MENU UTAMA

JARINGAN SYARAF TIRUAN

Lettu Lek. Arwin D. Wahyudi S.*
dan Ir. Yusep Rosmansyah**

JST DALAM KOMPUTER "CERDAS"

Bagian 2: Bekerja dengan JST

Evaluasi terhadap Jaringan Syaraf Tiruan (JST) untuk aplikasi-aplikasi tertentu memerlukan suatu pemahaman yang baik mengenai teknik-teknik pengembangan dan algoritma-algoritmanya.

Pada tulisan bagian pertama, penulis telah mengete ngahkan perkembangan JST serta prospeknya di masa mendatang dipandang dari segi perkembangan teknologi maupun ekonomi.

Dalam tulisan bagian kedua ini saya akan memberikan contoh kasus bagaimana memanfaatkan JST untuk membantu meringankan pekerjaan kita di berbagai bidang dan disesuaikan dengan kebutuhan kita. Untuk itu, di sini saya akan memberikan contoh penggunaan JST Back Propagation (BP) untuk menyelesaikan masalah XOR.

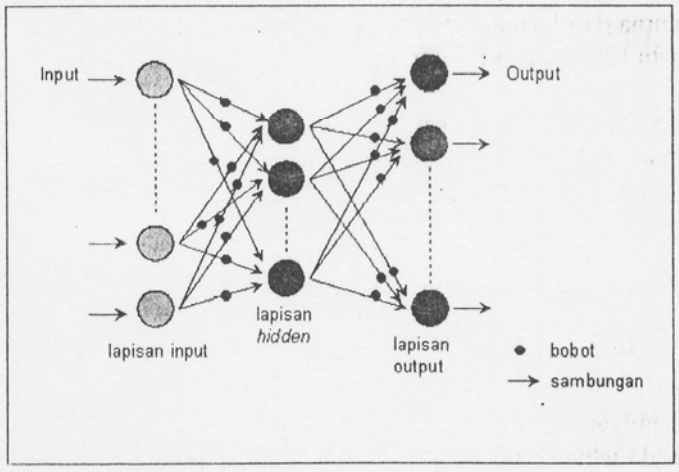
Prinsip Kerja JST

Sebelum kita melihat sisi aplikasi JST, ada baiknya bila kita tinjau dulu bagaimana JST ini bekerja.

Salah satu contohnya adalah JST tiga-lapis yang belajar dengan menggunakan algoritma propagasi balik atau Back Propagation (BP).

JST ini digunakan sebagai contoh dengan alasan JST ini yang saat ini cukup populer. Tetapi tidak semua JST mempunyai tiga-lapis dan algoritma propagasi balik bukan satau-satunya metode belajar yang bisa digunakan pada JST.

JST yang menggunakan BP dapat dipahami dalam beberapa tingkatan. Pada satu tingkat, ia merupakan kumpulan dari persamaan-persamaan vektor; pada tingkat lainnya, dapat dipandang sebagai suatu program komputer dan tingkat lainnya lagi, dipandang sebagai suatu sistem berlapis dengan simpul-simpul yang saling berinteraksi seperti tampak pada Gambar 1.



Gambar 1. JST dengan BP

JST terbentuk dari simpul-simpul yang mirip dengan neuron yang diatur dalam lapisan-lapisan dan melewatkan data melalui sambungan terbobot.

JST belajar dengan mengubah nilai-nilai bobotnya. Dengan bobot-bobot yang sesuai, suatu JST dapat memodelkan suatu fungsi yang dapat dikomputasi.

Komputasi dasar untuk suatu JST adalah hasil dot-vektor, dan kecepatan proses komputasinya bergantung pada pelaksanaan operasi-operasi akumulasi dan perkalian yang mendasar secara efisien.

Setiap simpul (node) adalah sebuah elemen pengolah tunggal yang mengolah data untuk memproduksi suatu hasil. Simpul-simpul tersebut diatur dalam lapisan-lapisan dan masing-masing terhubung ke simpul-simpul pada lapisan sebelumnya untuk input dan lapisan berikutnya untuk output.

