PERBANDINGAN MODEL *BACKPROPAGATION* STANDAR DENGAN *BACKPROPAGATION* YANG DIMODIFIKASI DENGAN PENAMBAHAN MOMENTUM
(studi kasus : peramalan harga saham harian Astra International, Tbk)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Sains dalam bidang Matematika

oleh :
WAHYU HENDRO SULAKSONO
0210940044-94

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

2008

LEMBAR PENGESAHAN SKRIPSI

PERBANDINGAN MODEL *BACKPROPAGATION* STANDAR DENGAN *BACKPROPAGATION* YANG DIMODIFIKASI DENGAN PENAMBAHAN MOMENTUM (studi kasus : peramalan harga saham harian Astra International, Tbk)

oleh :

WAHYU HENDRO SULAKSONO
0210940044-94

Setelah dipertahankan di depan majelis penguji
Pada tanggal : 2 Januari 2008
Dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Sains dalam bidang Matematika

Pembimbing I	Pembimbing II
Dra. Trisilowati, M.Sc	**Syaiful Anam, S.Si, MT**
NIP. 131 837 955	**NIP. 132 300 237**

Mengetahui,
a.n. Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya
Sekretaris

Dra. Ani Budi Astuti, M.Si
NIP. 131 993 385

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**:**	**WAHYU HENDRO SULAKSONO**
NIM	**:**	**0210940044**
Jurusan	**:**	**MATEMATIKA**
Judul Skripsi	**:**	**PERBANDINGAN MODEL *BACKPROPAGATION* STANDAR DENGAN *BACKPROPAGATION* YANG DIMODIFIKASI DENGAN PENAMBAHAN MOMENTUM (studi kasus : peramalan harga saham harian Astra International, Tbk)**

Dengan ini menyatakan bahwa :

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

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 2 Januari 2008
Yang menyatakan,

Wahyu Hendro Sulaksono
NIM. 0210940044

PERBANDINGAN MODEL *BACKPROPAGATION* STANDAR DENGAN *BACKPROPAGATION* YANG DIMODIFIKASI DENGAN PENAMBAHAN MOMENTUM
(studi kasus : peramalan harga saham harian Astra International, Tbk)

ABSTRAK

Jaringan Syaraf Tiruan (JST) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf pada manusia, yang dapat dilatih untuk mengenali suatu obyek yang memiliki pola tertentu. Salah satu model yang banyak digunakan adalah model *backpropagation*. Pada model *backpropagation* digunakan metode *gradient-descent* dalam proses perubahan bobotnya. Kelemahan model *backpropagation* standar adalah proses pelatihannya membutuhkan waktu yang cukup lama untuk mencapai kekonvergenan dari nilai bobot baru yang meminimumkan kesalahan. Dalam Skripsi ini dibandingkan antara model *backpropagation* standar dengan model *backpropagation* yang dimodifikasi dengan penambahan momentum, dimana nilai laju pembelajaran/*learning rate*-nya (α) bersifat dinamis.

Dari hasil implementasi peramalan pada data saham harian Astra International, Tbk, diketahui bahwa performansi model *backpropagation* standar cukup baik (keakuratan hasil peramalanmya), dan menjadi lebih baik setelah ditambahkan dengan konstanta momentum tertentu pada proses perubahan bobotnya.

Kata kunci : Jaringan Syaraf Tiruan (JST), *backpropagation*, metode *gradient-descent*, momentum, *learning rate*.

THE COMPARISON BETWEEN STANDARD *BACKPROPAGATION* AND *BACKPROPAGATION* WITH MOMENTUM MODELS
(case study : stock exchange of Astra International, Tbk prediction)

ABSTRACT

Artificial Neural Network (ANN) is an information processing system similar to human nervous system which is capable to be trained to characterize an object with certain pattern. One of the models widely used is backpropagation. In this model, gradient-descent method is used on its weight alteration process. The weakness of standart backpropagation model is its relatively slow training process in achieving the convergence of new weight value which minimizes the error. In this final project, the standard backpropagation model is compared with its modification (backpropagation momentum model) using dinamical learning rate (α).

