BAB II
TINJAUAN PUSTAKA

2.1 Saham

Saham bisa diartikan sebagai surat berharga sebagai bukti penyertaan atau kepemilikan individu maupun institusi dalam suatu perusahan atau lembaga.

Dari berbagai jenis saham yang diperdagangkan bisa dibagi menjadi dua yaitu saham biasa (*common stock*) dan saham preferen (*preferred stock*). Saham biasa yaitu saham yang hanya mempunyai hak untuk memperoleh dividen selama perusahaan memperoleh keuntungan, mempunyai hak bersuara pada rapat umum pemegang saham, dan juga memiliki hak memperoleh sebagian dari kekayaan setelah semua kewajiban terpenuhi.

Sedangkan saham preferen adalah saham yang mempunyai hak preferensi untuk mengajukan usul pencalonan direksi atau komisaris, selain itu mempunyai hak untuk mendapatkan dividend atau bagian kekayaan setelah perusahaan dilikuidasi terlebih dahulu dari saham biasa, tetapi saham ini tidak memiliki hak suara pada rapat umum pemegang saham.

2.2 Konsep Intraday

Konsep Intraday merupakan suatu konsep menentukan harga saham dalam satuan waktu harian. Pada konsep ini harga saham dibagi menjadi empat jenis yang semua terjadi pada satu hari yaitu *Open Price* adalah harga pembukaan saham, *Close Price* adalah harga penutupan saham, *High Price* adalah harga tertinggi yang pernah terjadi, dan *Low Price* adalah harga terendah yang pernah terjadi.

2.3 Week Daily Trading

Week Daily Trading adalah jumlah hari dalam seminggu yang digunakan untuk transaksi bursa saham. *Week Daily Trading* yang dianut oleh pasar dunis dibagi menjadi dua jenis yaitu 7 *days a week daily trading* dan 5 *days a week daily trading.* Indonesia menganut konsep 5 *days a week daily trading* yang

menerapakan 5 hari aktif senin hingga jumat yang digunakan untuk melakukan transaksi bursa saham.

2.4 Konsep Data Time Series

Data *Time Series* merupakan jenis data yang terdiri dari satu objek tetapi meliputi beberapa periode waktu misalnya harian, mingguan, bulanan, tahunan dan lain - lain. Beberapa contoh data *Time Series* misalnya adalah data saham, data nilai tukar uang, data produksi dan lain - lain. Masing - masing data tersebut terkait dengan waktu dan berurutan. Misalnya data saham dari tahun 2005 hingga 2010, data produksi barang pada tahun 2005, dan lain sebagainya.

Data *Time Series* dianggap sangat berguna untuk memprediksi kejadian di masa depan. Karena diyakini pola yang ada pada masa lalu akan terulang kembali di masa datang.

2.5 Peramalan Data *Time Series*

Ada dua macam jenis analisa yang digunakan untuk teknik peramalan yaitu analisis kualitatif dan analisa kuantitatif. Analisa kualitatif adalah teknik peramalan berdasarkan pendapat suatu pihak, dan datanya tidak bisa direpresentasikan secara tegas menjadi suatu nilai. Analisa kuantitatif merupakan teknik peramalan yang berdasarkan data pada masa lalu (*data histories*) dan dapat dibuat dalam bentuk angka yag biasa disebut sebagai data *Time Series* (Jumingan, 2009).

2.6 Algoritma Genetika

Algoritma genetika adalah cabang algoritma evolusi merupakan metode *adaptive* yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam suatu masalah optimasi. Algoritma ini didasarkan pada proses genetik yang ada dalam makhluk hidup yaitu perkembangan dalam sebuah populasi alami, secara lambat laun mengikuti prinsip seleksi alam (*survive*). Dengan meniru teori evolusi ini, algoritma genetika dapat digunakan untuk mencari solusi permasalahan-permasalahan dunia nyata.

Algoritma Genetika menggunakan analogi secara langsung dari kebiasaan yang alami yaitu seleksi alam. Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu-individu, tiap individu mempresentasikan sebuah solusi yang mungkin untuk persoalan yang ada. Individu dilambangkan dengan sebuah nilai *fitness* yang akan digunakan untuk mencari solusi terbaik dari persoalan yang ada. Pertahanan yang tinggi dari individu memberikan kesempatan untuk melakukan reproduksi melalui perkawinan silang dengan individu yang lain dalam populasi tersebut. Individu baru yang dihasilkan dalam hal ini dinamakan keturunan, yang membawa beberapa sifat dari induknya. Sedangkan individu dalam populasi yang tidak terseleksi dalam reproduksi akan mati dengan sendirinya. Dengan jalan ini, beberapa generasi dalam karakteristik yang bagus akan bermunculan dalam populasi tersebut, untuk kemudian dicampur dengan karakter yang lain. Dengan mengawinkan semakin banyak individu, maka akan semakin banyak kemungkinan terbaik yang dapat diperoleh.