Selanjutnya, setiap sambungan (connection) mempunyai sebuah nilai yang dapat diubah-ubah besarnya yang disebut dengan bobot (weight).

Data memasuki JST melalui lapisan input. Simpul-simpul pada lapisan input bersifat pasif, tidak melakukan perhitungan; masing-masing mengirimkan nilai data tersebut melalui bobot-bobot sambungan ke simpul-simpul tersembunyi (hidden). Setiap simpul BP juga mempunyai input tambahan yang disebut dengan nilai ambang (threshold), yang bertindak sebagai suatu tingkat referensi atau s simpul tersebut.

Semua simpul-simpul tersembunyi tersebut kemudian menerima data dari lapisan input, tetapi karena setiap simpul mempunyai paket bobot yang berbeda, maka nilai paket-paket juga berbeda. Setiap simpul tersembunyi ini selanjutnya mengolah input-inputnya dan mengirimkan hasilnya ke lapisan output. Simpul-simpul output juga mempunyai satu paket bobot yang berbeda dan mengolah nilai input untuk memproduksi suatu hasil.

Untuk BP, hasil latihan adalah satu paket nilai-nilai variabel. satu untuk setiap output. Simpul-simpul tersembunyi tidak mempunyai hubungan langsung dengan input atau output. Dengan memberikan lapisan antara ini akan meningkatkan kemampuan JST untuk memodelkan fungsi-fungsi yang rumit.

Simpul-simpul tersembunyi dan simpul-simpul output mengolah input-input yang diterimanya dalam dua tahap. Masing-masing dari mereka mengalikan setiap input dengan bobot-bobotnya, menambahkan hasilnya untuk mendatkan hasil total penjumlahan, dan kemudian melewatkan jumlahnya melalui suatu fungsi tertentu untuk mendapatkan hasil.

Fungsi transfer ini dinamakan dengan fungsi Sigmoid (1) yang juga memberikan ketidak-linieran dan selanjutnya akan meningkatkan kemampuan JST untuk memodelkan fungsi-fungsi yang rumit.

$$f(A) = 1/ (1 + \exp(-A)) \qquad (1)$$

dimana:

$$A = \sum x_i.w^*_{ji} - (\theta)_j \qquad (2)$$

x_i : input ke-i, dan
w_{ji} : bobot sambungan sel ke-j dari sel ke-i.
$(\theta)_j$: threshold (tegangan ambang sel ke-j).

Inti dari algoritma belajar propagasi balik ini terletak pada kemampuannya untuk mengubah nilai-nilai bobotnya untuk merespon adanya *error*.

Agar dapat menghitung error, data latihan harus mengandung serangkaian pola-pola input beserta pola-pola output yang menjadi targetnya. Dengan kata lain perlu adanya pola-pola output yang dijadikan referensi oleh JST; sehingga setiap JST mengeluarkan output, ia akan membandingkannya dengan hasil yang diharapkan tadi. Hasil perbandingan ini berupa error.

Selanjutnya JST melewatkan turunan-turunan dari error ke lapisan tersembunyi menggunakan sambungan terbobot yang masih belum diubah nilainya. Propagasi balik error-error inilah yang memberi nama JST ini sebagai JST BP.

Kemudian setiap simpul tersembunyi menghitung jumlah terbobot dari error yang dipropagasikan balik tadi untuk menghitung sumbangan tidak langsungnya kepada error-error output yang telah diketahui.

Setelah masing-masing simpul tersembunyi dan output menemukan besar errornya, simpul-simpul akan mengubah bobot-bobotnya untuk mengurangi error. Persamaan yang mengubah bobot-bobot tersebut dirancang untuk meminimisasi jumlah error kuadrat JST.

