himpunan fuzzy pada interval [0,1] yang disebut nilai kebenaran linguistik.

Jika v(A) dan v(B) adalah nilai kebenaran numerik dari proposisi A dan B, maka

$$\begin{aligned} &v(NOT\quad A) = 1 - v(A) \\ &v(A\quad AND\quad B) = v(A) \wedge v(B) = \min\{v(A), v(B)\} \\ &v(A \Rightarrow B) = v(A) \wedge v(B) = \max\{1 - v(A), \min\{v(A), v(B)\}\} \end{aligned} \qquad (10.5)$$

10.3 Penarikan Kesimpulan Berdasarkan Pendekatan (*fuzzy reasorning*)

Aturan dasar inferensi pada logika klasik dengan dua nilai kebenaran adalah modus ponen, yakni kebenaran dari proposisi B ditarik dari kebenaran proposisi A dan kebenaran implikasi A=>B. sebagai contoh jika proposisi A adalah "tomat berwarna merah" dan B adalah "tomat masak". Jika benar bahwa "tomat berwarna merah" maka proposisi "tomat masak" juga benar. Konsep ini diilustrasikan sebagai berikut:

Premise 1 (kenyataan)	:	x adalah A
Premise 2 (aturan)	:	jika x adalah A maka y adalah B
Konsekuen (kesimpulan)		y adalah B

Pemikiran manusia pada umunya dilakukan dengan cara modus ponen dan diaplikasikan dengan cara pendekatan. Sebagai contoh, jika terdapat implikasi "jika tomat berwarna merah, maka tomat masak" dan terdapat kenyataan bahwa "tomat berwarna agak merah" maka dapat disimpulkan bahwa "tomat agak masak". Konsep tersebut diilustrasikan sebagai berikut:

Premise 1 (kenyataan)	:	x adalah A'
Premise 2 (aturan)	:	jika x adalah A maka y adalah B
Konsekuen (kesimpulan)		y adalah B'

yang mana, A' mendekati A dan B' mendekati B. jika A, A', B dan B' adalah himpunan fuzzy pada suatu semesta pembicaraan, maka prosedur inferensi tersebut dinamakan *approximate reasoning* atau *fuzzy reasoning* atau disebut *generalized modus ponen* (GMP).

Misalkan A, A' dan B' adalah himpunan fuzzy berrturut-turut di x, X dan Y. asumsikan bahwa implikasi A=>B dinyatakan dengan relasi fuzzy R di X x Y. maka himpunan fuzzy B' yang dihasilkan dari "x adalah A' " dan aturan fuzzy "jika x adalah A maka y adalah B" didefinisikan sebagai:

$$\mu_{B'}(y) = \max_{x \in X} \min\{\mu_{A'}(x), \mu_R(x,y)\}$$

$$= v_x(\mu_{A'}(x) \wedge \mu_R(x,y)\} \qquad (10.6)$$

Atau ekuivalen dengan:

$$B' = A' \circ R = A' \circ (A \rightarrow B) \qquad (10.7)$$

Prosedur inferensi fuzzy reasorning dapat digunakan untuk menarik kesimpulan, jika implikasi fuzzy A=>B didefinisikan sebagai relasi fuzzy biner yang sesuai.

Berikutnya akan dibahas mengenai, aspek komputasi fuzzy reasorning dan memperluas pembahasan pada situasi dimana terdapat banyak aturan fuzzy dengan banyak anteseden yang mempengaruhi tingkah laku sistem. Namun, pembahasan akan dibatasi pada fungsi implikasi fuzzy menurut Mamdani dan

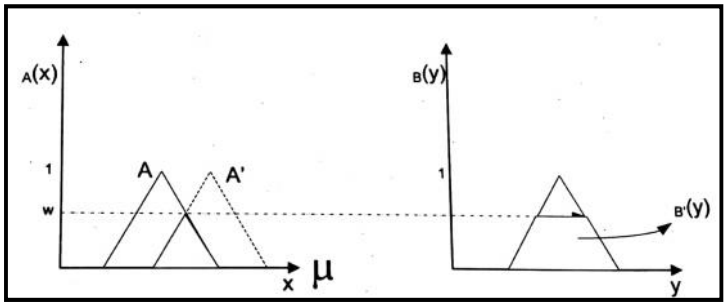
komposisi max-min. hal tersebut didorong oleh luasnya aplikasi dan interpretasi grafis yang mudah.

1. Aturan tunggal dengan anteseden tunggal

Hal ini merupakan tipe yang paling sederhana, dan persamaan yang digunakan adalah persamaan 10.6. penyederhanaan persamaan 10.6 diperoleh sebagai berikut:

$$\mu_{B'}(y) = \{\mu_{A'}(x), \mu_{A}(x)\} \wedge \mu_{B}(y)$$
$$= w \wedge \mu_{B}(y) \qquad (10.8)$$

Dengan kata lain, pertama-tama ditentukan nilai w sebagai nilai maksimum $\mu_{A'}$ (x) $\wedge$ μ_{A} (x) (daerah yang diarsir pada gambar 10.3), maka fungsi keanggotaan himpunan fuzzy B' ditentukan dengan memotong fungsi keanggotan B dengan w, sebagaimana ditunjukkan oleh daerah bagian konsekuen yanng diarsir pada gambar 10.3. secar intuisi, w adalah hasil pengukuran derajat kepercayaan bagian anteseden dari sebuah aturan, hasil pengukuran ini kemudian dirambatkan oleh aturan if-then dan derajat kepercayaan hasil atau derajat keanggotaan bagian konsekuen (B' pada gambar 10.3) tidak lebih tinggi dari w.



Gambar 10.3 Inferensi fuzzy dengan aturan tunggal dan anteseden tunggal

2. Aturan tunggal dengan beberapa anteseden

Sebuah aturan if-then fuzzy, dengan dua anteseden biasanya ditulis sebagai "x jika x adalah A dan y adalah B, maka z adalah

C". atau dapat direpresentasikan dengan generalized modus ponens (GMP) sebagai berikut:

Premise 1 (kenyataan)	: x adalah A' dan y adalah B'
Premise 2 (aturan)	: jika x adalah A maka y adalah B maka z
Konsekuen (kesimpulan)	z adalah C'

Aturan fuzzy pada premise 2 dapat dinyatakan dalam bentuk sederhana "A x B => C". secara intuisi, aturan fuzzy tersebut dapat ditransformasikan dalam bentuk relasi fuzzy ternari Rm maka berdasarkan fungsi implikasi fuzzy Mamdani, sebagai berikut:

$$R_m(A,B,C) = (AxBxc) = \int_{XxYxZ} «_A(x) \wedge «_B(y) \wedge «_C(z) \,/\, (x,y,z) \tag{10.9}$$

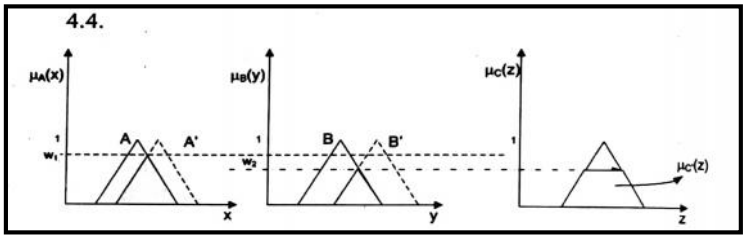
Maka C' dapat dinyatakan sebagai:

$$c' = (A'xB) \circ (AxB \rightarrow C) \tag{10.10}$$

Sehingga diperoleh:

$$\begin{aligned} «_{c'}(Z) &= V_{x,y}\{«_{A'}(x) \wedge «_{B'}(y)\} \wedge \{«_A(x) \wedge «_B(y) \wedge «_C(z)\} \\ &= V_{x,y}\{«_{A'}(x) \wedge «_{B'}(y)\} \wedge \{«_A(x) \wedge «_B(y)\} \wedge «_C(z) \\ &= \underbrace{\{V_x\{«_{A'}(x) \wedge «_A(x)\}\}}_{W_1} \wedge \underbrace{\{V_y\{«_B(y) \wedge «_{B'}(y)\}}_{W_2} \wedge «_C(z) \\ &= \underbrace{(w_1 \wedge w_2)}_{firing\,strength} \wedge «_c(z) \end{aligned} \tag{10.11}$$

Secara grafis persamaan 10.11 dapat ditunjukkan seperti Gambar 10.4.



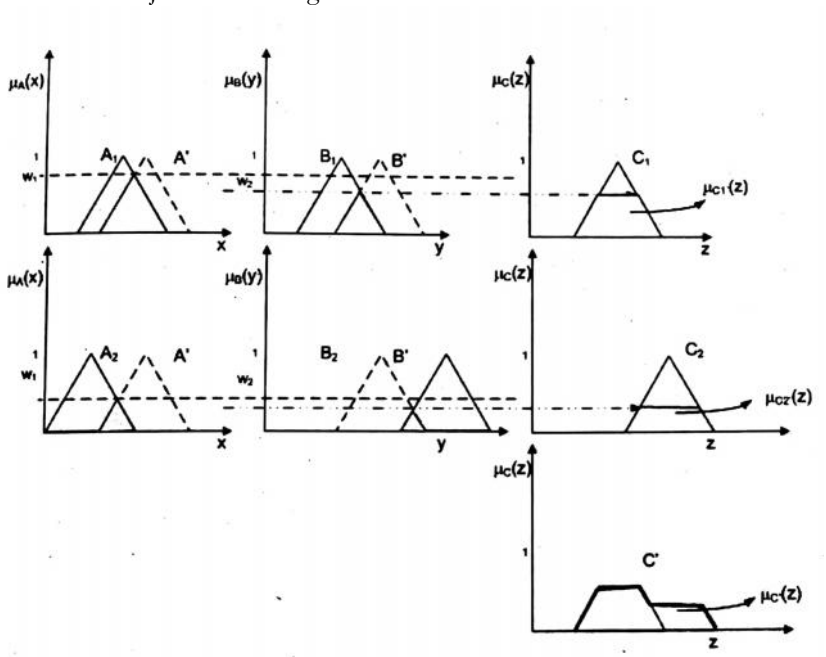
Gambar 10.4 inferensi fuzzy dengan aturan tunggal dan dua anteseden

3. Beberapa aturan dengan beberapa anteseden

Intepretasi banyak aturan biasanya dianggap sebagai gabungan relasi fuzzy yang berkaitan dengan aturan fuzzy tersebut. Oleh karena itu, untuk masalah GMP dapat dituliskan sebagai :

Premise 1 (kenyataan)	: x adalah A' dan y adalah B'
Premise 2 (aturan)	: jika x adalah A1 maka y adalah B1 maka z adalah C2,
Konsekuen (kesimpulan)	z adalah C'

Prosedur inferensi untuk menentukan himpunan fuzzy keluaran C' ditunjukkan oleh gambar 10.5



Gambar 10.5 inferensi fuzzy dengan beberapa aturan yang memiliki beberapa anteseden

C. Rangkuman

1. Variabel linguistik merupakan konsep penting pada logika fuzzy dan memegang peranan penting pada beberapa aplikasi, khususnya sistem pakar dan sistem kendali fuzzy. Pada dasarnya, variabel linguistik adalah variabel yang nilainya berupa kata-kata atau kalimat pada suatu bahasa natural maupun artifisial.
2. Beberapa aturan inferensi yang penting adalah:
   $(A \wedge (A=>B)) => B$ (modus ponen)
   $(B \wedge (A=>B)) => \bar{A}$ (modeus tollens)
   $(A => B) \wedge (B => C)) => (A => C)$ (hipotesis silogisme)
3. Hedge atau *modifier* linguistik adalah operator untuk memodifikasi arti suatu operan, atau lebih mudahnya, dari sebuah himpunan fuzzy A untuk membuat sebuah himpunan fuzzy baru h(A).
4. Beberapa aturan dalam penarikan kesimpulan berdasarkan pendekatan (fuzzy reasorning).
   1. Aturan tunggal dengan anteseden tunggal
   2. Aturan tunggal dengan beberapa antiseden
   3. Beberapa aturan dengan beberapa anteseden

D. Tugas

1. Carilah referensi lain mengenai logika fuzzy!
2. Buatlah kesimpulan mengenai materi yang telah dibahas pada bab 10 ini!