Sebelum algoritma genetika dapat dijalankan, maka sebuah kode yang sesuai untuk persoalan harus dirancang. Untuk ini maka titik solusi dalam ruang permasalahan dikoedekan dalam bentuk kromosom atau string yang terdiri dari komponen genetik terkecil yaitu gen. Di dalam algoritma genetika melibatkan beberapa operator , yaitu:

1. Operasi Evolusi yang melibatkan proses seleksi (*selection)*
2. Operasi genetika yang melibatkan operator pindah silang (*crossover*) dan mutasi (*mutation*).

Perbandingan istilah algoritma genetika dan sistem alamiah dapat ditunjukan pada tabel 2.1.

Tabel 2.1 Perbandingan istilah Algoritma Genetika dan sistem alamiah

Sistem Alamiah	**Algoritma Genetik**
Kromosom	String
Gen	Fitur, Karakter, atau detector
Allel	Nilai fitur

Locus	Posisi String
Posisi	String
Fenotip	Set parameter, solusi alternatif, struktur yang di-decode
Epitasis	Non linieritas

2.6.1 Parameter Algoritma Genetika

Beberapa definisi penting dalam algoritma genetika, yaitu :

1. Genotype (Gen) adalah sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini bisa berupa nilai biner, float, integer maupun karakter.
2. Allele adalah nilai dari gen.
3. Kromosom adalah gabungan gen-gen yang membentuk nilai tertentu.
4. Individu menyatakan satu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat
5. Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi.
6. Generasi menyatakan satu-satuan siklus proses evolusi.
7. Nilai Fitness menyatakan seberapa baik nilai dari suatu individu atau solusi yang didapatkan.

Ciri-ciri permasalahan yang dapat dikerjakan dengan menggunakan algoritma genetika adalah (Basuki,2003):

- Mempunyai fungsi tujuan optimalisasi non linear dengan banyak kendala yang juga non linear.
- Mempunyai kemungkinan solusi yang jumlahnya tak berhingga.
- Membutuhkan solusi "*real-time*" dalam arti solusi bisa didapatkan dengan cepat sehingga dapat diimplementasikan untuk permasalahan yang mempunyai perubahan yang cepat seperti optimasi pada pembebanan kanal pada komunikasi seluler.

- Mempunyai *multi-objective* dan *multi-criteria*, sehingga diperlukan solusi yang dapat secara bijak diterima oleh semua pihak.

2.6.2 Komponen-komponen Algoritma Genetika

Terdapat 6 komponen dalam algoritma genetika (Kusuma Dewi, 2005)

1. Penyandian / Representasi Kromosom

   Meliputi penyandian gen dan kromosom. Gen merupakan bagian dari kromosom, gen merupakan wakil dari satu variabel.

   Pada permasalahan penjadwalan praktikum representasi penyandian dapat dituliskan sebagai contoh.

   [4 2 5 1 3 6]

2. Prosedur Inisialisasi

   Inisialisasi kromosom dilakukan secara acak, dan tetap harus memperhatikan domain solusi dan kendala permasalahan yang ada. Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan.

3. Fungsi *Fitness*

   Fungsi *fitness* adalah suatu fungsi yang digunakan oleh algoritma genetika untuk menentukan nilai kecocokan (*fitness*) suatu kromosom. Sebagian besar permasalahan optimasi yang dihadapi, fungsi *fitness* dibangun berdasarkan fungsi objektifnya. Nilai *fitness* tiap kromosom menjadi ukuran kualitas kromosom.

   Kromosom dengan nilai *fitness* tertinggi memiliki kesempatan yang lebih besar untuk terpilih sebagai parent dan diharapkan dapat menghasilkan keturunan-keturunan yang lebih baik dari pada orang tuanya, sehingga didapatkan suatu solusi setelah beberapa generasi.

4. Seleksi

   Menurut Kusumadewi dan Purnomo (2005), tujuan dari seleksi adalah memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit. Metode yang paling sering digunakan untuk seleksi ini

adalah roulette wheels, karena metode ini yang dianggap paling sederhana. Langkah – langkah seleksi menggunakan *roulette wheels* adalah sebagai berikut (Kusumadewi dan Purnomo, 2005) :

1. Hitung total *fitness*
2. Hitung relatif *fitness* tiap individu
3. Hitung *fitness* komulatif
4. Pilih individu yang akan menjadi kandidat untuk *crossover*

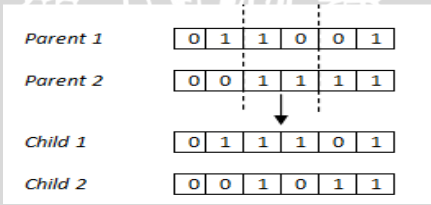
5. Operator Genetika

Pada algorita genetika, menggunakan dua macam operator, yaitu operator *crossover* dan operator mutasi.