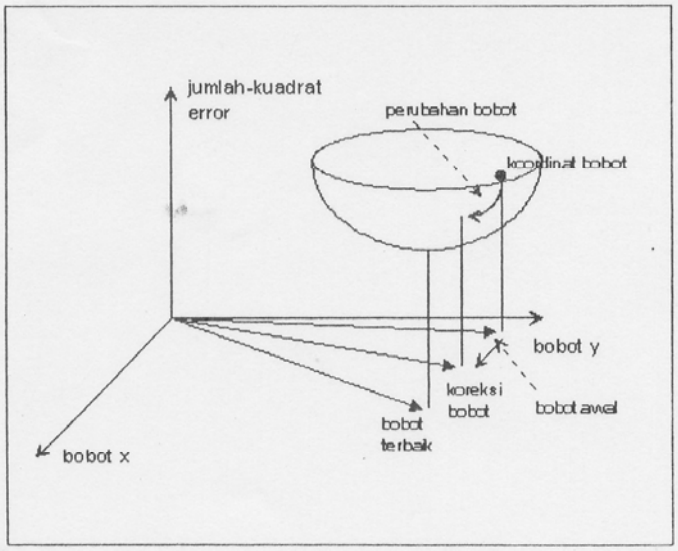
Proses minimisasi ini mempunyai maksud geometris secara intuitif. Untuk melihat hal ini, semua paket bobot harus diplot terhadap jumlah-kuadrat (sum squared) error yang berkaitan. Hasilnya adalah suatu permukaan error yang menyerupai bentuk sebuah baskom dimana bagian paling dasar menunjukkan jumlah kuadrat error yang terkecil. Dengan menemukan dasar baskom atau paket bobot yang paling baik merupakan tujuan utama selama melatih JST ini.

Algoritma propagasi balik mencapai kondisi ini dengan menghitung kemiringan permukaan error terhadap paket bobot saat itu. Selanjutnya algoritma ini akan mengubah paket bobot ke arah jalur yang tercepat untuk menuju ke dasar baskom.

Proses ini mirip dengan menggelindingkan bola dari atas bukit dan proses ini disebut dengan *gradient descent* (lihat Gambar 2). Oleh karena itu, algoritma propagasi balik dikatakan sebagai suatu prosedur untuk mendapatkan paket bobot yang meminimisasi jumlah-kuadrat error. Dengan kata lain, jumlah kuadrat error akan semakin mengecil dengan berjalannya waktu dan iterasi yang dilakukan oleh JST tersebut, kecuali bila JST gagal dalam melaksanakan tugasnya.

Algoritma propagasi balik belajar dengan mengubah paket bobotnya untuk mengikuti jalur tercuram (paling cepat) menuju ke dasar permukaan error yang berbentuk baskom.

Gambar 2 memperlihatkan bentuk ideal dari paket bobot dua-dimensi. Permukaan error sebenarnya tipikalnya menyerupai bentuk jurang dan local minimal menyerupai bentuk lekukan.



Gambar 2. Gradient descent

Arsitektur JST seperti di atas (tiga-lapis yang terhubung penuh) dan algoritma belajar propagasi balik ini hanya merupakan salah satu contoh.

Tidak semua hal yang mendetil digunakan pada semua JST. Elemen-elemen struktural seperti simpul-simpul, sambungan-sambungan, lapisan-lapisan, dan paket bobot merupakan hal yang universal, tetapi pengaturannya bisa berbeda-beda sesuai dengan aplikasinya.

Algoritma belajar atau aturan-aturan matematis yang mengontrol perubahan paket bobot juga bervariasi. Sedangkan algoritma-algoritma yang spesifik kadang dipasangkan dengan arsitektur-arsitektur JST yang juga spesifik, tingkah laku dan struktur JST tetap berdiri sendiri.

Tidak semua algoritma belajar memerlukan proses perulangan melalui data latihan atau mereka memerlukan data target. Algoritma yang memerlukan data target atau jawab yang benar terhadap suatu masalah disebut dengan algoritma *supervised* (dengan bantuan "guru").