E. Tes Formatif

Carilah referensi lain yang membahas mengenai logika fuzzy!

1. Jelaskan pengertian dari variabel linguistik!
2. Apa yang dimaksud dengan logika fuzzy?
3. Apa kelebihan dan kekurangan dari logika fuzzy?
4. Apa yang dimaksud dengan fuzzyfikasi dengan defuzzyfikasi?

BAB 11

AGORITMA GENETIK

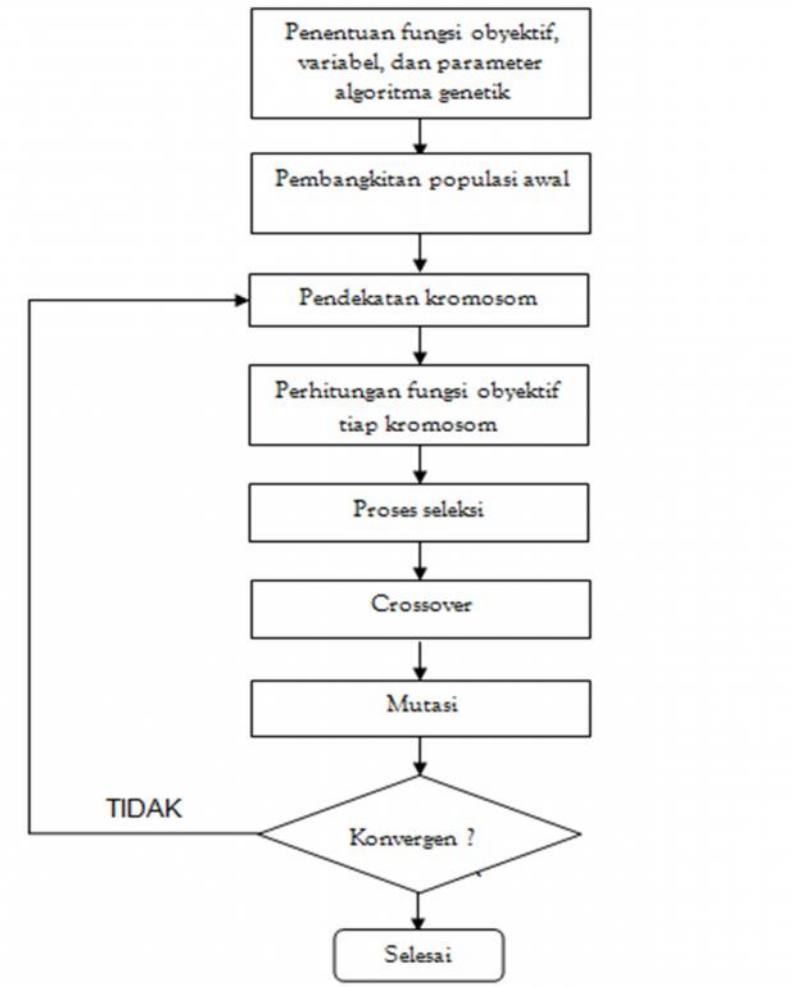
A. Tujuan Instruksional Khusus

Setelah mempelajari uraian materi, diharapkan mahasiswa dapat:

1. Menjelaskan defenisi dari algoritma genetik
2. Menjelaskan mengenai fungsi objektif, representasi kromosom, populasi, seleksi, algoritma crossover, algoritma mutasi, elitisme, pengkodean dengan bilangan rill, inisialisasi parameter algoritma genetik dan penanganan optimasi dengan batasan menggunakan algoritma genetik

B. Uraian Materi

11.1 Definisi Algoritma Genetik



Gambar 11.1 Diagram alir algoritma genetik

Algoritma genetik menggunakan tiga aturan utama pada setiap iterasi untuk menghasilkan generasi baru dari populasi saat ini, yaitu:

1. Aturan seleksi untuk memilih individu sebagai induk yang berkontribusi pada terbentuknya generasi baru

2. Aturan crossover untuk menggabngkan dua induk untuk membentuk keturunan pada generasi baru.
3. Aturan mutasi yang mengaplikasikan perubahan acak pada individu induk untuk membentuk keturnan.

Sebagaimana halnya algoritma optimasi yang lain, algoritma genetik dimulai dengan menentukan fungsi yang akan dioptimasi atau disebut fungsi objektif dan menentukan variabel optimasi. Algoritma genetik akan berhenti setelah menguji kondisi konvergensi. Dalam proses iterasinya, algoritma genetik akan berhenti setelah menguji kondisi konvergensi.

Proses iterasi dimulai dengan membangkitkan populasi awal. Selanjutnya individu yang merupakan kandidat solusi dan berupa bilangan biner akan dikodekan sehingga bisa dihitung nilai fungsi obyektif atau nilai kebugaran (fitness) bagi masing-masing individu tersebut. Berdasarkan nilai fitness tersebut, individu-individu dalam populasi akan memiliki peluang yang lebih besar untuk terpilih sebagai induk untuk menghasilkan generasi baru.

Selanjutnya induk-induk tersebut akan melakukan perkawinan silang (*crossover*) dan mutasi untuk menghasilkan keturunan. Dengan adanya keturunan-keturunan tersebut, terbentuklah populasi bar yang selanjutnya akan dievaluasi nilai-nilai fitnessnya untuk mengetahui apakah proses algoritma genetik sudah mencapai konvergensi atau belum. Jika kondisi konvergensi sudah tercapai maka proses iterasi akan dihentikan dan ini berarti solusi optimal mungkin sudah didapatkan, jika kondisi konvergen belum tercapai maka iterasi akan terus diulangi.

Tabel 11.1 Perbedaan algoritma standar dengan algoritma genetik

Algoritma standar	Algoritma genetik
Membangkitkan satu titik tunggal setiap iterasi, sehingga titik tersebut akan mencapai solusi optimal	Membangkitkan populasi titik-titik solusi pada setiap iterasi, sehingga titik-titik pada populasi tersebut mencapai solusi-solusi optimal

Memilih titik pada urutan berikutnya dengan komputasi yang ditentukan	Memilih populasi berikutnya dengan komputasi yang melibatkan pilihan acak

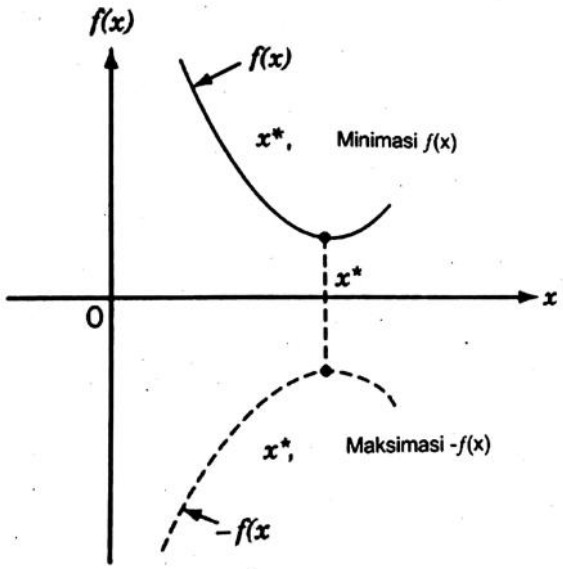
11.2 Fungsi Objektif

Optimasi adalah proses pengaturan input dan karakteristik suatu alat, proses, atau eksperimen untuk menghasilkan output yang maksimum atau minimum. Input terdiri atas variabel, sedangkan proses atau fungsi disebut sebagai fungsi objektif, fungsi biaya (cost), atau fungsi fitness, dan output proses adalah nilai biaya atau nilai fitness. Fungsi objektif yang digunakan pada algoritma genetik adalah representasi yang akan dimaksimalkan, yang dinyatakan dengan fungsi fitness. Sedangkan jika masalah yang akan diselesaikan adalah masalah minimasi maka fungsi objektif perlu dimodifikasi yaitu dikalikan dengan -1 sehingga masalah yang akan dipecahkan menjadi masalah maksimasi. Hal ini ditunjukkan pada Gambar 11.2 pada algoritma genetik, masalah yang akan diselesaikan adalah masalah maksimasi karena fungsi yang menjadi tujuan adalah fungsi fitness.

Contoh 11.1

Masalah maksimasi dari dua fungsi yang dinyatakan pada persamaan 11.1 memiliki dua variabel yaitu x dan y. gambar grafik tiga dimensi fungsi tersebut ditunjukkan oleh gambar 11.3 sedangkan gambar contournya ditunjukkan pada gambar 11.4. pada gambar tersebut dapat dilihat adanya beberapa titik maksimum lokal dan satu titik maksimum global. Tujuan penyelesaian masalah optimasi adalah mendapatkan solusi yaitu titik maksimum global. Dengan algoritma genetik yang memiliki karakteristik acak dalam mencari solusi optimal, dimungkinkan untuk terhindar dari konvergensi pada titik optimum lokal.

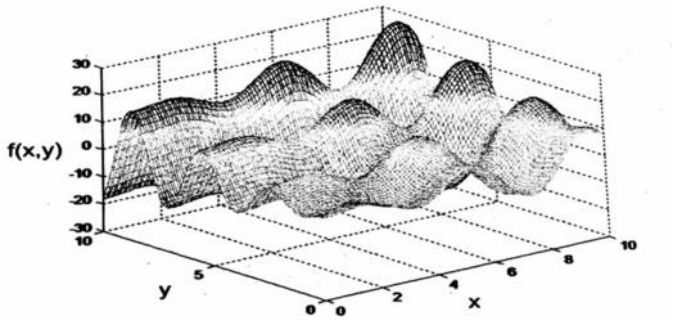
Carilah nilai maksimum fungsi: f(x,y) 1.2xsinx(1.5)-2ysin(sy)
Untuk: $0 \leq x \leq 10$ dan $0 \leq y \leq 10$ (11.1)



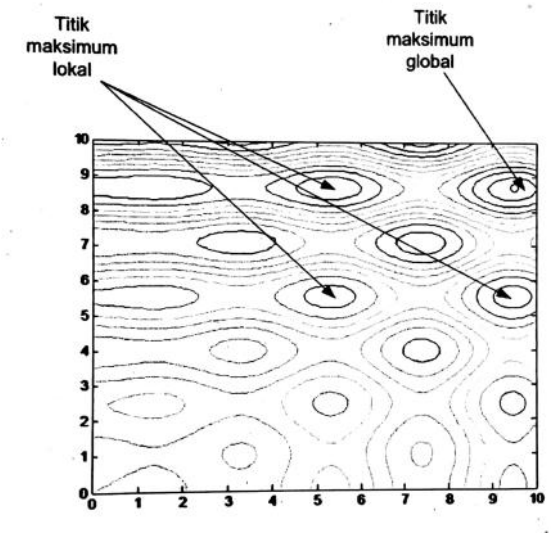
Gambar 11.2 Mencari nilai maksimum suatu masalah minimasi dengan cara mengalikan dengan -1

11.3 Representasi Kromosom

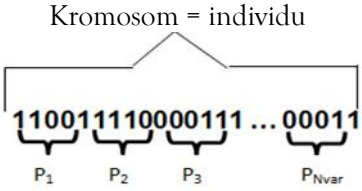
Kromosom adalah string yang berisi kode/sandi yang mungkin untuk parameter/variabel yang akan diopttimasikan. Kromosom ini umumnya direpresentasikan sebagai string biner. Salah satu tantangan utama dalam merancang suat algoritma genetika adalah menemukan cara yang cocok untuk menyandikan parameter.