The result of backpropagation implementation in predicting the stock exchange of Astra International, Tbk shows a well performance of standard backpropagation model related to its prediction result accuracy, and it becomes better with certain momentum constant addition on its weight alteration process.

Keywords : Artificial Neural Network (ANN), backpropagation, gradient-descent method, momentum, learning rate.

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Alhamdulillahirobbil ’alamin, puji syukur senantiasa penulis panjatkan kepada Allah SWT yang telah memberikan rahmat, karunia serta hidayah-Nya sehingga penulis dapat menyelesaikan Skripsi yang berjudul **” Perbandingan Model *Backpropagation* Standar dengan *Backpropagation* yang Dimodifikasi dengan Penambahan Momentum”.** Sholawat serta salam tetap tercurah kepada junjungan kita Nabi Besar Muhammad SAW. Tak lupa penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Dra. Trisilowati, M.Sc selaku dosen pembimbing I atas petunjuk, bimbingan, dukungan dan kesabarannya selama penyusunan Skripsi ini.
2. Syaiful Anam, S.Si, MT selaku dosen pembimbing II atas bimbingan, masukan dan saran serta kesabarannya selama penulis menyelesaikan Skripsi ini.
3. Dr. Agus Suryanto, M.Sc selaku Ketua Jurusan Matematika atas semua bantuan serta petunjuknya dalam kelancaran kelulusan penulis.
4. Drs. Hery Subagyo, M.Kes, Drs. Imam Nurhadi, MT, Drs. Marji, MT selaku dosen penguji atas masukan dan saran selama penyelesaian Skripsi ini.
5. Serta semua pihak yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa manusia mempunyai kekhilafan dan tak lepas dari kesalahan, serta penulis juga menyadari bahwa penulisan Skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, penulis membuka diri atas segala kritik dan saran yang bersifat membangun demi perbaikan selanjutnya.

Akhir kata, semoga Skripsi ini dapat bermanfaat bagi penulis khususnya dan semua pihak pada umumnya, serta dapat menambah wawasan dan menjadi sumbangan untuk ilmu pengetahuan.

Wassalamualaikum Wr. Wb.

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jaringan Syaraf Tiruan (JST) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf pada manusia, yang dapat dilatih untuk mengenali suatu obyek yang memiliki pola tertentu. Dengan pelatihan yang terstruktur, JST yang telah terlatih dapat mengenali obyek tertentu sekalipun sudah ada sedikit perubahan.

JST banyak diaplikasikan dalam berbagai cabang ilmu, dari kasus yang membutuhkan proses pelatihan yang sederhana sampai yang rumit, misalnya untuk pengenalan pola, *signal processing*, kompresi data, penguraian data, peramalan dan masih banyak lagi penerapan yang lain.

Salah satu model yang paling banyak digunakan dalam JST adalah *backpropagation* dengan metode yang digunakan untuk proses pencarian bobot barunya adalah metode *gradient-descent*. Proses pelatihan (*training*) dalam model ini adalah pelatihan terawasi (*supervised learning*). Sejauh ini, hasil yang didapat dalam banyak aplikasi cukup memuaskan. Namun ternyata model dengan metode *gradient-descent* ini masih mempunyai kelemahan, yaitu prosesnya membutuhkan waktu yang cukup lama untuk mencapai kekonvergenan dari nilai bobot baru yang meminimumkan kesalahan yang terjadi. Untuk mengatasi kelemahan tersebut, telah banyak dikembangkan variasi-variasi atau modifikasi dari metode yang digunakan pada proses perubahan bobotnya. Salah satunya adalah dengan penambahan momentum pada nilai bobot baru yang dicari. Penambahan momentum ini dilakukan untuk mempercepat kekonvergenan bobot yang dicari, sehingga mempersingkat waktu proses.

Dalam Skripsi ini akan diulas lebih jauh mengenai model JST *backpropagation* standar dengan nilai laju pembelajaran α yang dinamis dan mengenai pengaruh penambahan momentum terhadap kinerja *backpropagation* standar, serta akan dibuat program untuk mempermudah pengimplementasian model *backpropagation* dalam prediksi atau peramalan harga penutupan data perdagangan saham harian Astra International, Tbk.