1. Crossover

Crossover merupakan salah satu operator Algoritma Genetika yang sangat penting peranannya dalam menghasilkan sebuah solusi yang optimal. *Crossover* dapat dibedakan menjadi tiga jenis cara yang berbeda dalam menentukan titik segmen pertukaran string bit kromosomnya (Tamba 2004). Salah satunya adalah *Two-point Crossover* adalah sebuah cara penentuan titik segmen pertukaran string bit kromosom. dengan memilih dua titik potong pertukaran. Titik potong dipilih secara acak, kemudian bagian pertama dan ketiga dari *parent* 1 digabungkan dengan bagian kedua *parent* 2.

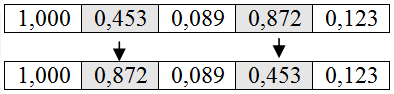
Skema *two-point crossover* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Skema *two – point crossover*

2. Mutasi

Mengubah nilai gen individu induk untuk mendapatkan individu anak. Berikut gmbar 2.2 yang menunjukkan proses mutasi *exchange mutation.*



Gambar 2.2 Proses Mutasi *exchange mutation*

Proses mutasi ini dilakukan dengan cara memilih dua gen secara acak kemudian posisi gen pertama ditukar dengan posisi gen kedua (Yento, 2008).

6. Penentuan Parameter

Nilai parameter ditentukan berdasarkan permasalahan yang akan dipecahkan. Parameter kontrol algoritma genetika yaitu : ukuran populasi, peluang *crossover*, dan peluang mutasi.

2.7 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) atau *neural network* adalah suatu metode komputasi yang meniru sistem jaringan syaraf biologis. Metode ini menggunakan elemen perhitungan *non-linier* dasar yang disebut *neuron* yang diorganisasikan sebagai jaringan yang saling berhubungan, sehingga mirip dengan jaringan syaraf manusia. Jaringan syaraf tiruan dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi karena proses pembelajaran.

Layaknya *neuron* biologi, JST juga merupakan sistem yang bersifat *fault tolerant* dalam 2 hal. Pertama, dapat mengenali sinyal *input* yang agak berbeda dari yang pernah diterima sebelumnya. Sebagai contoh, manusia sering dapat mengenali seseorang yang wajahnya pernah dilihat dari foto atau dapat mengenali seseorang yang wajahnya agak berbeda karena sudah lama tidak menjumpainya. Kedua, tetap mampu bekerja meskipun beberapa *neuron*-nya tidak mampu bekerja dengan baik. Jika sebuah *neuron* rusak, *neuron* lain dapat dilatih untuk menggantikan fungsi *neuron* yang rusak tersebut.

Jaringan syaraf tiruan, seperti manusia, belajar dari suatu contoh karena mempunyai karakteristik yang adaptif, yaitu dapat belajar dari data-data sebelumnya dan mengenal pola data yang selalu berubah. Selain itu, JST merupakan sistem yang tak terprogram, artinya semua keluaran atau kesimpulan

yang ditarik oleh jaringan didasarkan pada pengalamannya selama mengikuti proses pembelajaran / pelatihan.

Hal yang ingin dicapai dengan melatih JST adalah untuk mencapai keseimbangan antara kemampuan mengingat dan generalisasi. Yang dimaksud kemampuan mengingat adalah kemampuan JST untuk mengambil kembali secara sempurna sebuah pola yang telah dipelajari. Kemampuan generalisasi adalah kemampuan JST untuk menghasilkan respons yang bisa diterima terhadap pola-pola *input* yang serupa (namun tidak identik) dengan pola-pola yang sebelumnya telah dipelajari. Hal ini sangat bermanfaat bila pada suatu saat ke dalam JST itu dimasukkan informasi baru yang belum pernah dipelajari, maka JST itu masih akan tetap dapat memberikan tanggapan yang baik, memberikan keluaran yang paling mendekati (Puspitaningrum, 2006).

Jaringan syaraf tiruan berkembang secara pesat pada beberapa tahun terakhir. Jaringan syaraf tiruan telah dikembangkan sebelum adanya suatu komputer konvensional yang canggih dan terus berkembang walaupun pernah mengalami masa vakum selama beberapa tahun.

JST menyerupai otak manusia dalam dua hal, yaitu:

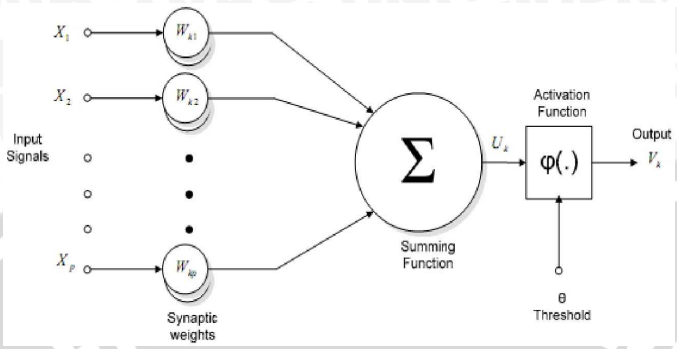
1. Pengetahuan diperoleh jaringan melalui proses belajar.
2. Kekuatan hubungan antara sel saraf(*neuron*) yang dikenal sebagai bobot - bobot sinaptik digunakan untuk menyimpan pengetahuan.