Gambar 11.3 Grafik permukaan fungsi f(x,y) pada contoh 11.1



Gambar 11.4 Grafik *contour* fungsi objektif pada contoh 11.1



Gambar 11.5 Kromosom

Panjang string biner ditentukan berdasarkan tingkat kepresisian solusi yang diinginkan. Jika dalam suatu masalah optimasi terhadap N_{var} variabel yaitu p_1, p_2, p_3,...,p_{Nvar} maka kromosom dapat ditulis sebagai vektor baris dengan elemen sebanyak N_{var} seperti pada persamaan 11.2

$$kromosom = [p_1, p_2, p_y ... p_{N\,var}] \qquad (11.2)$$

Panjang kromosom pada (L) ditentukan berdasarkan nilai presisi yang diinginkan untuk sebuah variabel, dan selanjutnya panjang string biner untuk setiap variabel p_1, p_2, p_3,...,p_{Nvar} dapat ditentukan dengan persamaan:

$$l = \log_2 \left((p_{\max} - p_{\min}) x 10^{nilai\_presisi} + 1 \right), \text{ dibulatkan ke atas} \qquad (11.3)$$

Contoh 11.2

Pada contoh 11.1 diketahui masing-masing variabel memiliki rentang nilai [0,10]; jika diinginkan kepresisian 2 angka dibelakang tanda desimal (koma) maka panjang kromosom yang dibutuhkan adalah:

$L = \log_2((10 - 10) \times 10^2 + 1) = 9.9672 \approx 10bit$

Dalam proses pengkodean variabel menggunakan kode biner, setiap nilai variabel akan dikuatisasi dengan persamaan 11.4.

$$\text{Kuantitasi} = \frac{Pmax - Pmin}{2^L - 1} \qquad (11.4)$$

Contoh 11.3

Nilai kuantutasi untuk soal 11.2 adalah:

$$\text{Kuantitasi} = \frac{10-0}{2^{10}-1} = 0.0098$$

Nilai desimal dari suatu level kuantisasi setiap nilai variabel p, dihitung dengan rumus:

$$\text{desimal}_{string} = \frac{P - Pmin}{\text{kuantisasi}} \qquad (11.5)$$

Contoh 11.4

Misalkan pada contoh 11.3 , diketahui nilai variabel x adalah 5, sehingga nilai desimal untuk x=5 tersebut adalah:

$$\text{desimal}_{string} = \frac{5-0}{0.0098} = 510.2041 \approx 511$$

Pengkodean biner untuk setiap variabel dapat mengikuti persamaan sebagai berikut.

$$\text{biner}_p = \text{nilai\_biner}(\text{desimal}_{string}) \qquad (11.6)$$

Contoh 11.5

Hasil yang diperoleh pada contoh 9.4 dapat dibinerkan menjadi 0111111111

Dalam proses operasinya algoritma genetik menggunakan kode biner pada kromosom-kromosomnya, namun dalam perhitungan nilai fitness dilakukan terhadap nilai rill dari variabel yang diwakili oleh suatu kromosom. Oleh karena itu, perlu pengkodean kromosom ke nilai rillnya. Pengkodean kromosom tersebut dapat mengikuti persamaan:

$$rill_p = (desimal_{string} * kuantitasi) + p_{min} \quad (11.7)$$

Contoh 11.6

Misalkan string biner pada contoh 9.5 akan memiliki nilai $desimal_{string}$ = 511, sehingga nilai rillnya dapat dihitung dengan:

$rill_p$ = (511*0.0098) + = 5.0078

Dari hasil di atas dibandingkan dengan nilai x yang sesungguhnya pada contoh 11.4 yaitu x = 5, terdapat nilai *error* kantitasi sebesar 5.0078 – 5 = 0.0078. Namun perlu diketahui bahwa berdasarkan nilai presisi yang ditentukan di awal adalah 2 angka di belakang tanda desimal. Sehingga dalam kasus ini nilai *error* kuantitasi tersebut tidak penting.

Setelah nilai rill dihitung dalam pengodean kromosom, maka setiap kromosom memiliki nilai fitness hasil evaluasi fungsi f terhadap p_1, p_2, p_3,...,P_{nvar} yang dihitung dengan persamaan:

$$Nilai_{fitness} = f(kromosom) = f(p_1, p_2, p_3,...,P_{nvar}) \quad (11.8)$$

Contoh 11.7

Misalkan untuk fungsi fitness pada contoh 11.1 diketahui sebuah kromosom yang telah dikodekan sehingga diperoleh nilai x = 5 dan y = 3 maka nilai fitness dapat dihitung sebagai berikut:

f(x,y) = 1.2xsin(1.5x) – (2ysin(2y)

f(5,3) = 1.2x5xsin(4x5) – 2x3xsin(2x3) = 7.3045

11.4 Populasi

Algoritma genetik dimulai dengan kumpulan kromosom-kromosom yang disebut sebagai populasi. Jika jumlah individu adalah N_{pop} dan ukuran bit-bit kromosom adalah N_{bit} maka populasi awak dapat dibangkitkan dengan cara membuat bilangan sebanyak N_{pop} x N_{bit} secara acak dalam rentang [0,1] yang kemudian dibutuhkan atau dituliskan sebagai:

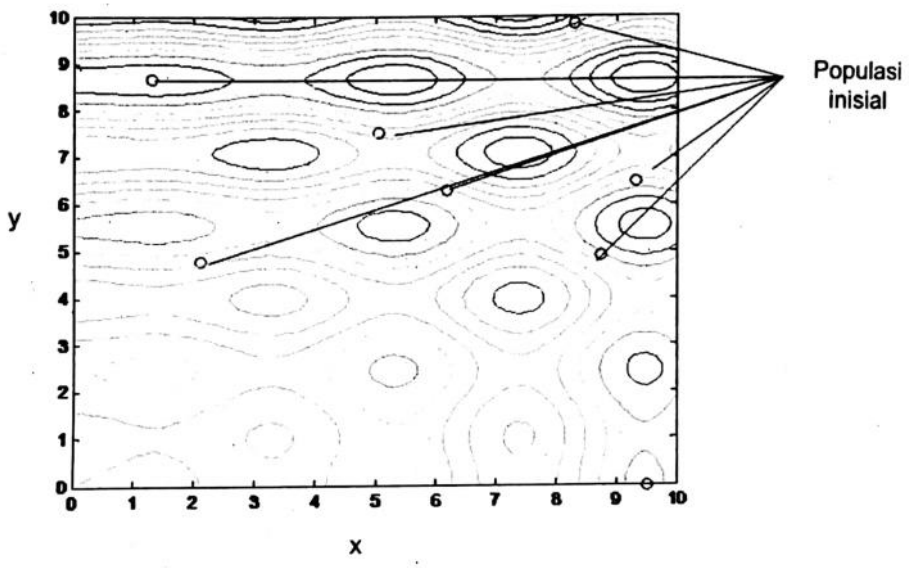
$$pop = round\ [rand(N_{pop}, N_{bit})] \qquad (11.9)$$

sehingga akan diperoleh populasi awal yang direpresentasikan dengan matriks berukuran N_{pop} x N_{bit} yang berisi biner 0 atau 1. Setiap baris matriks tersebut merupakan sebuah kromosom.

Untuk masalah optimasi pada conth 11.1 dapat dibuat populasi awal sebagaimana pada tabel 11.2. kromosom-kromosom tersusun atas 20 bit karena setiap variabel dikodekan dengan 10 bit. Kolom ke-3 dan ke-4 adalah nilai variabel hasil pengkodean masing-masing kromosom.

Tabel 11.2 populasi awal dengan individu sebanyak delapan

Individu ke-	Kromosom	x	y	fitness
1	11010011110010000101	8.3006	9.8392	-15.6536
2	11110010101101111101	1.3034	8.6632	18.7567
3	00110101111000000100	9.506	0	11.3226
4	11101110001001111000	8.7514	4.9098	9.3619
5	11111011001101110100	2.107	4.7726	1.0989
6	00000000000111110101	5.0568	7.5264	-3.3212
7	01111001111100000000	9.3296	6.468	6.4082
8	10100101001010000100	6.1936	6.3112	0.2889



Gambar 11.6 posisi individu-individu dalam populasi awal

11.5 Seleksi

Dari individu-individu yang ada dalam suatu popularitas, perlu dipilih individu-individu terbaik yang dapat melakukan perkawinan untuk menghasilkan individu baru. Seleksi bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit. Masing-masing individu dalam populasi akan menerima probabilitas reproduksi sebanding dengan nilai fitnessnya. Namun dalam kenyataan yang terjadi di alam, tidak semua anggota populasi akan digantikan oleh keturunan yang baru. Beberapa individu yang memiliki nilai fitness yang baik akan bertahan sampai pada generasi berikutnya. Oleh karena itu dalam algoritma genetik juga dilakukan proses untuk melestarikan individu-individu terbaiknya dengan cara memperkenalkan laju seleksi, X_{rate}, X_{rate} menggambarkan proposisi N_{pop} yang bertahan pada generasi berikutnya yang dirumuskan:

$$N_{keep} = X_{rate} N_{pop}$$

Umumya X_{rate} dipilih sebesar 50%. Jika dalam populasi terdapat 8 individu, maka nilai X_{rate} tersebut sebanyak 4 individu

akan dipertahankan dan dilakukan seleksi untuk memiliki induk dari keempat individu tersebut.

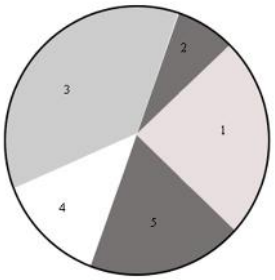
Ada berbagai macam algoritma seleksi yang dapat dipakai dalam algoritma genetik. Diantaranya dijelaskan sebagai berikut:

- Pembentukan pasangan dari atas ke bawah
  Dimulai dari kromosom paling atas dalam populasi ke bawah sampai individu sejumlah N_{keep}, Algoritma ini memasangkan kromosom pada baris ganji dengan kromosom pada baris genap. Misalkan terdapat 4 N_{keep} kromosom maka pasangannya adalah (1,2) dan (3,4). Meskipun tidak menggambarkan proses seleksi alamiah, namun metode ini cukup mudah diprogram.
- Pembentukan pasangan secara acak
  Metode ini menggunakan bilangan acak yang dibangkitkan untuk memilih kromosom. Dua pasang induk diperoleh dengan cara:

  $$ibu = \text{ceil}(N_{keep} * \text{rand}(1,N_{keep}))$$
  $$bapak = \text{ceil}(N_{keep} * \text{rand}(1,N_{keep})) \quad (11.11)$$

  dengan fungsi ceil untuk membulatkan ke bilangan bulat yang tepat di atasnya.

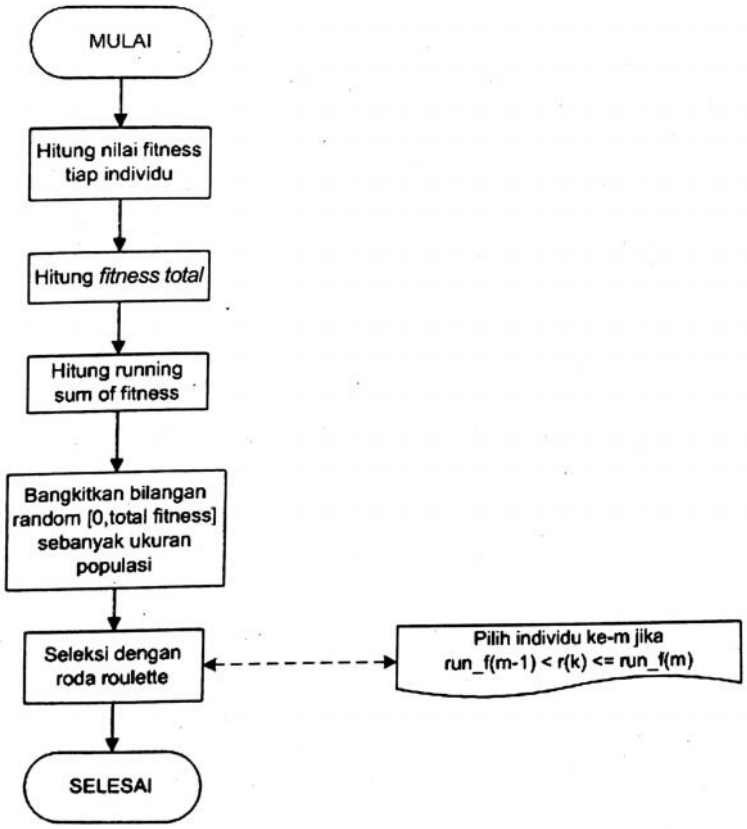
- Pembentukan pasangan terbobot secara acak
  Pada metode ini, dibangkitkan probabilitas seleksi yang sebanding dengan nilai fitness masing-masing kromosom. Metode ini juga dikenal sebagai metode seleksi roda roullete. Metode ini ditunjukkan pada Gambar 11.7.