1.2 Rumusan Masalah

Permasalahan yang akan dibahas dalam Skripsi ini adalah :

- Bagaimana implementasi metode *gradient-descent* pada model *backpropagation* ?
- Bagaimana mencari nilai laju pembelajaran (*learning rate*) untuk mengoptimalkan proses perubahan bobot pada model *backpropagation* ?
- Bagaimana perbandingan kinerja Jaringan Syaraf Tiruan menggunakan model *backpropagation* standar dengan model *backpropagation* yang telah dimodifikasi dengan penambahan momentum dari hasil analisa secara numerik pada aplikasi peramalan harga penutupan dari data perdagangan saham harian Astra International, Tbk ?

1.3 Batasan Masalah

Adapun batasan-batasan masalah dalam Skripsi ini adalah sebagai berikut :

- Arsitektur jaringan yang digunakan adalah jaringan lapis jamak dengan satu lapisan tersembunyi.
- Fungsi aktivasi yang digunakan adalah fungsi sigmoid biner pada lapisan tersembunyi dan fungsi identitas (*purelin*) pada lapisan keluaran.
- Sebagai kriteria penghentian adalah nilai *Mean Square Error* (MSE) atau jumlah iterasi maksimal yang tercapai.

1.4 Tujuan

Tujuan yang ingin dicapai pada penulisan Skripsi ini adalah :

- Mengetahui bagaimana implementasi metode *gradient-descent* pada model *backpropagation.*
- Mencari nilai laju pembelajaran (*learning rate*) yang dapat mengoptimalkan proses perubahan bobot dalam model *backpropagation.*
- Membandingkan kinerja model *backpropagation* standar dengan model *backpropagation* yang telah dimodifikasi dengan penambahan momentum dengan melihat hasil dari analisa secara numerik pada aplikasi peramalan harga penutupan dari data perdagangan saham harian Astra International, Tbk.

BAB II
TINJAUAN PUSTAKA

Untuk mempermudah pemahaman tentang isi dari Skripsi ini, akan diuraikan terlebih dahulu teori-teori yang mendasari penulisan ini. Adapun teori yang dapat dijadikan sebagai dasar atau acuan tersebut antara lain adalah mengenai matriks beserta operasi dasarnya, vektor, masalah optimasi, Jaringan Syaraf Tiruan (JST), algoritma *Backpropagation,* metode g*radient-descent* (*steepest-descent*) serta teori dasar tentang pasar modal sebagai sarana dalam transaksi saham, dimana data mengenai saham harian Astra International, Tbk akan dijadikan sebagai aplikasi peramalan.

2.1 Konsep Dasar

2.1.1 Matriks

Definisi 2.1 (matriks)

Matriks adalah suatu susunan bilangan berbentuk segiempat. Bilangan-bilangan tersebut disebut entri atau anggota dari matriks. Pada umumnya, suatu matriks dituliskan dengan huruf besar *A*, *B*, *C* dan sebagainya. Misalkan *A* adalah matriks berukuran $m \times n$ yang anggota-anggotanya adalah a_{ij}, $i = 1, 2, ..., m$ dan $j = 1, 2, ..., n$, maka bentuk penulisan matriksnya sebagai berikut

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{2} & \cdots & a_{mn} \end{pmatrix},$$

dimana *m* menyatakan jumlah baris dan *n* jumlah kolom pada matriks *A*. Notasi dari matriks *A* adalah $A = \left(a_{ij}\right)$.

Operasi-operasi dasar pada matriks yang sering digunakan adalah operasi penjumlahan dan perkalian matriks.

Definisi 2.2 (operasi penjumlahan matriks)

Jika $A = \left(a_{ij}\right)$ dan $B = \left(b_{ij}\right)$ kedua-duanya adalah matriks $m \times n$, maka **jumlah** $A + B$ adalah matriks $m \times n$ yang anggota ke-*ij* adalah $a_{ij} + b_{ij}$ untuk setiap pasang (i, j) (Leon, 1998).