Proses belajar yang hanya memerlukan satu tahapan disebut dengan proses belajar *reinforcement*. Algoritma *unsupervised* (tanpa bantuan "guru") sama sekali tidak menggunakan data target dan tipikalnya mencari hubungan statistik di antara pola-pola latihan.

Penggunaan JST untuk Aplikasi Tertentu

Tahap pertama dalam mencocokkan suatu aplikasi dengan JST yang akan digunakan adalah mempertimbangkan hal-hal sebagai berikut:

1. karateristik-karakteristik JST yang diinginkan;
2. persyaratan-persyaratan aplikasi; dan
3. kelemahan-kelemahan dari JST.

Faktor-faktor lain seperti masalah kecepatan komputasi juga harus menjadi perhatian.

Seperti telah dijelaskan sebelumnya, salah satu karakteristik umum aplikasi JST yang cukup berhasil adalah dalam tugas-tugas estimasi fungsi.

Tugas-tugas lainnya meliputi keperluan untuk menangani data dalam jumlah besar yang tidak lengkap, kemampuan untuk memberikan toleransi terhadap suatu hasil perkiraan, atau adanya sifat-sifat ketidak-linieran dan rumit.

Dari semuanya itu, yang paling penting adalah ketersediaan paket data latihan. Tidak terlalu sulit untuk mengumpulkan data dalam jumlah cukup untuk mewakili data dalam jumlah besar; atau kita akan mengalami kesulitan dalam melatih JST. Kadang-kadang diperlukan hasil akhir yang diinginkan terhadap data latihan yang dilatihkan pada JST.

JST mempunyai beberapa kelemahan untuk beberapa aplikasi tertentu. Kelemahan-kelemahan itu antara lain:

1. JST mempunyai kemungkinan untuk gagal dalam mendapatkan solusi yang memuaskan terhadap suatu masalah. Mungkin disebabkan tidak adanya fungsi yang dapat dipelajari atau jumlah data yang tidak mencukupi.
2. JST tidak dapat menjelaskan jawaban yang dikeluarkannya. Dalam satu hal, jawaban terhadap suatu masalah bergantung pada ribuan proses perhitungan yang melibatkan pola input dan bobot-bobot sambungan. Melihat bagaimana bobot-bobot tersebut menjadi "penyebab" suatu solusi akan lebih rumit daripada melihat bagaimana suatu program komputer bekerja. Di lain hal, nilai bobot-bobot itu sendiri merupakan hasil dari prosedur mesin-belajar (machine-learning) yang rumit dan susah untuk dijelaskan.
3. Melatih JST bisa lambat dan mahal. Hal ini menyangkut biaya yang dikeluarkan untuk keperluan mengumpulkan, menganalisis, dan memanipulasi data serta biaya ekseperimen-eksperimen terhadap parameter-parameter yang dilakukan untuk mendapatkan nilai-nilai parameter yang baik. Salah satu cara untuk mempercepat proses latihan JST adalah mengkombinasikannya dengan perangkat keras paralel yang dapat menjalankan eksperimen dengan cepat.
4. Kecepatan proses komputasi JST yang akan bertambah lambat bersamaan dengan bertambahnya jumlah simpul-simpul dalam JST. Ini karena waktu eksekusi secara langsung bergantung pada jumlah sambungan, secara kasar bergantung pada akar dari jumlah simpul-simpul dalam JST.

Membangun JST

Ketepatan suatu JST terhadap suatu aplikasi adalah dengan mengetahui tugas-tugas yang diperlukan agar mereka dapat bekerja, termasuk di dalamnya adalah merancang dan melatih JST, memilih prosedur perancangan, dan evaluasi terhadap pemilihan rancangan.

Secara umum, yang paling baik adalah membentuk apa yang diketahui dari masalah yang sedang dihadapi ke dalam data dan ke dalam JST. Atau cara lain dengan membangun pengetahuan ke dalam rancangan akan membuat JST menjadi lebih mudah dan lebih akurat.