Menurut (Siang, 2004) JST ditentukan oleh 3 hal:

1. Pola hubungan antara *neurons* yang disebut arsitektur jaringan
2. Metode untuk menentukan bobot penghubung yang disebut dengan metode pembelajaran atau *learning*
3. Fungsi aktivasi, yaitu fungsi yang digunakan untuk menentukan keluaran suatu *neurons*.

2.7.1 Model *Neuron*

Dalam sel syaraf terdapat tiga bagian, yaitu: fungsi penjumlah (*summing function*), fungsi aktivasi (*activation function*), dan keluaran(*output*).



Gambar 2.3 Model *Neuron*

Pada gambar 2.3 bisa dilihat bahwa *neuron* buatan ,diatas mirip dengan sel *neuron* biologis. Informasi (*input*) akan dikirim ke *neuron* dengan bobot tertentu. *Input* ini akan diproses oleh suatu fungsi yang akan menjumlahkan nilai-nilai bobot yang ada. Hasil penjumlahan kemudian akan dibandingkan dengan suatu nilai ambang (*threshold*) tertentu melalui fungsi aktivasi setiap *neuron*. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka *neuron* tersebut akan diaktifkan, jika tidak, maka *neuron* tidak akan diaktifkan. Apabila *neuron* tersebut diaktifkan, maka *neuron* tersebut akan mengirimkan *output* melalui bobot-bobot *output*-nya ke semua *neuron* yang berhubungan dengan neuron tersebut.

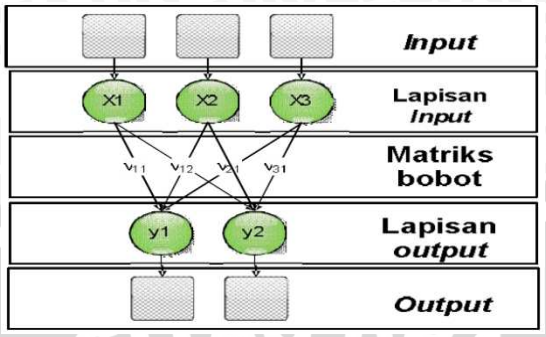
2.7.2 Arsitektur Jaringan

JST memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai aplikasi. Arsitektur JST tersebut, antara lain (Kusumadewi, 2003):

1. Jaringan layar tunggal *(single layer network*)

Jaringan dengan lapisan tunggal terdiri dari 1 *layer input* dan 1 *layer output*. Setiap *neuron*/unit yang terdapat di dalam lapisan/*layer input* selalu terhubung dengan setiap *neuron* yang terdapat pada *layer output*. Terlihat pada Gambar 2.4, jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi. Contoh

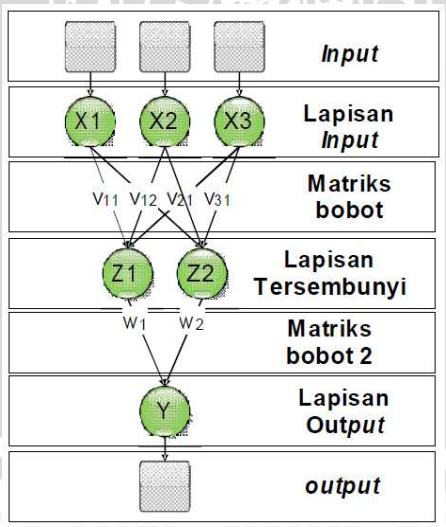
algoritma JST yang menggunakan metode ini yaitu :A*DALINE*, *Hopfield*, *Perceptron*.



Gambar 2.4 Arsitektur *layer* tunggal

2. Jaringan layar banyak (multi *layer* network)

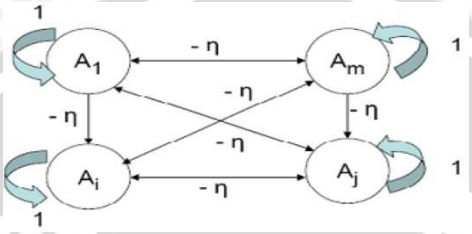
Jaringan dengan lapisan jamak memiliki ciri khas tertentu yaitu memiliki 3 jenis *layer* yakni *layer input*, l*ayer output*, dan juga *layer* tersembunyi seperti pada Gambar 2.5. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan jaringan dengan lapisan tunggal. Namun, proses pelatihan sering membutuhkan waktu yang cenderung lama. Contoh algoritma JST yang menggunakan metode ini yaitu : *MADALINE*, *backpropagation*, *Neocognitron.*



Gambar 2.5 Arsitektur *layer* jamak

3. Jaringan dengan lapisan kompetitif(*competitive layer network*)

Pada jaringan ini sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif seperti yang terlihat pada gambar 2.6. Contoh algoritma yang menggunakan metode ini adalah LVQ.