Definisi 2.3 (operasi perkalian matriks)

Jika $A=\left(a_{ij}\right)$ adalah matriks $m\times n$ dan $B=\left(b_{ij}\right)$ adalah matriks $n\times r$, maka hasil **kali** $AB=C=c_{ij}$ adalah matriks $m\times r$ yang anggota-anggotanya didefinisikan oleh $c_{ij}=\sum_{k=1}^{n}a_{ik}b_{kj}$ (Leon, 1998).

2.1.2 Vektor

Definisi 2.4 (vektor)

Untuk bilangan bulat positif *n*, R^n adalah himpunan *n-tuples* terurut dari bentuk $\{x_1,x_2,\ldots,x_n\}$, yang dapat dipandang sebagai koordinat titik $\boldsymbol{x}$ di R^n.

Dalam penulisannya, sebuah vektor merupakan sebuah matriks $n\times 1$ yang disebut matriks kolom. Contohnya $\boldsymbol{x}=\begin{pmatrix}x_1\\x_2\\\vdots\\x_n\end{pmatrix}$. Sedangkan matriks berukuran 1×1 merupakan sebuah skalar (Schalkoff, 1997).

Definisi 2.5 (*inner products*)

Jika $\boldsymbol{x}$ dan $\boldsymbol{y}$ merupakan vektor riil $n\times 1$, perkalian dalam dari keduanya (*inner products*) didefinisikan sebagai hasil penjumlahan dari perkalian masing-masing elemen vektor yang bersesuaian, dimana salah satu dari dua vektor tersebut di-*transpose*-kan terlebih dahulu. *Inner products* dinotasikan dengan $\langle \boldsymbol{x},\boldsymbol{y}\rangle=\boldsymbol{x}^T\boldsymbol{y}=\sum_{k=1}^{n}\boldsymbol{x}_k\boldsymbol{y}_k$ $=\boldsymbol{y}^T\boldsymbol{x}=\sum_{k=1}^{n}\boldsymbol{y}_k\boldsymbol{x}_k$ (Schalkoff, 1997).

Definisi 2.6 (*norm*)

Panjang dari sebuah vektor (*norm*) dapat dinyatakan dengan akar dari perkalian dalam vektor itu sendiri, yaitu $\|\boldsymbol{x}\| = \sqrt{\sum_{k=1}^{n} x_k^2} = \langle \boldsymbol{x}, \boldsymbol{x} \rangle^{1/2}$ (Schalkoff, 1997).

Definisi 2.7 (jarak dua vektor)

Di dalam $\mathbb{R}^n$ fungsi jarak $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ didefinisikan sebagai $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ (Schalkoff, 1997).

2.1.3 Optimasi

Dalam kehidupan nyata, banyak sekali hal-hal yang membutuhkan suatu pemikiran dalam menyelesaikan suatu permasalahan sehingga didapatkan hasil yang terbaik/optimal, terutama dalam bidang kerekayasaan, hal tersebut sangat perlu untuk diperhatikan. Secara umum, masalah optimasi tersebut dapat dituliskan sebagai Max/Min $f(X)$, dimana $f(X)$ adalah fungsi obyektif yang akan dioptimalkan (dimaksimalkan atau diminimumkan). Nilai X dapat berupa skalar ataupun berupa vektor. Apabila berupa vektor, maka fungsi tersebut terdiri atas lebih dari satu variabel bebas (Luknanto, 2000). Dalam Skripsi ini, akan dibahas mengenai optimasi dari fungsi multivariabel nonlinier. Untuk lebih memahami permasalahan ini, berikut diberikan beberapa teori sebagai dasar pemahaman.

Definisi 2.8 (fungsi linier)

Suatu fungsi $f(x)$ dikatakan sebagai fungsi linier apabila variabel bebasnya berorde satu (www.wikipedia.org, 2007).

Definisi 2.9 (fungsi multivariabel nonlinier)

Fungsi multivariabel nonlinier merupakan fungsi yang terdiri atas lebih dari satu variabel bebas, dimana variabel bebasnya berorde lebih dari satu (www.wikipedia.org, 2007).