Membuat Paket Data

Tahap paling awal dan merupakan tahap yang terlama serta secara umum merupakan hal yang paling kritis adalah membuat paket-paket data. Hal-hal yang berkaitan meliputi mengumpulkan data kasar, menganalisisnya, memilih variabel-variabel, dan pengolagan awal terhadap data-data tersebut sehingga JST dapat mempelajari dengan efisien.

Tujuan secara garis besar adalah membentuk satu file yang berisi rangkaian pola-pola input dimana masing-masing merupakan satu paket nilai-nilai yang terukur. Keseluruhan pola dilihat sebagai suatu vektor, dan nilai-nilai individu dilihat sebagai komponen-komponen vektor.

Pengolahan data awal mempunyai maksud mentransformasikan data tersebut ke dalam bentuk yang dapat dengan mudah dipelajari oleh JST. Dalam kondisi nyata, bentuk data latihan mempengaruhi hasil-hasil latihan.

Pengolahan data awal mungkin melibatkan operasi matematika atau juga teknik ektraksi ciri-ciri khusus (feature) dan pengolahan sinyal juga dapat digunakan.

Jumlah data yang diperlukan bergantung pada macam aplikasinya. Dalam kondisi praktis, kecukupan data bergantung pada beberapa faktor seperti ukuran JST, keperluan pengujian, dan distibusi input dan output. Faktor yang paling penting adalah ukuran JST itu sendiri. Faktor lainnya adalah distribusi pola-pola input dan hasil-hasil yang diinginkan. Distribusi yang terkelompok (clustered) cenderung untuk mengurangi jumlah data yang diperlukan.

Konfigurasi JST

Merancang JST bisa semudah membalikkan tangan dan dapat juga sesulit kita merancang suatu IC. Pemilihan rancangan JST meliputi pendefinisian tingkah laku simpul-simpulnya, pemilihan prosedur pelatihan, topologi JST, dan nilai parameter-parameter latihan.

Pada umumnya algoritma latihan JST mengikuti bentuk yang sudah standar. Yang harus kita lakukan adalah memilih suatu algoritma yang sesuai dengan kebutuhan aplikasi kita, mengkonfigurasikan JST agar sesuai dengan data yang digunakan, dan melatih JST serta memilih nilai-nilai yang mempengaruhi proses latihan.

Algoritma BP, sebagai contoh, cenderung mampu melakukan estimasi fungsi dan tugas-tugas time series dengan baik. BP juga baik untuk merepresentasikan hubungan yang tidak linier dan rumit dalam bentuk jaringan yang kompak dan efisien. Dengan kata lain, algoritma BP sangat mudah untuk digunakan. Kelemahan BP ini adalah proses belajar yang lambat dan tidak selalu menjamin solusi yang dihasilkannya merupakan yang terbaik.

Setelah memilih JST yang sesuai, tahap selanjutnya adalah mengatur jumlah simpul-simpul di lapisan input dan di lapisan output dengan referensi jumlah data input yang digunakan dan data output yang diinginkan.

Latihan pada JST BP, sebagai contoh, memerlukan suatu nilai awal untuk jumlah simpul-simpul tersembunyinya dan nilai akhir bergantung pada eksperimen selama latihan.

Latihan dan Pengujian JST

Tahap akhir perancangan JST adalah melatih dan menguji JST tersebut. Selama latihan, sebagian besar JST melakukan siklus melalui data secara berulang-ulang dan mengubah bobot-bobot sambungan mereka untuk meningkatkan unjuk kerjanya.

Setiap satu siklus data latihan disebut dengan *epoch* dan JST belajar melalui perubahan keseluruhan bobot-bobot yang diakumulasikan terhadap banyaknya epoch. Proses pelatihan akan berjalan terus sampai nilai-nilai dari bobot-bobot tersebut menyebabkan JST mampu memetakan pola-pola input sesuai hasil-hasil yang diinginkan.