Gambar 2.6. Algoritma Metode LVQ

2.7.3 Fungsi Aktivasi

Fungsi ini diawali dengan fungsi penjumalan yang berguna untuk menjumlahkan masukan-masukan yang diterima berdasarkan bobot dari masukan tersebut. Masukan yang diterima dikalikan dengan bobotnya lalu hasil seluruh perkalian tersebut dijumlahkan. Fungsi penjumlahan dapat didefinisikan melalui persamaan 2.1.

$$net = \sum_{i=0}^{n} wixi \qquad (2.1)$$

yang dalam hal ini net adalah hasil keluaran dari fungsi penjumlahan, w_i menyatakan bobot koneksi masukan ke-i, dan x_i menyatakan masukan pada bobot tersebut.

Sedangkan untuk fungsi aktivasi adalah fungsi yang menentukan keluaran sebuah neuron dari hasil penjumlahan yang didapat melalui Persamaan 2.1. Fungsi aktivasi ini dilambangkan dengan notasi σ. Terdapat beberapa jenis fungsi aktivasi, yaitu(Kusumadewi, 2003):

a. Fungsi Linier

   Nilai keluaran neuron sama dengan hasil penjumlahan yang didapat melalui Persamaan 2.2.

$$\sigma(net) = \text{net} \qquad (2.2)$$

b. Fungsi Ambang(*threshold*)

Nilai keluaran neuron dikeluarkan secara diskrit jika nilai hasil penjumlahan dari persamaan 2.1 melebihin nilai ambang tertentu. Dalam penggunaan funsi ambang, biasanya batasan nilai tersebut adalah nol karena nilai batasan telah ikut diperhitungkan dari adanya bobot bias yang dimiliki unit neuron. Fungsi ambang dapat didefinisikan sebagai Persamaan 2.3

$$\sigma(net) = \begin{cases} 1 \text{ untuk } net > 0 \\ 0 \text{ untuk } net \leq 0 \end{cases} \quad (2.3)$$

c. Fungsi *Sigmoid*

Nilai keluaran dipetakan dari rentang (-∞, +∞) menjadi bilangan riil dengan rentang antara [0,1] dengan menggunakan fungsi *sigmoid biner*. Sedangkan nilai keluaran yang dipetakan menjadi bilangan riil dengan rentang antara [-1,1] menggunakan fungsi *sigmoid bipolar*. Fungsi ini dipilih agar pembelajaran yang menggunakan turunan dari fungsi aktivasi dapat menggunakan fungsi kontinu. Fungsi *sigmoid biner* dapat didefinisikan melalui persamaan 2.4.

$$\sigma(net) = \frac{1}{1+e^{-net}} \quad (2.4)$$

sedangkan *sigmoid bipolar* dapat didefinisikan melalui persamaan 2.5.

$$\sigma(net) = \frac{1-e^{-net}}{1+e^{-net}} \quad (2.5)$$

2.8 Metode *Backpropagation*

Kelemahan ANN yang terdiri dari lapisan tunggal membuat perkembangan ANN menjadi terhenti pada sekitar tahun 1970 an. Penemuan *backpropagation* yang terdiri dari beberapa lapisan membuka kembali cakrawala baru. Terlebih setelah berhasil ditemukannya berbagai aplikasi yang dapat diselesaikan dengan metode *backpropagation*, membuat ANN semakin diminati orang, sekalipun dengan lapisan tunggal memiliki keterbatasan dalam pengenalan pola 15 Kelemahan dengan lapisan tunggal, dapat ditanggulangi dengan

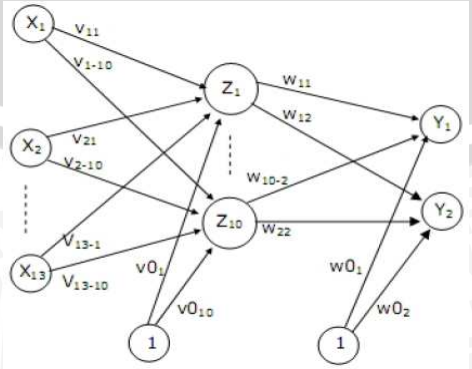
menambahkan satu/beberapa lapisan tersembunyi diantara lapisan masukan dan keluaran. Meskipun penggunaan lebih dari satu lapisan tersembunyi memiliki kelebihan manfaat untuk beberapa kasus, tapi pelatihannya memerlukan waktu yang lama. Maka umumnya orang mulai mencoba dengan sebuah lapisan tersembunyi lebih dahulu. Seperti halnya model ANN lain, *backpropagation* melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa (tetapi tidak sama) dengan pola yang dipakai selama pelatihan.