Definisi 2.10 (maksimum dan minimum lokal)

Fungsi f mempunyai maksimum lokal di c jika $f(c) \geq f(x)$ bilamana x dekat c. Ini berarti bahwa $f(c) \geq f(x)$ untuk semua x di dalam suatu selang terbuka yang mengandung c. Secara serupa, f mempunyai minimum lokal di c jika $f(c) \leq f(x)$ bilamana x dekat dengan c (Stewart, 2001).

Definisi 2.11 (maksimum dan minimum global)

Fungsi f mempunyai maksimum global di c jika $f(c) \geq f(x)$ untuk semua x di D, dengan D adalah daerah asal f. Bilangan $f(c)$ disebut nilai maksimum f pada D. Secara serupa, f mempunyai minimum global di c jika $f(c) \leq f(x)$ untuk semua x di D dan bilangan $f(c)$ disebut nilai minimum f pada D. Nilai maksimum dan minimum f disebut nilai ekstrim f (Stewart, 2001).

Definisi 2.12 (kuadrat kesalahan rata-rata (*MSE*))

Nilai kuadrat kesalahan rata-rata dirumuskan sebagai

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - x)^2$$

dimana x_i = nilai pendekatan data ke-i

x = nilai data yang diharapkan

n = banyaknya data

Nilai *MSE* sering dipakai untuk menilai seberapa bagus sebuah usaha pendekatan terhadap suatu hal, terutama dalam suatu penelitian yang berkaitan dengan data. Semakin kecil nilai *MSE* yang diperoleh, semakin akurat hasil dari pendekatan tersebut (www.wikipedia.org, 2007).

Definisi 2.13 (*gradient*)

Misalkan $E(\boldsymbol{w})$ adalah fungsi dari $\boldsymbol{w}$, dimana $\boldsymbol{w} = (w_1, w_2, ..., w_n)$ suatu vektor dalam himpunan terbuka $\mathbb{R}^n$, sedemikian sehingga $\frac{\partial E(\boldsymbol{w})}{\partial w_k}$ ada untuk $k = 1, 2, ..., n$. Gradien dari $E(\boldsymbol{w})$, dinotasikan dengan $\nabla E(\boldsymbol{w})$, adalah vektor kolom

$$\nabla E(\boldsymbol{w}) = \left(\frac{\partial E(\boldsymbol{w})}{\partial w_1} \quad \frac{\partial E(\boldsymbol{w})}{\partial w_2} \quad \cdots \quad \frac{\partial E(\boldsymbol{w})}{\partial w_n} \right)^T \qquad (2.1)$$

(Mathews, 2004).

Definisi 2.14 (metode *descent*)

Metode *descent* didefinisikan sebagai iterasi dari

$$w_{k+1} = w_k + \alpha_k d_k \qquad (2.2)$$

dimana $\alpha_k > 0$ adalah laju pembelajaran/*learning rate* (Boyd dan Vandenberghe, 2004).

Definisi 2.15 (metode *gradient-descent*)

$d_k = -\nabla E(w_k)$ pada metode *descent* adalah arah pencarian dengan gradien bernilai negatif, yang selanjutnya metode *descent* pada definisi (2.14) disebut sebagai metode *gradient-descent* atau *steepest-descent* (Boyd dan Vandenberghe, 2004).

Metode *gradient-descent* ini sering digunakan dalam proses pencarian titik optimum (minimum atau maksimum) suatu fungsi, dimana gradien atau kemiringan dari lintasan fungsinya bernilai 0 (nol) apabila telah tercapai titik minimum atau maksimum tersebut (Bose dan Liang, 1996).