Fakta yang sangat menarik bahwa JST belajar dari data latihan kadang-kadang tidak jelas sampai mendetil sehingga tidak semua yang dipelajarinya berguna. JST dapat mempelajari suatu yang berbeda dari apa yang telah dilatihkan kepadanya. Ia juga dapat mengingat contoh-contoh paket belajar tanpa mempelajari suatu kesamaan yang dimiliki oleh paket belajar tersebut.

Dalam kasus ekstrimnya, suatu JST yang bertindak sebagai *look-up table* dapat mengenali setiap pola latihan. Hasil-hasil yang ditunjukkan hanya memberikan sedikit informasi mengenai kehandalan JST tersebut dalam aplikasinya.

JST hanya berguna bila ia dapat memberikan hasil-hasil yang layak terhadap data yang tidak digunakan dalam data latihannya. Untuk mengukur kemampuan ini, disebut dengan generalisasi, memerlukan pengujian JST dengan paket data bebas.

Latihan adalah suatu proses interaktif dimana si pelatih mencoba suatu konfigurasi JST dan mengevaluasi hasilnya, membuat beberapa perubahan, mencobanya lagi, dan seterusnya sampai mendapatkan hasil yang paling memuaskan.

Pendekatan trial-and-error ini menyebabkan pengembangan JST terhambat, karena kadang menjadi tidak praktis untuk mencoba kombinasi dari beberapa parameter.

Perangkat lunak paralel membuat metode-metode sistematik menjadi layak digunakan dan meningkatkan kecepatan proses komputasi sehingga pembuat JST dapat menulis program dengan sejumlah parameter dan menggali semua kombinasi susunan JST untuk mendapatkan generalisasi yang terbaik.

Parameter-parameter yang mengontrol ukuran JST mempunyai dampak yang besar terhadap generalisasi. Dalam JST BP, jumlah simpul-simpul tersembunyi mempu-

nyai dampak yang besar. JST BP dengan jumlah simpul tersembunyi yang terlalu banyak akan cenderung mengingat semua data latihan; sedangkan JST BP dengan jumlah simpul tersembunyi terlalu sedikit tidak dapat mempelajari masalah yang diberikan kepadanya. Pemilihan jumlah yang layak dilakukan melalui eksperimen.

Tujuan kritis selama latihan adalah mencari suatu JST yang cukup besar untuk mempelajari suatu aplikasi dan cukup kecil untuk dapat melakukan generalisasi.

Yang paling baik adalah JST dengan nilai bobot-bobot paling kecil tapi mampu mengolah data uji secara akurat. Dalam prakteknya, ukuran "ideal" ini berhubungan dengan kuantitas dan kualitas dari data latihan.

Contoh Kasus

Perceptron yang ditemukan oleh Frank Rosenblatt ternyata mempunyai beberapa kelemahan yang kecil tetapi prinsip, yaitu tidak mampu menyelesaikan masalah sederhana dalam sistem digital yang berwujud fungsi 'XOR' (exclusive-OR).

Sanggahan ini disampaikan oleh Marvin Minsky dan Seymour Papert dalam bukunya yang berjudul 'Perceptrons'.

Hal ini dikarenakan Perceptron satu-lapis hanya mampu menyelesaikan masalah-masalah yang bersifat terpisah secara linier (linearly separable).

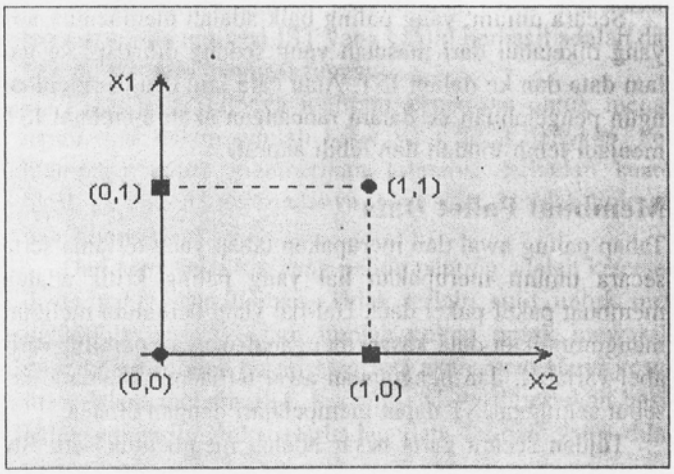
Untuk masalah XOR yang tidak terpisah secara linier sangat sulit untuk diselesaikan oleh Perceptron dan bahkan tidak mungkin. Untuk mempermudah pemahaman masalah XOR ini, berikut ini adalah tabel fungsi XOR dan gambar 3 menunjukkan kondisi fungsi XOR yang tidak terpisah secara linier.