2.8.1 Arsitektur Jaringan Metode *Backpropagation*

Ada beberapa pola arsitektur Jaringan syaraf tiruan, yaitu terdiri dari

1. 2 (dua) lapisan, yaitu lapisan masukan/*input* dan lapisan *output*.
2. 3 (tiga) lapisan, yaitu lapisan masukan/*input*, satu lapisan hidden dan lapisan *output*.
3. Lebih dari 3 (tiga) lapisan, yaitu lapisan masukan atau *input*, beberapa lapisan hidden dan lapisan *output*.

Pada Gambar 2.7 ditunjukkan salah satu jenis contoh arsitektur jaringan *backpropagation*, yaitu dengan model 3 (tiga) lapisan. Satu kelompok lapisan *input*, satu lapisan *hidden layer* dan satu lapisan *output*. Sedangkan variabel yaitu X1 sampai dengan X13 adalah lapisan *input*. Artinya ada 13 variabel yang digunakan untuk data *input*. Variable Z1 hingga Z10 dan lapisan keluaran/*output* terdiri atas 2 (dua) sel syaraf dengan variable Y1 dan Y2. Arsitektur jaringan *Backpropagation* ditunjukkan pada Gambar 2.7.



Gambar 2.7 Arsitektur Jaringan *Backpropagation*

Pelatihan pada dalam *backpropagation* adalah sebagai berikut(Siang, 2005):

1. Inisialisasi bobot

   Bobot awal ditentukan secara acak dengan nilai sekecil mungkin.

2. Untuk setiap data pelatihan , lakukan langkah-langkah sebagai berikut:

   a. Setiap unit *input* (x_i, i=1,2,...,n) menerima sinyal dan menyalurkan sinyal ini ke semua unit lapisan tersembunyi.

   b. Mengalikan nilai *input* dan bobot sinyal *input*, lalu dijumlahkan di setiap unit tersembunyi (z_j, j=1,2,...,p) dengan persamaan 2.6.

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (2.6)$$

   Keterangan:

   z_in_j = input jaringan ke *hidden layer* z

   v_{0j} = bobot awal bias ke *hidden layer* j

   n = jumlah unit input

   v_{ij} = bobot sinyal *input* menuju *hidden layer* j

   x_i = nilai *input*

3. Menghitung *output* menggunakan fungsi aktivasi *sigmoid bipolar*, yaitu dengan menggunakan persaman 2.7.

$$z_j = f(z_{inj}) = \frac{1-e^{-net}}{1+e^{-net}} \quad (2.7)$$

   Keterangan:

   z_j = unit *hidden layer* j

   z_in_j = *input* jaringan ke *hidden layer z*

   lalu mengirimkan sinyal tersebut ke semua unit di lapisan atasnya(lapisan *output*)

4. Setiap unit *output* (y_k; k=1,2,...,m) akan menjumlahkan bobot sinyal *output* dengan persaman 2.8.

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad (2.8)$$

   Keterangan:

   y_in_k = *input* jaringan ke lapisan *output*

w_{0k} = bobot awal bias ke *output*

p = jumlah unit tersembunyi

z_j = unit tersembunyi j

w_{jk} = bobot menuju *output*

Gunakan fungsi aktivasi untuk menghitung sinyal *output*, ditunjukkan pada Persamaan 2.9.

$$y_k = f(y_{ink}) = \frac{1}{1+e^{-yin_k}} \quad (2.9)$$

Keterangan:

y_k = unit *output*

y_{ink} = input jaringan ke *output*

5. Tiap unit keluaran (y_k , $k = 1,2,3 ,....,m$) menerima pola target yang saling berhubungan pada masukan pola pelatihan, hitung kesalahan informasinya seperti yang ditunjukan pada persamaan 2.10.

$$\delta_k = (t_k - y_k)\, f^1(y_{ink}) = (t_k - y_k)\, y_k(1-y_k) \quad (2.10)$$

Suku perubahan bobot w_{jk} dihitung dengan persamaan 2.11 (digunakan untuk memperbaharui w_{jk} dengan laju pelatihan α).

$$\Delta w_{jk} = \alpha\, \delta_k\, zj \quad (2.11)$$

Hitung perubahan biasnya (digunakan untuk memperbaharui w_{ok}), dan kirimkan δ_k ke unit-unit pada lapisan di bawahnya.

6. Setiap unit lapisan tersembunyi (dari unit ke-1 hingga ke p; i=1...n; k=1...m) dilakukan perhitungan kesalahan lapisan tersembunyi(δ_j). δ_j kemudian digunakan untuk menghitung besar koreksi bobot dan bias (Δw_{ij} dan ΔV_{jo}) antara lapisan *input* dan lapisan tersembunyi.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.12)$$

Hasil dari perhitungan pada persamaan 2.12 dikalikan dengan turunan fungsi aktivasinya untuk menghitung informasi kesalahannya, sehingga untuk mencari δ_j dapat digunakan persamaan 2.13.

$$\delta_j = \delta\_in_j\, f'(\delta\_in_j) \quad (2.13)$$

Suku perubahan bobot Δv (yang digunakan untuk perbaikan bobot v_{ij}) di hitung dengan persamaan 2.14.