Individu ke	Nilai fitness
1	25
2	5
3	40
4	10
5	20

Gambar 11.7 Metode seleksi roda *roulette*

Dalam algoritma roulette wheel ini, setiap individu dalam populasi dialokasikan ke dalam potongan bususr roda, yang mana ukuran busur tersebut sebanding dengan nilai fitnessnya. Sebuah pointer/jarum penunjuk diputar (dengan kata lain bilangan acak dibangkitkan) dan individu yang ditunjuk oleh pointer tersebut akan dipilih. Proses ini akan diulangi sampai sejumlah individu terpilih. Dalam metode ini, peluang setiap individu untuk terseleksi tergantng pada nilai fitnessnya. Salah satu masalah yang muncul dalam metode ini adalah individu akan terpilih beberapa kali dalam jumlah yang tidak menentu sehingga tidak ada jaminan bahwa individu yang lebih besar fitnessnya akan diwakili dalam generasi berikutnya. Proses seleksi dengan roda roulette dapat digambarkan dengan diagram alir pada Gambar 11.8.



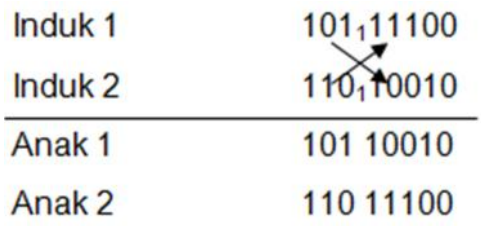
Gambar 11.8 Seleksi dengan metode roda roulette

- Seleksi dengan turnamen

  Dalam metode ini, kromosom dipilih secara acak dari N_{keep} dan kromosom dengan nilai fitness tertinggi menjadi induk dalam menghasilkan keturunan. Pemilihan diulangi secara acak sampai sejumlah induk diperoleh. Metode ini lebih mendekati proses seleksi alamiah karena tidak ada proses pergantian. Metode seleksi turnamen ini banyak digunakan pada aplikasi dengan jumlah populasi yang besar, karena jika digunakan proses seleksi yang menggunakan pengurutan akan membutuhkan waktu komputasi yang lama.

11.6 Algoritma Crossover

Setelah individu-individu terpilih dalam proses seleksi, selanjutnya pada individu-individu tersebut akan dilakukan perkawinan silang atau disebut *crossover*. *Crossover* yang paling dasar adalah *crossover* satu titi (*single point crossover*), yaitu sebuah titik pada string kromosom dipilih dan string disilangkan/ditukar pada titik tersebut.

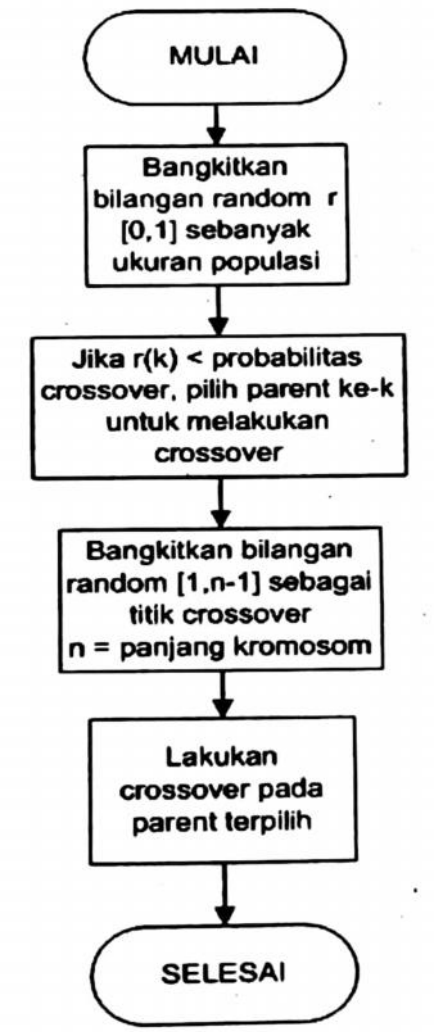


Gambar 11.9 Crossover satu titik

Pada penyilangan satu titik, posisi penyilangan k (k = 1,2,...,N-1) dengan N = panjang kromosom, ditentukan secara acak. Kemudian kita tukarkan bagian kanan titik potong dari kedua induk kromosom tersebut untuk menghasilkan kromosom anak. Proses *crossover* satu titik dapat digambarkan dengan diagram alir pada Gambar 11.10.

Multipoints crossover adalah perluasan dari *crossover* satu titik. Pada *multipoints crossover* ini dipilih beberapa titik *crossover* secara acak, lalu bagian string diantara titik-titik ini dipertemukan antar dua individu yang melakukan *crossover* dua titik yang ditunjukkan pada Gambar 11.10.

Parameter penting yang menentukan proses *crossover* adalah probabilitas (pc). Nilai pc ini akan menentukan peluang terjadinya *crossover* dalam suatu perkawinan. Semakin besar nilai pc maka akan semakin banyak induk-induk yang terpilih untuk melakukan *crossover* dan semakin banyak bervariasi keturunan yang dihasilkan.



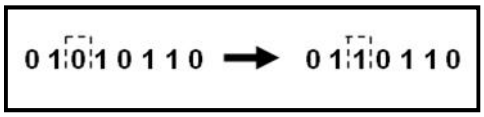
Gambar 11.10 Diagram alir crossover satu titik

Induk 1	10_1111_1100
Induk 2	11_1010_1010
Anak 1	10 010 100
Anak 2	11 111 010

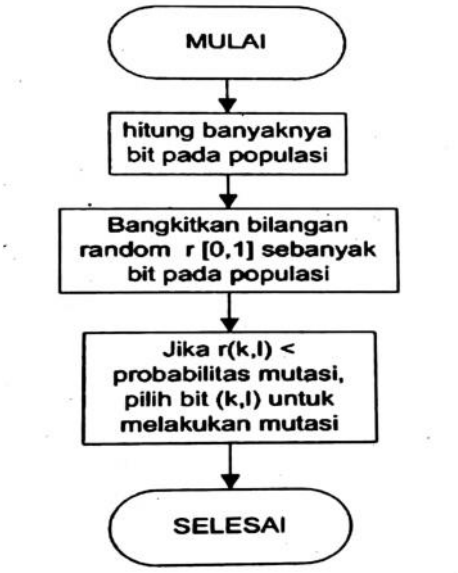
Gambar 11.11 Crossover dua titik

11.7 Algoritma Mutasi

Jika algoritma menggunakan kode biner dalam kromosomnya, maka hanya ada satu cara mutasi. Untuk setiap unit bit, dibangkitkan bilangan acak. Jika bilangan acak tersebut lebih kecil dari probabilitas mutasi, maka baik nilai bit tersebut, yakni jika bit = 1 akan diubah menjadi bit = 0, dan sebaliknya. Pada mutasi ini sangat dimungkinkan memunculkan kromosom baru yang semula belum muncul dalam populasi awal. Contoh mutasi ditunjukkan pada Gambar 11.12.



Gambar 11.12 Contoh proses mutasi



Gambar 11.13 contoh proses mutasi

Pada mutasi ada satu parameter yang sangat penting yaitu probabilitas mutasi (pm). Probabilitas mulai menunjukkan persentase jumlah total bit pada populasi yang akan mengalami mutasi. Untuk melakukann mutasi, terlebih dahulu harus dihitung

jumlah total bit pada populasi tersebut. Kemudian bangkitkan bilangan random yang akan menentukan posisi mana yang akan dimutasi (bit ke berapa pada kromosom ke berapa). Misal-kan ukuran populasi (popize = 100), setiap kromosom memiliki panjang 20 bit, maka total bit adalah 100x20 = 2000 bit. Jika peluang mutasi (p_m = 0,01), berarti bahwa diharapkan ada (1/100) x 2000 = 20 bit akan mengalami mutasi. Proses mutasi dapat digambarkan dengan diagram alir pada gambar 11.12.

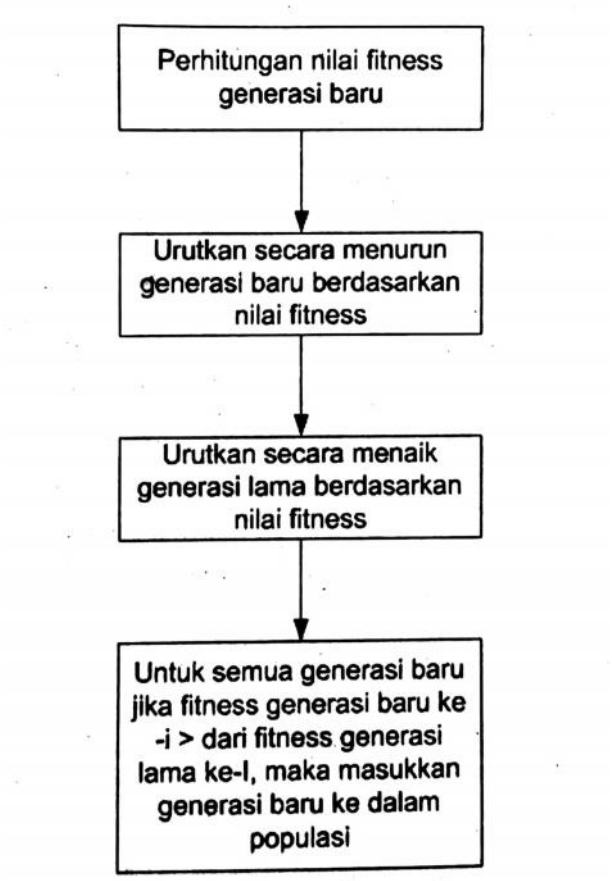
Probabilitas mutasi dipilih nilai yang kecil dan tetap selam proses algoritma genetik. Namun demikian, dalam algoritma dasar yang telah dimodifikasi, nilai probabilistik mutasi dapat diubah selama proses algoritma berjalan, dimulai dengan nilai yang agak besar dan dikurangi secara bertahap selama proses iterasi. Hal ini memungkinkan algoritma genetik mencari solusi potensial pada daerah yang luas dipermulaan iterasi dan kemudian pencarian solusi akan terjadi didaerah yang lebih sempit, saat telah mendekati konvergen.

11.8 Elitisme

Dengan adanya proses *crossover* dan mutasi, terdapat risiko hilangnya solusi optimum karena tidak ada jaminan bahwa operasi ini akan melindungi fitness. Untuk mengatasi hal ini individu-individu terbaik dalam populasi akan diselamatkan terlebih dahulu sebelum operasi *crossover* dan mutasi dilakukan. Strategi ini dinamakan elitisme. Setelah populasi yang baru terbentuk dan dievaluasi, akan diperiksa apakah struktur populasi ini sudah mengakomodasi individu yang tadi diselamatkan. Jika belum, maka salinan dari individu yang tadi diselamatkan akan diselipkan kembali ke dalam populasi tersebut, biasanya untuk menggantikan individu yang paling kecil nilai fitnessnya.

Cara yang lain adalah dengan membentuk generasi baru dari individu-individu lama yang fitnessnya tinggi dan dari individu-individu baru yang fitnessnya lebih tinggi dari individu yang lama. Langkah pertama adalah mengecek apakah nilai fitness individu baru lebih besar dari nilai fitness individu yang ada pada populasi.

Jika demikian maka individu baru akan dimasukkan ke dalam populasi baru. Metode ini dilakukan dengan mengikuti diagram alir pada Gambar 11.14.