Tabel 1. Fungsi XOR

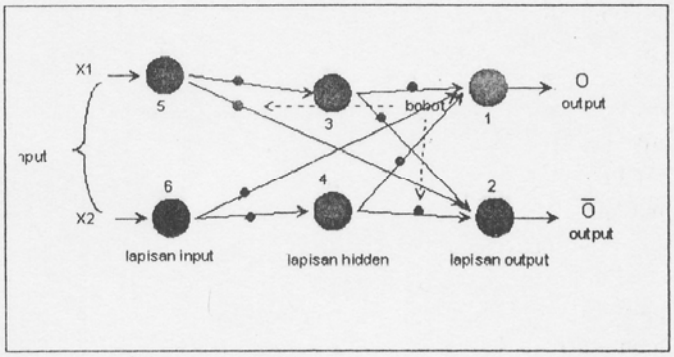
X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Dengan memplot keempat kombinasi tersebut pada koordinat Cartesius tampak bahwa fungsi XOR ini tidak dapat dipisahkan secara linier karena posisi input (0,0) dan (1,1) terletak dalam satu garis.

Untuk memecahkan masalah ini digunakan JST BP dengan simpul tersembunyi sebanyak satu buah, simpul input dua buah dan simpul output satu buah. Gambar 4 memperlihatkan struktur JST BP yang digunakan untuk menyelesaikan masalah XOR ini.



Gambar 3. Fungsi XOR dalam bidang koordinat Cartesius



Gambar 4. Struktur JST BP untuk menyelesaikan masalah XOR

Terlihat di sini dengan urutan dari kiri ke kanan, lapisan input menggunakan dua simpul untuk kombinasi data input 0 dan 1, lapisan tersembunyi menggunakan satu simpul, dan lapisan output menggunakan satu simpul yang akan mengeluarkan output 0 atau 1 untuk setiap kombinasi data input yang diberikan.

Untuk menyelesaikan masalah XOR di atas, maka JST BP ini terlebih dulu dilatih dengan data input seperti di atas. Algoritma belajar JST BP secara singkat telah dijelaskan di atas dan proses minimisasi error serta perubahan nilai bobot-bobot menggunakan rumus matematika yang di sini tidak akan dibahas karena sudah menyangkut masalah teknis (silakan bila ada pembaca *MIKRODATA* yang berminat untuk membahasnya).

Secara umum, pada kondisi awal bobot-bobot sambungan diberikan nilai random yang tidak terlalu besar ataupun terlalu kecil demikian juga dengan nilai tegangan ambangnya. Untuk kasus di atas, parameter-parameter JST yang dibuat tetap adalah:

Tabel 2. Parameter-paramter Latihan JST BP

No.	Parameter	Nilai
1	Jumlah Lapisan	3
2	Jumlah Input	2
3	Jumlah Hidden	2
4	Jumlah Output	2
5	Konstanta Belajar	0.9
6	Iterasi Maksimum	100,000
7	Jumlah Pola	4

Jumlah lapisan JST dan jumlah sel syaraf input, hidden dan output sesuai dengan yang ditunjukkan dalam Gambar 4.