$$\Delta v_j = \alpha\, \delta_j x_i \quad (2.14)$$

Perubahan bias (untuk memperbaiki v_{jo}) dihitung dengan menggunakan persamaan 2.15.

$$\Delta v_{jo} = \alpha \, \delta_j \qquad (2.15)$$

7. Tiap unit keluaran (y_k , k = 1,2,3, ... , m) dilakukan pembaharuan bias dan bobotnya (j=0,1,2,3,.... ,p) sehingga menghasilkan bobot dan bias baru dan persamaan 2.16.

$$W_{kj}(\text{baru}) = w_{kj} (\text{lama}) + \Delta w_{kj} \qquad (2.16)$$

Demikian juga untuk setiap unit tersembunyi mulai dari unit ke-1 sampai dengan unit ke-p dilakukan pembaharuan bias dan bobotnya dengan persamaan 2.17.

$$V_{ji} (\text{baru}) = v_{ji}(\text{lama}) + \Delta v_{ji} \qquad (2.17)$$

8. Uji kondisi berhenti (akhir iterasi).

Setelah proses pelatihan, tahap pengujian data dilakukan dengan menggunakan tahap perambatan maju (*feedforward*) yang sama seperti tahap pelatihan. Inisialisasi bobot pada tahap pengujian menggunakan bobot yang dihasilkan pada proses *training* sebelumnya (Kusumadewi, 2003).

2.8.2 Pengukuran Error

Perhitungan kesalahan merupakan pengukuran bagaimana jaringan dapat belajar dengan baik. Kesalahan pada keluaran dari jaringan merupakan selisih antara keluaran aktual (*current output*) dan keluaran target (*desired output*). Langkah berikutnya adalah menghitung nilai SSE *(Sum Square Error)* yang merupakan hasil penjumlahan nilai kuadrat *error neuron*1 dan *neuron*2 pada lapisan *output* tiap data, dimana hasil penjumlahan keseluruhan nilai SSE akan digunakan untuk menghitung nilai RMSE *(Root Mean Square Error)* tiap iterasi (Kusumadewi, 2003).

1. *Sum Square Error* (SSE).

SEE dihitung sebagai berikut :

1. Lapisan prediksi atau keluaran model dihitung untuk masukan pertama.
2. Selisih antara nilai luar prediksi (D) dan nilai target (T) atau sinyal latihan dihitung untuk setiap keluaran.

3. Setiap keluaran dikuadratkan kemudian seluruhnya dihitung.

$$\text{SSE} = \sum_{i=1}^{N} (\text{D}_i - \text{T}_\text{i})^2 \tag{2.18}$$

2. *Root Mean Square Error* (RMS *Error*)

Dihitung sebagai berikut:

1. SSE dihitung terlebih dahulu seperti yang ditunjukkan pada persamaaan 2.2.
2. Hasilnya dibagi dengan perkalian antara banyaknya data pada latihan dan banyaknya keluaran, kemudian diakarkan dan diperoleh persamaan 2.3.

$$\text{RMSE} = \sqrt[2]{\frac{SSE}{N*K}} \tag{2.19}$$

Dimana:

RMSE = *Root Mean Square Error*

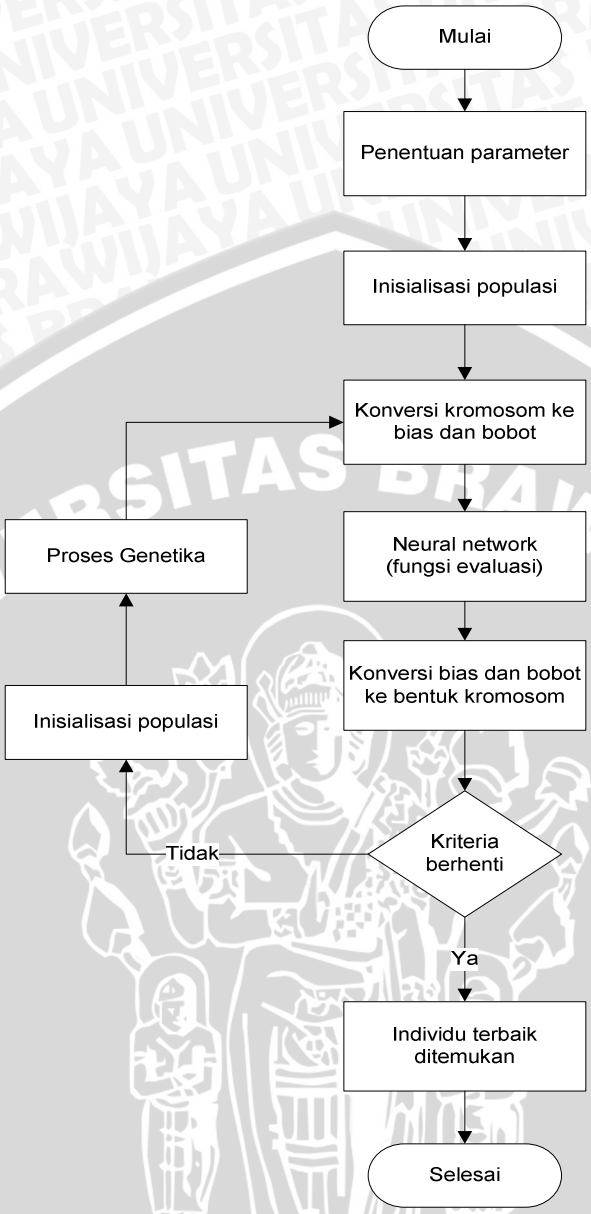
SSE = *Sum Square Error*

N = Banyaknya data pada latihan

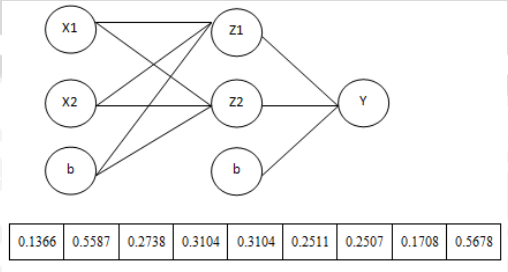
K = Banyaknya keluaran/*output*

2.9 Penerapan Algoritma Genetika Pada JST

Algoritma genetika dapat diterapkan pada JST untuk menggantikan metode pelatihan standard dalam pencarian bobot yang optimal pada JST. Hal yang perlu diperhatikan dalam penerapan algoritma genetika pada JST yaitu bagaimana melakukan pengkodean kromosom dari bobot-bobot yang ada pada JST ke dalam bentuk individu pada algoritma genetika. Solusi permasalahan tersebut yaitu dengan cara menggabungkan semua bobot menjadi suatu string yang mewakili suatu individu pada algoritma genetika. Representasi dari gen-gen pada individu bisa menggunakan *real-number encoding* (Noertjahyana, 2002). *Flowchart* penerapan algoritma genetika pada jaringan syaraf tiruan dapat dilihat pada gambar 2.8 dan untuk *skema* Pengkodean Kromosom untuk pelatihan jaringan syaraf tiruan (sumber : Suyanto ,2005) dapat dilihat pada gambar 2.9.



Gambar 2.8 *Flowchart* penerapan algoritma genetika untuk melatih jaringan syaraf tiruan



Gambar 2.9 *skema* Pengkodean kromosom untuk pelatihan jaringan syaraf tiruan

Langkah berikutnya adalah membuat fungsi evaluasi yang akan dijadikan sebagai nilai *fitness* yang menentukan keoptimuman suatu individu. Untuk masalah pelatihan JST menggunakan algoritma genetika, nilai *fitness* didapatkan dari *invers* dari *mean-square deviation* (*delta)* antara *output* target dengan *output* yang dihasilkan jaringan syaraf tiruan. Dimana *delta* dapat dihitung menggunakan persamaan 2.4 (Suyanto, 2005).

$$delta = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(d(i) - y(i))^2} \qquad (2.20)$$

2.10 Normalisasi dan Denormalisasi

Sebelum data diproses sebagai masukan sistem maka sebelumnya data juga perlu dilakukan normalisasi dengan menyesuaikan *range output* fungsi aktivasi. Misalkan menggunakan fungsi aktivasi *sigmoid* maka data masukan harus dirubah dengan *range*[0...1], namun karena merupakan fungsi kontinu maka nilai dan 0 dan 1 tidak pernah tercapai. Maka dari itu *range* dirubah menjadi [0.1...0.9], sehigga dapat dirumuskan pada persamaan 2.20 (Siang, 2004).

$$\text{Nilai baru} = \left(\frac{nilai\ lama - a}{b - a} \text{ x } 0.8\right) + 0.1 \qquad (2.21)$$

Sedangkan untuk mengembalikan data yang telah dinormalisasi ke nilai awal dapat dilakukan denormalisasi seperti pada persamaan 2.6

$$\text{x} = \frac{(b-a)(x' - 0.1)}{0.8} + a \qquad (2.22)$$

Keterangan

x : input

a : nilai minimum dari data

b : nilai maksimum dari data

x' : *input* yang telah dinormalisasi

0.8 : jarak dari skala 0.1-0.9

2.11 Pemilihan Jumlah *Hidden Layer* Dan Neuron

Kendala yang sering dihadapi dalam jaringan syaraf tiruan adalah menentukan berapa *hidden layer* yang dibutuhkan dan berapa banyak jumlah neuron pada *layer* tersebut. Menurut pernyataan Jeff Heaton (Heaton, 2010) menggunakan satu *hidden layer* sudah cukup untuk menyelesaikan berbagai macam kasus. Sedangkan untuk menentukan jumlah *neuron* pada *hidden layer* harus memenuhi aturan bahwa jumlah *neuron* disarankan 2/3 dari jumlah *input* ditambah dengan jumlah *output* dan jumlah *neuron* kurang dari dua kali jumlah neuron *input*.