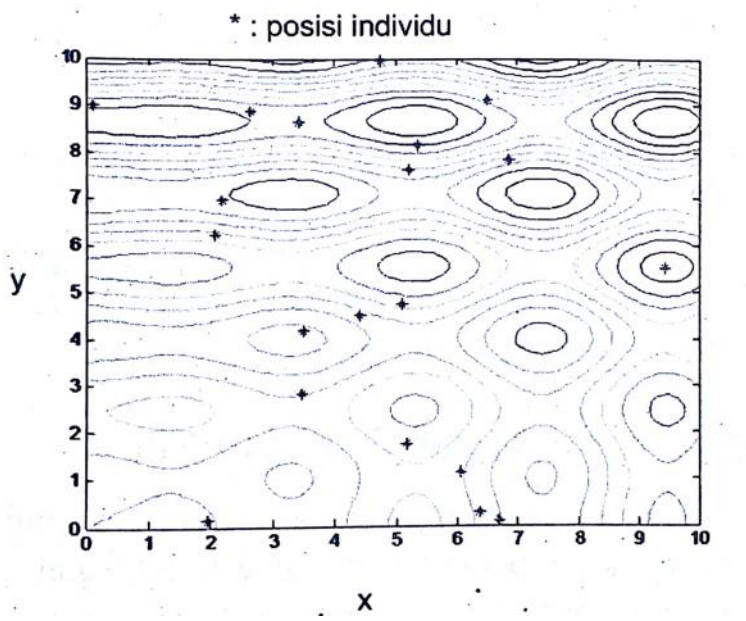


Gambar 11.14 Diagram alir proses eliminasi dengan elitisme

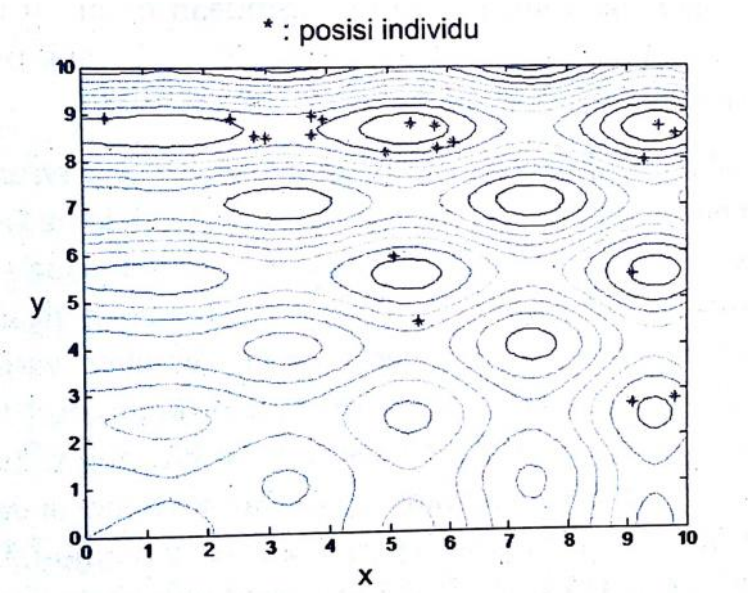
Contoh 11.8

Masalah optimasi pada contoh 11.1 diselesaikan dengan algoritma genetik. Parameter yang digunakan dalam program ini adalah ukuran populasi = 20; probabilitas *crossover* = 0.95; probabilitas mutasi = 0.50; dan maksimum generasi = 200. Gambar 11.15 mennjukkan sebaran individu dalam populasi inisial. Gambar 11.16 menunjukkan sebaran individu pada generasi ke-200, yang

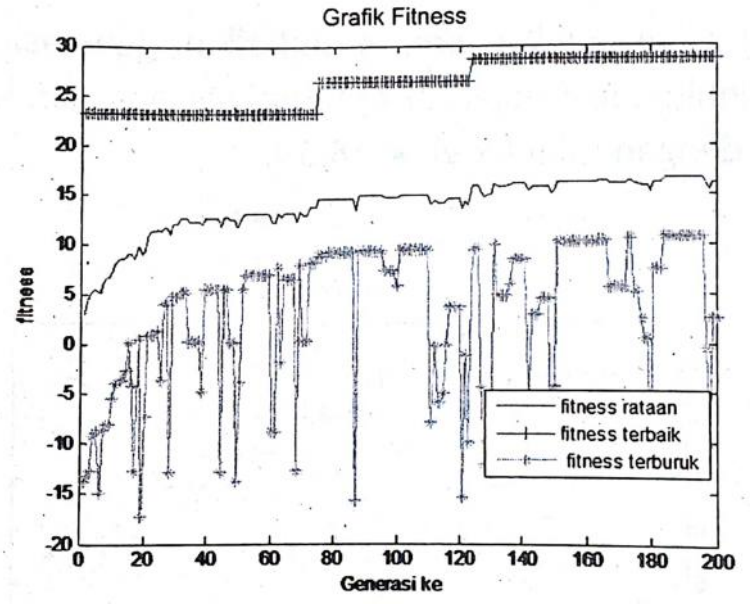
sudah mengumpul pada titik-titik maksimum lokal dan ada juga yang mendeteksi posisi maksimum global. Gambar 11.17 menunjukkan grafik fitness selama iterasi meliputi fitness maksimum, rata-rata fitness, dan fitness minimum. Dari grafik tersebut dapat dilihat adanya kenaikan nilai fitness populasi yang menunjukkan proses perbaikan generasi ke generasi. Algoritma genetik solusi masalah optimasi tersebut adalah x = 9.52 dan y = 8.64 dengan nilai f(x,y) = 28,59.



Gambar 11.15 Sebaran individu dalam populasi inisial



Gambar 11.16 Sebaran individu pada generasi ke-200

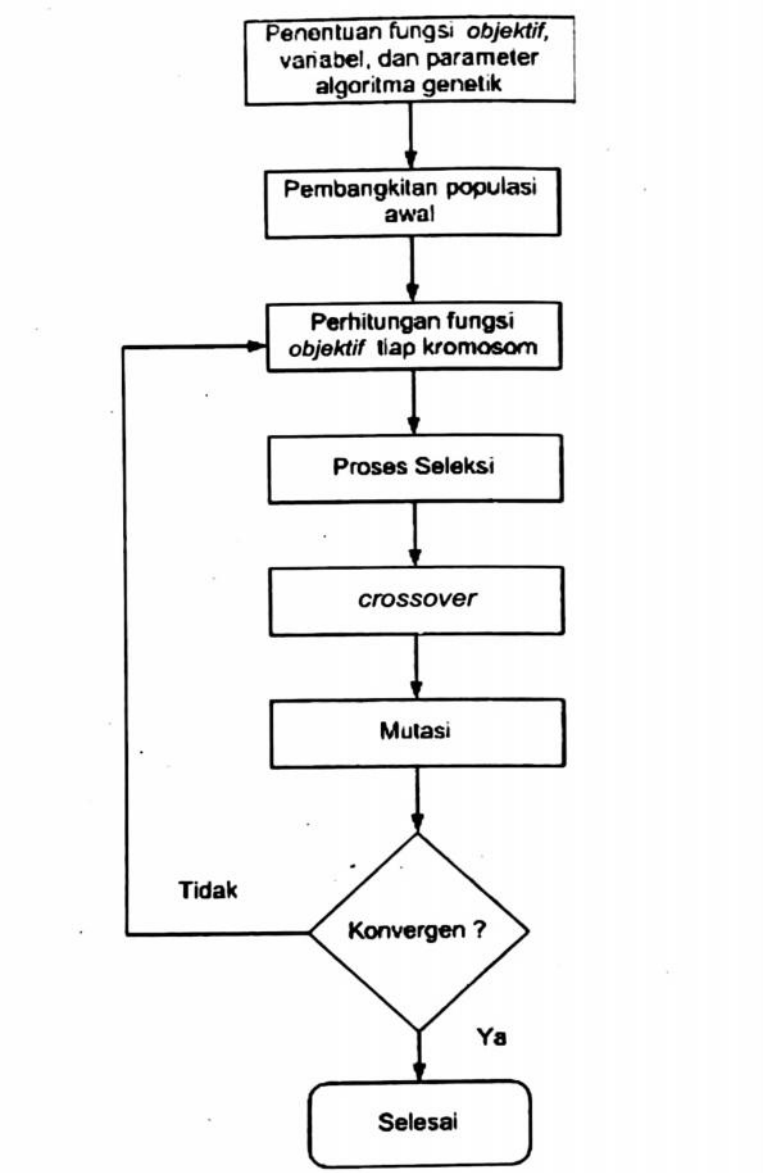


Gambar 11.17 Grafik nilai fitness

11.9 Pengkodean Dengan Bilangan Rill

Algoritma genetik dapat dijalankan dengan menggunakan pengkodean biner maupun pengkodean dengan bilangan rill. Dengan pengkodean biner, setiap parameter akan dikonversi ke string biner. String-string ini akan digabungkan dan operasi genetik akan dilakukan pada string gabungan ini. Dengan pengkodean bilangan rill, parameter akan dinyatakan dalam bilangan rill. Keda bentuk pengkodean ini diaplikasikan dalam praktik.

Kelemahan dari pengkodean biner adalah adanya *error* kuantisasi dan juga membutuhkan waktu komputasi yang lebih lama karena harus ada proses konversi nilai biner ke nilai rill. Untuk persoalan optimasi yang melibatkan nilai kontinu, maka pengkodean dengan bilangan rill lebih menguntungkan. Dengan cara ini ketelitian variabel dapat dimaksimalkan sesuai dengan kemampuan komputer, sedangkan dengan pengkodean biner diperlukann deretan bit yang cukup panjang untuk mendapatkan ketelitian yang tinggi. Dari sisi penggunaan memori, pengkodean dengan bilangan rill juga lebih hemat memori. Di samping itu, dari sisi komputasi, maka dengan pengkodean bilangan rill akan lebih cepat. Diagram alir algoritma genetik yang menggunakan pengkodean bilangan rill ditunjukkan pada Gambar 11.18.



Gambar 11.18 Diagram alir algoritma genetik dengan pengkodean bilangan rill

Variabel yang akan dikodekan dengan bilangan rill dinormalkan dengan persamaan 11.12. sehingga nilainya berada dalam rentang [0,1].

$$P_{nom} = \frac{(p - p10)}{(phi - p10)} \quad (11.12)$$

Sedangkan untuk proses dekoding digunakan persamaan:

$$P = P_{nom}(P_{hi} - P_{10}) + P_{10} \quad (11.13)$$

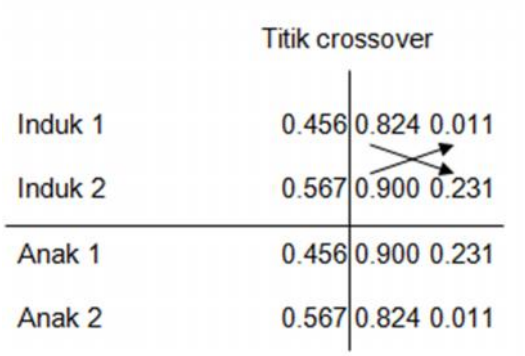
Contoh 11.9

Jika digunakan pengkodean dengan bilangan rill pada masalah optimasi pada contoh 11.1, variabel x = 5 dan y = 7, dinormalkan menjadi:

$$x_{nom} = \frac{5-10}{10-0} = 0.5$$

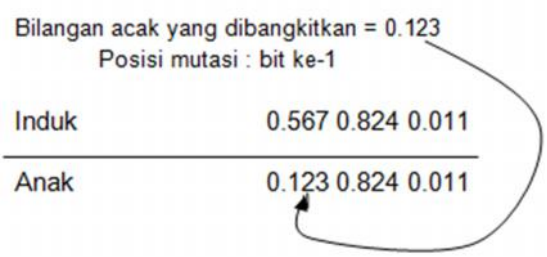
$$y_{nom} = \frac{7-10}{10-0} = 0.7$$

Penentuan populasi awal dilakukan dengan cara membangkitkan bilangan acak dalam rentang [0,1] sebanyak $N_{pop}xN_{var}$, proses *crossover* dilakukan dengan cara memindah silang bagian-bagian kromosom sebagaimana dijelaskan pada gambar 11.19.