Jumlah sel output 2 buah tersebut digunakan untuk mengeluarkan output XOR beserta komplemennya. Konstanta belajar dipilih sebesar 0.9 (bisa dipilih antara 0.1 sampai 1) dengan tujuan mempercepat proses belajar JST dan iterasi maksimal dibatasi sebanyak 100.000 kali (pada umumnya pada iterasi 6000 - 7000, hasil sudah cukup akurat). Jumlah pola yang dilatihkan sebanyak 4 buah (lihat Tabel 1).

Selama latihan, nilai bobot-bobot tersebut akan berubah sampai nilai minimal jumlah-kuadrat error yang diinginkan telah tercapai. Setelah nilai tersebut tercapai, latihan dihentikan dan JST BP diuji coba untuk menyelesaikan kasus fungsi XOR di atas.

Simulator JST Back Propagation untuk Solusi 'XOR'

Agar proses di atas dapat dipahami dengan baik, disertakan juga program untuk melatih dan menguji JST BP ini untuk kasus fungsi XOR tersebut.

Program ditulis dalam bahasa Turbo C++ versi 3.0. Meskipun program ini sangat sederhana tetapi sudah cukup menunjukkan bagaimana tingkah laku JST BP untuk menangani masalah di atas.

Cara menggunakan Simulator JST BP adalah sebagai berikut:

1. Jalankan program BPSTD5.EXE.
2. Ketikkan pilihah Data Baru (B).
3. Ketikkan nama file parameter BPPAR.
4. Tekan sembarang tombol.
5. Ketikkan nama file input data latihan (IX), BP1IX atau BP2IX.
6. Ketikkan nama file output yang diharapkan (OX), BP1OX atau BP2OX. Ingat file input dan output masing-masing merupakan pasangan (misal :BP1IX) dengan BP1OX)
7. Tekan sembarang tombol, maka selanjutnya JST akan belajar.
8. Bila error sudah mencukupi (di bawah 0.0001), latihan bisa dihentikan dengan menekan tombol <F10>.
9. Selanjutnya adalah pengujian hasil belajar JST.
10. Tekan pilihan Pakai (P).
11. Masukkan data file bobot BPSTD.TOT.
12. Masukkan kombinasi input seperti dalam Tabel 1. Lihat apa yang terjadi.

File BP1 dan BP2 mempunyai perbedaan nilai input dan output yang dimasukkan. Bila menggunakan file BP1, proses belajar sedikit lebih lama atau mencapai error minimal sedikit lebih lama (sifat fungsi Sigmoid), sedang bila menggunakan file BP2 proses belajar akan lebih cepat, boleh dibuktikan!

Dari simulasi program yang telah (Anda) dilakukan, kesimpulan apa yang bisa kita tarik? Apakah JST bisa belajar? Apakah JST memberikan solusi yang mendekati output yang diharapkan? Nah, inilah dasar dari pengembangan komputer pada saat ini dan di masa mendatang sehingga bisa menunjukkan sifat "cerdas",

Daftar Pustaka

1. Beale, R. dan Jackson T., *Neural Computing : An Introduction*, IOP Publishing Ltd., USA, 1990.
2. Fu, LiMin, *Neural Networks in Computer Intelligence*, McGraw-Hill Inc., USA, 1994.
3. Hammerston, Dan, *Working with Neural Networks*, IEEE Spectrum, USA, July 1993.
4. Lafore, Robert, *C Programming Using Turbo C++*, The Waite Group, SAMS, USA, 1990.
5. *Turbo C++ User's Guide*, Borland International, USA, 1991.
6. Wasserman, Phillip D., *Neural Computing : Theory and Practice*, Van Nostrand Reinhold, USA, 1989.

* ***Perwira TNI-AU tugas belajar di Jurusan Teknik Elektro, ITB dan peneliti Sistem Cerdas.***

** ***Staf Peneliti Profesional ISRG, Jurusan Teknik Elektro, ITB dan peneliti sistem cerdas.***

Perubahan karakter pada listing program GAME BUSTERS

Ä	–		³	‖
¿	┐		²	▒
Û	▌		È	╔
À	└		í	=
Ù	┘		»	╗
ß	■		¼	╝
°	▒			
Ú	┌			