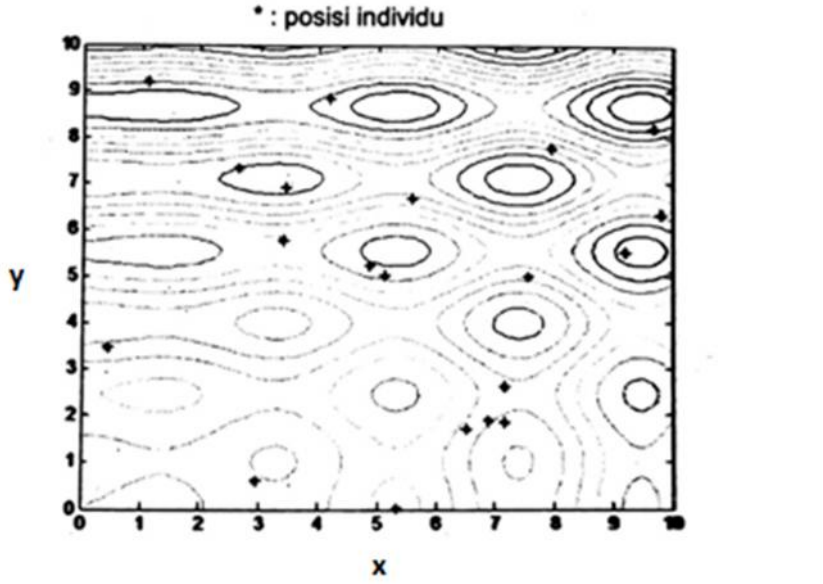


Gambar 11.9 Crossover sat titik pada pengkodean bilangan rill

Sedangkan proses mutasi dilakukan dengan cara yang sedikit berbeda dari proses mutasi yang terjadi pada pengkodean bilangan biner. Setelah bagian kromosom yang akan dimutasi ditentukan berdasarkan probabilitas mutasi, maka mutasi dilakukan dengan cara menggantikan bagian kromosom tersebut dengan sebuah nilai acak. Proses tersebut diilustrasikan pada Gambar 11.20.



Gambar 11.20 Mutasi pada pengkodean dengan bilangan rill

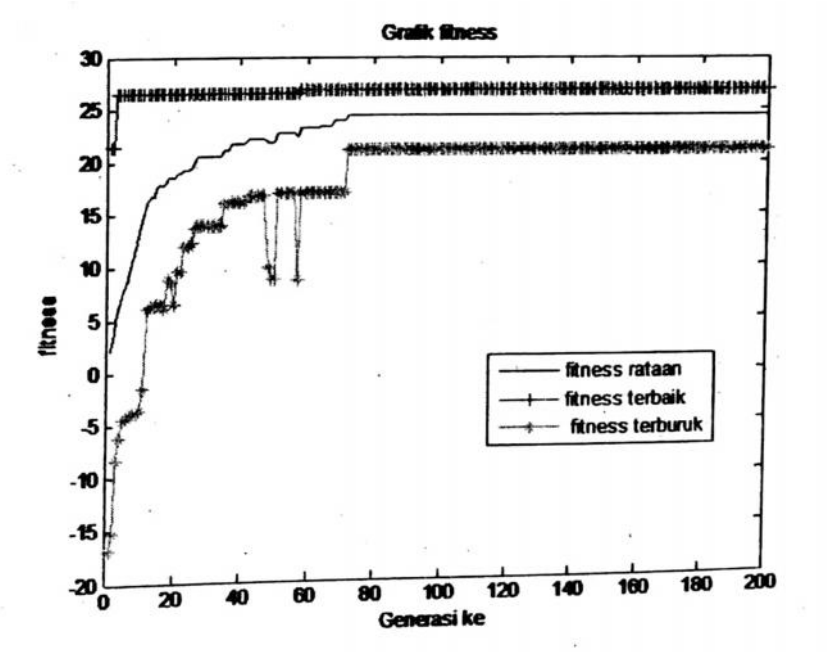


Gambar 11.21 sebaran individu pada populasi inisial

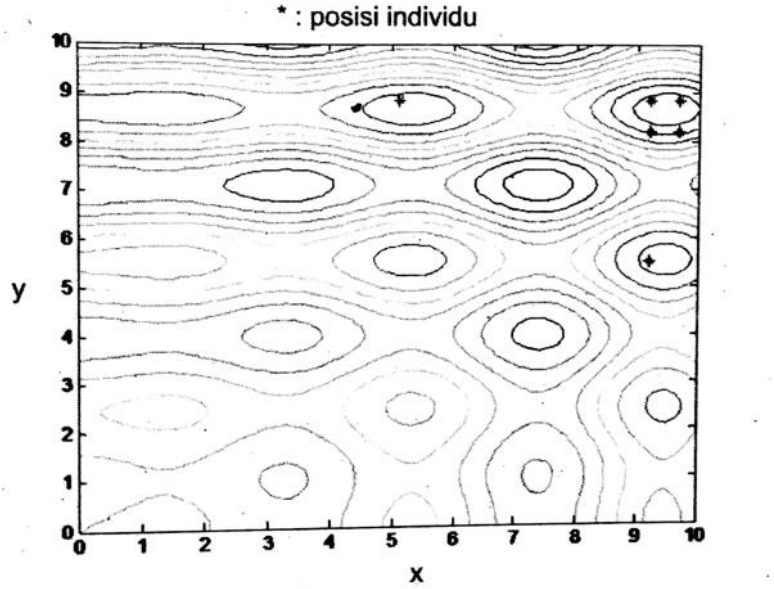
Contoh 11.10

Masalah optimasi pada contoh 11.1 dapat diselesaikan dengan aplikasi algoritma genetik yang menggunakan pengkodean bilangan rill. Pada aplikasi ini dibangkitkan populasi dengan 20 individu, nilai probabilitas crossover = 0.95, nilai probabilistik mutasi = 0.20, dan maksimum generasi = 200. Gambar 11.20 menunjukkan sebaran individu pada populasi awal. Gambar 11.22 menunjukkan grafis fitness selama proses iterasi yang menunjukkan adanya perbaikan individu dari generasi ke generasi. Gambar 11.22

menunjukkan sebaran individu dalam populasi setelah iterasi ke-200. Pada gambar tersebut dapat dilihat individu-individu konvergen menuju ke posisi solusi optimal. Hasil terbaik yang diperoleh baru mendeteksi solusi optimal belum sampai pada titik maksimumnya, yaitu x = 9.23 dan y = 8.91 dengan nilai fitness f(x,y) = 25.813.



Gambar 11.22 Grafik fitness algoritma genetik dengan pengkodean bilangan rill



Gambar 11.23 Sebaran individu pada populasi iterasi ke-200

11.10 Inisialisasi Parameter Algoritma genetik

Beberapa parameter algoritma genetik yang perlu diinisialisasi adalah ukuran populasi (popsize), probabilitas crossover (p_c) dan probabilitas mutasi (p_m). penentuan parameter dilakukan dengan memperhatikan permasalahan yang akan dipeccahkan. Ada beberapa rekomendasi yang bisa digunakan, antara lain:

- untuk permasalahan yang memiliki kawasan solusi cukup besar, direkomendasikan nilai parameter:
  (popsize; p_c; p_m) = (50; 0,6; 0,001).
- Bila rata-rata fitness setiap generasi digunakan sebagai indikator, maka direkomendasikan nilai parameter:
  (popsize; p_c; p_m) = (30; 0,95; 0,01).
- Bila fitness dari individu terbaik dipantau pada setiap generasi, maka usulannya adalah:
  (popsize; p_c; p_m) = (80; 0,45; 0,01).
- Ukuran populasi sebaiknya tidak lebih kecil dari 30, untuk sembarang jenis permasalahan.

11.11 Penanganan Optimasi Dengan Batasan Menggunakan Algoritma Genetik

Pada umumnya, masalah optimasi menyertakan suatu batasan (*constraint*) yang harus dipenuhi oleh solusi yang ada. Algoritma genetik dapat menyelesaikan persoalan semacam itu dengan dengan menggunakan metode fungsi pinalti yaitu pengurangan rangking nilai fitness sebanding dengan violasi batasan tersebut. Dalam metode pinalti, masalah optimasi dengan batasan ditransformasikan ke dalam bentuk optimasi tanpa batasan (*unconstraint*) dengan cara menggabungkan fungsi fitness dengan nilai pinalti untuk semua batasan yang ada. Misalnya fungsi obyektif pada masalah minimasi dituliskan sebagai persamaan 11.14.

Minimumkan g(x)
Dengan batasan $h_i(x) \geq 0$, i=1,2,...,n (11.14)

dapat ditransformasikan ke dalam masalah minimasi tanpa batasan seperti pada persamaan 11.15.

Minimumkan $g(x) + \sum_i^n r\phi(h_i(x))$

dengan $\phi(.)$ adalah fungsi pinalti dan r adalah koefisien pinalti. Dengan transformasi ini algoritma genetik dapat dijalankan untuk menyelesaikan masalah tersebut.

C. Rangkuman

1. Algoritma genetik menggunakan tiga aturan utama pada setiap iterasi untuk menghasilkan generasi baru dari populasi saat ini, yaitu:
   a. Aturan seleksi untuk memilih individu sebagai induk yang berkontribusi pada terbentuknya generasi baru
   b. Aturan crossover untuk menggabngkan dua induk untuk membentuk keturunan pada generasi baru.
   c. Aturan mutasi yang mengaplikasikan perubahan acak pada individu induk untuk membentuk keturnan.
2. Perbedaan algoritma standar dengan algoritma genetik

Algoritma standar	Algoritma genetik
Membangkitkan satu titik tunggal setiap iterasi, sehingga titik tersebut akan mencapai solusi optimal	Membangkitkan populasi titik-titik solusi pada setiap iterasi, sehingga titik-titik pada populasi tersebut mencapai solusi-solusi optimal
Memilih titik pada urutan berikutnya dengan komputasi yang ditentukan	Memilih populasi berikutnya dengan komputasi yang melibatkan pilihan acak

3. Optimasi adalah proses pengaturan input dan karakteristik suatu alat, proses, atau eksperimen untuk menghasilkan output yang maksimum atau minimum. Input terdiri atas variabel, sedangkan proses atau fungsi disebut sebagai fungsi objektif, fungsi biaya (cost), atau fungsi fitness, dan output proses adalah nilai biaya atau nilai fitness.
4. Kromosom adalah string yang berisi kode/sandi yang mungkin untuk parameter/variabel yang akan diopttimasikan.
5. Kelemahan dari pengkodean biner adalah adanya *error* kuantisasi dan juga membutuhkan waktu komputasi yang lebih lama karena harus ada proses konversi nilai biner ke nilai rill.
6. *Multipoints crossover* adalah perluasan dari *crossover* satu titik. Pada *multipoints crossover* ini dipilih beberapa titik *crossover* secara acak, lalu bagian string diantara titik-titik ini dipertemukan antar dua individu yang melakukan *crossover* dua titik.

D. Tugas

Carilah referensi lain mengenai algoritma genetik seperti yang telah dibahas pada bab 11 ini.

E. Tes formatif

1. Apa yang dimaksud dengan algoritma genetik?
2. Apa yang dimaksud dengan gen?
3. Jelaskan apa yang dimaksud dengan elitisme!
4. Dalam pembelajaran algoritma genetika, David Golberg adalah tokoh yang pertama kali memperkenalkan siklus dari algoritma genetika. Sebutkan siklus algoritma genetika yang dimaksud!

KUNCI JAWABAN

BAB 1

1. Pengertian AI:
   Artificial Intelligence (AI) atau Kecerdasan Buatan didefinisikan sebagai kecerdasan yang ditunjukan oleh suau entitas buatan. Kecerdasan diciptakan dan dimasukkan kedalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti apa yang dilakukan manusia.
2. Kelebihan kecerdasan buatan antara lain:
   a. Lebih bersifat permanen. Kecerdasan alami bisa berubah karena sifat manusia pelupa. Kecerdasan buatan tidak berubah selama sistem komputer dan program tidak mengubahnya.
   b. Lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari 1 orang ke orang lain membutuhkan proses yang sangat lama dan keahlian tidak akan pernah dapat diduplikasi dengan lengkap.Jadi jika pengetahuan terletak pada suatu sistem komputer, Pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer yang lain.
   c. Lebih murah. Menyediakan layanan komputer akan lebih mudah dan murah dibandingkan mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
   d. Bersifat konsisten dan teliti karena kecerdasan buatan adalah bagian dari teknologi komputer sedangkan kecerdasan alami senantiasa berubah-ubah.
   e. Dapat didokumentasi.Keputusan yang dibuat komputer dapat didokumentasi dengan mudah dengan cara

melacak setiap aktivitas dari sistem tersebut.Kecerdasan alami sangat sulit untuk direproduksi.

f. Dapat mengerjakan beberapa task lebih cepat dan lebih baik dibanding manusia.

3. Beberapa dari aplikasi kecerdasan buatan, yaitu:
   a. Game playing
   b. Sistem bahasa alami
   c. Sistem perancangan
   d. sistem pakar reparasi perangkat keras
   e. menejemen data cerdas
   f. sistem otomatisasi kantor, dll.

BAB 2

1. Pengertian jaringan syaraf tiruan:
   Jaringan syaraf tiruan (artificial neural network) adalah sistem komputasi arsitektur dan operasinya diilhami dari pengetahuan tentang sel syaraf jaringan biologis didalam otak.

2. Kelebihan jst
   a. Adaptive learning: Suatu kemampuan untuk melakukan suatu kegiatan yang didasarkan atas data yang diberikan pada saat pembelajaran atau dari pengalaman sebelumnya.

   b. Self-Organisation: Dapat membuat organisasi sendiri atau me-representasikan informasi yang didapat pada saat pembelajaran.

   c. Real Time Operation: Dapat menghasilkan perhitungan parallel dan dengan device hardware yang khusus yang dibuat akan memberikan keuntungan dengan adanya kemampuan tersebut.

   d. Fault Tolerance melalui Redundant Information Coding: Kerusakan pada bagian tertentu dari jaringan akan mengakibatkan penurunan kemampuan. Beberapa

jaringan mempunyai kemampuan untuk menahan kerusakan besar pada jaringan.

e. Kelebihan Jaringan Syaraf Tiruan terletak pada kemampuan belajar yang dimilikinya. Dengan kemampuan tersebut pengguna tidak perlu merumuskan kaidah atau fungsinya. Jaringan Syaraf Tiruan akan belajar mencari sendiri kaidah atau fungsi tersebut. Dengan demikian Jaringan Syaraf Tiruan mampu digunakan untuk menyelesaikan masalah yang rumit dan atau masalah yang terdapat kaidah atau fungsi yang tidak diketahui.

f. Kemampuan Jaringan Syaraf Tiruan dalam menyelesaikan masalah yang rumit telah dibuktikan dalam berbagai macam penelitian.

3. Perbandingan jaringan syaraf tiruan dengan konvensional, yaitu:
Jaringan syaraf tiruan memiliki pendekatan yang berbeda untuk memecahkan masalah bila dibandingkan dengan sebuah komputer konvensional. Umumnya komputer konvensional menggunakan pendekatan algoritma (komputer konvensional menjalankan sekumpulan perintah untuk memecahkan masalah). Jika suatu perintah tidak diketahui oleh komputer konvensional maka komputer konvensional tidak dapat memecahkan masalah yang ada. Sangat penting mengetahui bagaimana memecahkan suatu masalah pada komputer konvensional dimana pengguna belum mengetahui bagaimana melakukannya. Jaringan syaraf tiruan dan suatau algoritma komputer konvensioanl tidak dapat saling bersaing namun melengkapi satu sama lain. Pada suatu kegiatan yang besar, sistem yang diperlukan biasanya menggunakan kombinasi keduanya (biasanya sebuah komputer konvensional digunakan untuk mengontrol jaringan syaraf tiruan untuk menghasilkan efisiensi yang maksimal. Jaringan syaraf tiruan tidak

memberikan suatu keajaiban tetapi jika digunakan secara tepat akan menghasilkan sesuatu hasil yang luar biasa.

4. Tiga arsitektur jaringan syaraf tiruan, yaitu:
   a. jaringan dengan lapisan tunggal (*single layer net*)
   b. Jaringan dengan banyak lapisan (multilayer net)
   c. Jaringan dengan lapisan kompetitif (competitive layer net)

BAB 3

1. Pengertian perceptron:
   Perceptron adalah salah satu metode jaringan syaraf tiruan yang sederhana dipakaikan prosedur algoritma training yang pertama kali. Terdiri dari neuron tunggal dengan bobot synaptic yang diatur menggunakan fungsi aktivasi hard limit.

2. Apa yang dimaksud dengan perceptron multilapis?

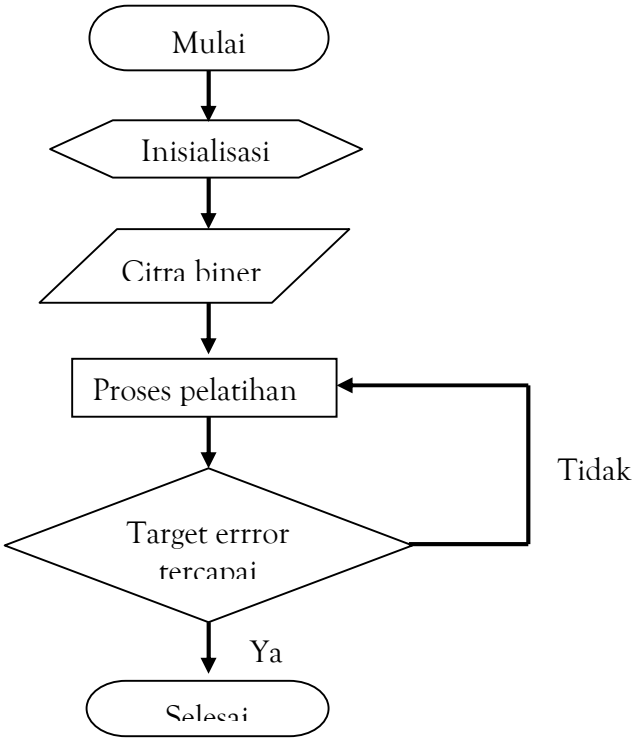
   Perceptron multilapis adalah jaringan syaraf tiruan umpan maju (feedforward neural networks) dan merupakan jaringan yang pembelajarannya terawasi sehingga ke dalam jaringan perlu dimasukkan contoh-contoh respons untuk dikenali

3. Tujuan dari algoritma pelatihan dan cara penerapannya, yaitu:

   Algoritma pelatihan digunakan untuk melatih jaringan syaraf tiruan, yaitu dengan cara mengajarinya dengan contoh-contoh kasus/pola sampai jaringan syaraf tiruan berhasil mengenali pola tersebut. Setiap kali output yang dihasilkan jaringan tidak sesuai dengan target yang diharapkan maka setiap kali pula bobotnya di-update. Hal ini terus-menerus dilakukan sampai tidak ada lagi bobot yang berubah untuk setiap pasangan latihan sensor dan terget. Bobot-bobot terakhir yang diperoleh pada saat pelatihan jaringan syaraf tiruan inilah yang akan digunakan pada saat pengaplikasian (dengan menggunakan algoritma aplikasi perceptron).

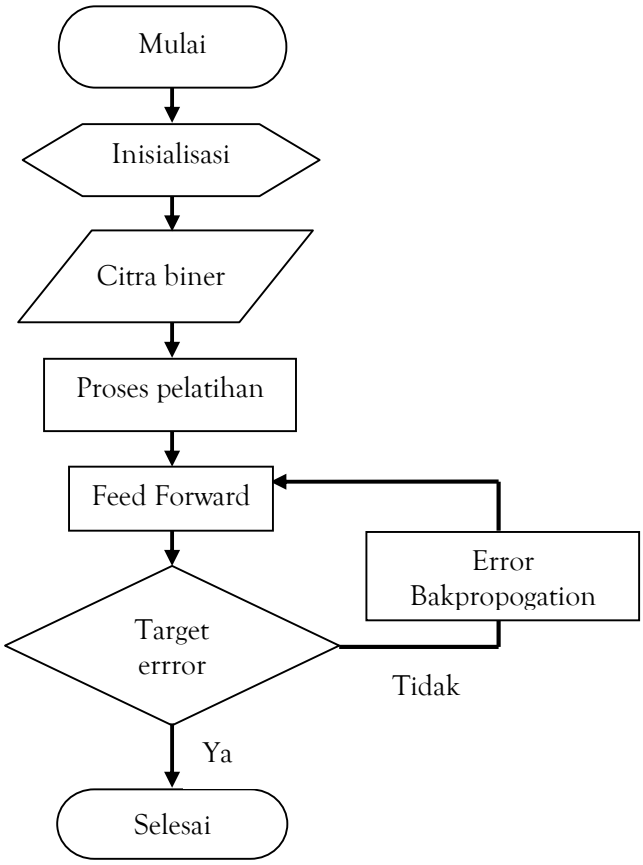
BAB 4

1. Pengertian jaringan hopfield diskrit:
   Jaringan Hopffield diskrit merupakan jaringan syaraf tiruan yang terhubung penuh (*fully connected*), yaitu bahwa setiap unit terhubung dengan setiap unit lainnya.
2. Tiga contoh jaringan hopfield diskrit:
   - Untuk pengenalan huruf pada citra digital
   - Untuk prakiraan cuaca
   - Pada traveling salesman problem
3. Gambar alur proses dari jaringan syaraf tiruan secara umum!



BAB 5

1. Pengertian jarngan syaraf propagasi balik:
   Jaringan syaraf Jaringan saraf propagasi balik merupakan jaringan saraf yang menggunakan konsep jaringan berlapis jamak. Lapisan pertama adalah lapisan masukan (input)dan yang terakhir adalah lapisan keluaran (output). Lapisan diantara lapisan masukan dan lapisan keluaran disebut dengan lapisan tersembunyi (hidden)
2. Contoh aplikasi jaringan syaraf tiruan propagasi balik
   - Penegenalan huruf digital pada citra digital
   - Memprediksi tingkat suku bunga bank
3. Gambaran umum alur proses dari jaringan syaraf propagasi balik



4. Metode PB pertama kali dirumuskan oleh Werbos dan dipopulerkan oleh Rumelhart dan McClelland.

BAB 6

1. Beberapa contoh perangkat keras yang mengimplementasikan model jaringan syaraf tiruan:
   - Spert-II (*synthetic Perceptron Testbed II*) merupakan prototip mikroprosesor vektor *full-custom*. Arsitektur mikroprosesor vektor akan membuat elemen-elemen

pengolahan untuk aplikasi multimedia dan atar muka manusia-mesin yang sering berisi algoritma berbentuk data paralel. Spert-II mempercepat pelatihan multiparameter jaringan syaraf tiruan untuk pengenalan suara.

- *Corprocessor board* yang khusus dirancang untuk aplikasi mesin yang bisa mengkonversi suara ke dalam tulisan (*neural phonetic typewriter*).
- Penelitian mengenai pengimplementasian jaringan syaraf tiruan pada sirkuit VLSI CMOS telah banyak dilakukan. Salah satunya adalah pengimplementasian model jaringan syaraf tiruan pada sirkuit VLSI CMOS sebagai memori asosiatif maupun sebagai pengklasifikasi pola.

2. Hal yang diperhatikan dalam pemilihan paradigma:
- ukuran jaringan
- Pembawaan input dan output
- Mekanisme memori
- Tipe pelatihan: terawasi dan tidak terawasi
- Batasan waktu operasi rutin dari sistem berjalan.

3. Tiga aspek deklaratif dalam verifikasi jaringan syaraf tiruan:
- Jaringan
- Unit
- Perilaku

BAB 7

1. Aplikasi pengenalan karakter alfanumerik:
   Aplikasi pengenalan karakter alfanumerik merupakan suatu implementasi dari algoritma neural network. Neural network atau jaringan syaraf tiruan adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan syaraf manusia.
2. Lapisan yang terdapat pada jaringan syaraf tiruan dan prinsip kerjanya:

Jaringan saraf tiruan untuk aplikasi pengenalan karakter alfanumerik terdiri dari 3 lapisan yaitu lapisan input, sebuah lapisan tersembunyi dan sebuah lapisan output. Jaringan propagasi balik akan menerima input biner sehingga data harus diatur sebagai sekumpulan angka (vektor). Karakter alfanumerik diterjemahkan kedalam bentuk matriks ukuran 5 x 7 yang melambangkan pixel-pixel. Data dimasukkan dalam bentuk biner, angka 1 menunjukkan pixel berisi citra, angka 0 menunjukkan kosong. Pengujian jaringan dilakukan dengan menggunakan program aplikasi MATLAB yang akan mengenali karakter alfanumerik.

3. Contoh kemungkinan penerapan pengenalan karakter alfanumerik:
   Salah satu contoh kemungkinan penerapan aplikasi ini misalnya dipergunakan untuk mengenali plat kendaraan secara otomatis dari hasil tangkapan kamera digital yang dipasang di lapangan parkir di pusat perbelanjaan. Pengembangan aplikasi ini lebih lanjut memungkinkan untuk mengkombinasikan dengan algoritma yang ain dan juga proses pengolahan citra yang baik sehingga tingkat akurasnya bisa lebih ditngkatkan lagi.

BAB 8

1. Pengertian hypernet:
   Hypernet adalah perpaduan antara sistem pakar dengan jaringan syaraf tiruan yang digunakan untuk pendiagnosisan dan perawatan penyakit darah tinggi.

2. Beberapa aplikasi jaringan syaraf tiruan:
   - Detektor virus komputer
   - Pendeteksian kanker kulit
   - Pengidentifikasian pola-pola data pasar saham
   - Pendeteksi bom
   - Pengontrol gerakan dan penglihatan robot

- Pendukung pengolahan bahasa alami
- Pendukung DSS (Descission Support System)
- Sistem hibrid
- Basis data cerdas

3. Tujuan DCM:
   Tujuan DCM adalah untuk menganalisis entri-entri terpenting dari laporan kinis pasien dan untuk menentukan derajat kecocokan si pasien dengan masing-masing obat yang dipertimbangkan akan diberikan kepadanya.

BAB 9

1. Expert system:
   Sistem Pakar (Expert System) merupakan suatu sistem yang menggunakan pengetahuan manusia dalam komputer untuk memecahkan masalah yang biasanya dikerjakan oleh seorang pakar, misalnya : Dokter, Lawyer, Analist Keuangan, Tax Advisor. Sistem pakar dapat mendorong perhatian besar diantara ahli komputer dan spesialist informasi untuk mengembangkan sistem membantu manajer dan non manajer memecahkan masalah.

2. Kelebihan dan kekurangan model terintegrasi penuh:

   Kelebihan
   - Tegrasi yang kokoh
   - Perbaikan untuk kerja
   - Kemmampuan pemecahan masalah yang lebih baik

   Kekurangan model integrasi penuh:
   - Adanya kompleksitas dalam penspesifikasian, pendesainan, dan pengembangan sistem.
   - Kurangnya tool-tol yang memfasilitasi model integrasi penuh.
   - Sulit memverifikasi, dan memelihara sistem yang terintegrasi penuh.

3. Stand alone merupakan Sistem Operasi lengkap yang bekerja pada komputer desktop, notebook atau perangkat komputer bergerak.

BAB 10

1. Variabel linguistik:
   Variabel linguistik adalah variabel yang bernilai kata/kalimat, bukan angka. Sebagai alasan menggunakan kata/kalimat daripada angka karena peranan linguistik kurang spesifik dibandingkan angka, namun informasi yang disampaikan lebih informatif. Variabel linguistik ini merupakan konsep penting dalam logika fuzzy dan memegang peranan penting dalam beberapa aplikasi.
2. Logika fuzzy:
   Logika fuzzy adalah suatu cara yang tepat untuk memetakan suatu ruang input dalam suatu ruang output dan memiliki nilai yang berlanjut. Kelebihan logika fuzzy ada pada kemampuan penalaran secara bahasa. Sehingga, dalam perancangannya tidak memerlukan persamaan matematis yang kompleks dari objek yang akan dikendalikan.
3. Kelebihan dan kekurangan dari logika fuzzy:

   Kelebihan:
   a. Konsep logika fuzzy mudah dimengerti. Konsep matematis yang mendasari penalaran fuzzy sangat sederhana dan mudah dimengerti.
   b. Logika Fuzzy sangat fleksibel.
   c. Logika fuzzy memiliki toleransi terhadap data-data yang tidak tepat.
   d. Logika Fuzzy mampu memodelkan fungsi-fungsi non linearyang sangat kompleks.
   e. Logika fuzzy dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.

f. Logika Fuzzy dapat bekerjasama dengan teknik-teknik kendali secara konvensional.
g. Logika fuzzy didasarkan pada bahasa alami.

Kekurangan:

a. Model Mamdani atau Sugeno atau model lain?
Penentuan model inference harus tepat, Mamdani biasanya cocok untuk masalah intuitive sedangkan sugeno untuk permasalahan yang menangani control
b. Jumlah Nilai Linguistik untuk setiap variabel?
Kita harus merubah nilai crisp menjadi nilai linguisik. Jumlah dari nilai linguistik yang digunakan harus sesuai dengan permasalahan yang akan kita selesaikan.
c. Batas-batas Nilai Linguistik?
Batas-batas nilai linguistik akan sangat berpengaruh pada akurasi fuzzy logic.

4. Fuzzifikasi dan defuzzifikasi:
Fuzzifikasi melakukan pengubahan dari suatu nilai skalar ke dalam himpunan yang bernilai atau bersifat fuzzy. Sedangkan defuzzifikasi adalah proses untuk mengubah dari nilai fuzzy ke dalam nilai crisp/real.

BAB 11

1. Algoritma genetik:

   Algoritma genetik adalah teknik pencarian heuristik yang didasarkan pada gagasan evolusi seleksi alam dan genetik.

2. Gen (genotype), yaitu:
Gen (genotype) adalah variabel dasar yg membentuk suatu kromosom. Dalam algoritma genetika, gen ini bisa bernilai biner, float,integrer maupun karakter.

3. Pengertian elitisme:
Elitisme adalah prosedur untuk meng-copy individu yang mempunyai nilai fitness tertinggi sebanyak satu (bila jumlah individu dalam suatu populasi adalah ganjil) atau dua (bila

jumlah individu dalam suatu populasi adalah genap). Hal ini dilakukan agar individu ini tidak memiliki kerusakan (nilai fitnessnya menurun) selama proses pindha silang maupun mutasi.

4. Siklus algoritma genetika yang dimaksud David Golberg:
   a. Populasi Awal
   b. Evaluasi
   c. Fitness
   d. Seleksi Individu
   e. Reproduksi:
   f. Cross-Over
   g. Dan Mutasi
   h. Populasi Baru

GLOSARIUM

AI : Artificial Intelligence

Aktivasi : Proses membuat aktif

Ambang : Balok yang melintang

Asosiasi : Persatuan antar rekan usaha

Backpropogation : Pembelajaran untuk memperkecil tingkat error dengan cara menyesuaikan bobotnya berdasarkan perbedaan output dan target yang diinginkan.

Bipolar : Mempunyai dua kutub

Booming : Peledakan informasi

Coding : Secara garis besar merupakan pemrograman

Diagonal : Berhubungannya dua titik sudut yang tidak bersebelahan dalam suatu segi empat

Deklaratif : Bersifat pernyataan ringkas dan jelas

Epoch : Skala waktu geologi yang menyususn suatu periode

Error : Kekeliruan, tidak tepat, kesalahan secara perangkat lunak, atau kerusakan pada perangkat keras.

Fleksibilitas : Kelenturan

Generalized : Menyamaratakan

Inisialisasi : Tugas pemberian nilai awal (data awal) yang dilakukan saat deklarasi variabel atau obyek.

Intensif : Secara sungguh-sungguh dan terus-menerus dalam mengerjakan sesuatu hingga memperoleh hasil yang optimal.

Interpretasi : Pemberian kesan, pendapat, atau pandangan teoretis terhadap sesuatu.

Konsistensi : Ketetapan dan kemantapan (dalam bertindak).

Kompatibilitas : Keadaan menyesuaikan diri.

Linguistik : Ilmu yang merinci bahasa secara umum atau tentang bahasa tertentu.

Logika : Pengetahuan tentang kaidah berpikir

Matriks : Kerangka, bagan

Monoton : Jenis adegan dengan dinamika yang selalu sama.

Neuron : Merupakan satuan kerja utama dari sistem saraf yang berfungsi menghantarkan impuls listrik yang terbentuk akibat adanya suatu stimulus (rangsang).

Node : Adalah setiap komputer, printer atau periferal yang terhubung dalam jaringan.

Noise : Suatu sinyal gangguan yang bersifat akustik (suara).

Paradigma : Kerangka berpikir

Siklus : Putaran waktu yang di dalamnya terdapat rangkaian kejadian yang berulang-ulang secara tetap dan teratur.

Sekuensial : Rangkaian logika yang kondisi keluarannya dipengaruhi oleh masukan dan keadaan keluaran sebelumnya atau dapat dikatakan rangkaian yang bekerja berdasarkan urutan waktu.

Segmen : Bagian

Supervised learning : Sebuah pendekatan dimana sudah terdapat data yang dilatih, dan terdapat variable yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengkelompokan suatu data ke data yang sudah ada.

Stabilitas : Kemantapan; kestabilan; keseimbangan:

Transfer : Pindah atau beralih tempat

Unsupervised learning : Sebuah pendekatan dimana ai tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokan data tersebut menjadi 2 bagian atau 3 bagian dan seterusnya.

Vektor : Besaran yang memiliki ukuran dan arah, misalnya gaya kecepatan.

Verifikasi : Pemeriksaan tentang kebenaran laporan, pernyataan, perhitungan uang, dan sebagainya.

Validasi : Pengujian kebenaran atas sesuatu

DAFTAR PUSTAKA

Diyah Puspita Ningrum. 2006. *Pengantar Jaringan Syaraf Tiruan*. Yogyakarta: C.V ANDi OFFSET

Dwi Ana Ratna Wati. 2011. *Sistem Kendali Cerdas (Fuzzy Logic Controller, Jaringan Syaraf Tiruan, Algoritma Genetik (AG)*. Yogyakarta: Graha Ilmu

Handy Wicaksono. 2009. *Programmable Logic Controller*. Yogyakarta: Graha Imu

Hendra Jaya. 2016. *Desain dan Implementasi Sistem Robotika Berbasis Mikrokontroller*. Makassar: Edukasi Mitra Grafika. Majid Abdul. 2005. *Perencanaan Pembelajaran (Mengembangkan Kompetensi Guru).* Bandung: Remaja Rosdakarya.

Kusuma Dewi, Sri . 2003. *Artificial Intelligence* (*Teknik dan Aplikasinya*). Yogyakarta: Graha Ilmu

Siswanto. 2010. *Kecerdasan Tiruan*. Yogyakarta: Graha Ilmu

Suparman. 1991. *Mengenal Artifial intelligence*. Yogyakarta: C.V ANDI OFFSET

Suyanto. 2014. *Artificial Intelligence (Searching – Reasoning – Planning – Learning).* Bandung: Informatika Bandung.

Sri Kusumadewi dan Sri Hartati. 2010. *Neuro Fuzzy (Integrasi Sistem Fuzzy & Jaringan Syaraf)*. Yogyakarta: Graha Ilmu.

Sri Kusumadewi dan Hari Purnomo. 2004. *Aplikasi Logika Fuzzy Untuk Pendukung Keputusan*. Yogyakarta: Graha Ilmu.