Algoritma s*teepest-descent* adalah sebagai berikut:

Data : $w_0 \in \mathbb{R}^n$

Langkah 1 : tentukan $k = 0$

Langkah 2 : jika $\nabla E(w_k) = 0$ stop,
jika tidak hitung *search direction* $d_k = -\nabla E(w_k)$

Langkah 3 : hitung *step-size* $\alpha_k \in \arg\min_{\alpha \geq 0} E(w_k + \alpha \boldsymbol{d}_k)$

Langkah 4 : hitung $w_{k+1} = w_k + \alpha_k d_k$, $k = k+1$, kembali ke langkah 2

Dalam kasus peminimuman, besarnya laju pembelajaran α_k ditentukan sedemikian sehingga $E(w_{k+1})$ adalah minimum dari $E(\boldsymbol{w})$ (Bose dan Liang, 1996). Untuk mencari nilai α_k digunakan metode *line-search* yang akan meminimumkan nilai fungsi *E*.

Teorema 2.1 dan 2.2 diberikan sebagai landasan dasar tentang kekonvergenan global:

Teorema 2.1 (Bolzano-Weierstrass)

Setiap subhimpunan tak berhingga yang terbatas dalam $\mathbb{R}$ mempunyai titik limit (Lewin, 1993).

Teorema 2.2 (kekonvergenan global)

Misalkan fungsi $E(\boldsymbol{w}):\mathbb{R}^n \to \mathbb{R}$ terdiferensial kontinu pada himpunan $S=\{\boldsymbol{w}\in\mathbb{R}^n \mid E(\boldsymbol{w})\le E(w_0)\}$, S terbatas dan tertutup. Maka setiap titik limit w^* dari himpunan $\{w_k\}$ memenuhi $\nabla E(w^*)=0$ (Freund, 2004).

Bukti:

Akan dibuktikan dengan menggunakan kontradiksi. Karena $\{w_k\}\subset S$ terbatas dan tertutup, menurut Teorema 2.1 terdapat paling sedikit satu titik limit dari $\{w_k\}$. Misalkan w^* titik limit dari $\{w_k\}$. Andaikan w_k konvergen ke w^*, tetapi $\nabla E(w^*)\neq 0$. Karena $d^*=-\nabla E(w^*)\neq 0$ maka terdapat $\alpha>0$ sedemikian hingga $\delta=E(w^*)-E(w^*+\alpha d^*)>0$.

Perhatikan barisan d_k berikut :

$d_1=-\nabla E(w_1=w_0+\alpha_0 d_0)$

$d_2=-\nabla E(w_2=w_1+\alpha_1 d_1)$

$\vdots$

$d_k=-\nabla E(w_k=w_{k-1}+\alpha_{k-1} d_{k-1})$

$\vdots$

Karena $w_k\to w^*$ dan *f differentiable* maka $d_k\to d^*=-\nabla E(w^*)$ dan $w_k+\alpha d_k\to w^*+\alpha d^*$. Untuk *k* cukup besar didapatkan

$$E(w_k+\alpha d_k)\le E(w^*+\alpha d^*)+\frac{\delta}{2}=E(w^*+\alpha d^*)+\delta-\frac{\delta}{2}=E(w^*)-\frac{\delta}{2}$$

. Di lain pihak, $E(w^*)<E(w_k+\alpha_k d_k)\le E(w_k+\alpha d_k)\le E(w^*)-\frac{\delta}{2}$.

Hal ini adalah sebuah kontradiksi, karena $\frac{\delta}{2}$ adalah sebuah bilangan positif, seharusnya $E(w^*)>E(w^*)-\frac{\delta}{2}$. Jadi $\nabla E(w^*)=0$.

2.1.4 Dasar teori lain

Definisi 2.16 (aturan rantai)

Jika f dan g keduanya dapat didiferensialkan, dan $F = f \circ g$ adalah fungsi komposisi yang didefinisikan oleh $F(x) = f(g(x))$, maka F dapat didiferensialkan menjadi F' yang diberikan oleh hasil kali

$$F'(x) = f'(g(x))g'(x).$$

Dalam notasi Leibniz, jika $y = f(u)$ dan $u = g(x)$ keduanya fungsi yang dapat didiferensialkan, maka

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

(Stewart, 2001).

Definisi 2.17 (turunan parsial)

Misalkan fungsi $z = f(x, y)$ didefinisikan dalam domain D dan (x_1, y_1) titik pada D. Kemudian fungsi $f(x, y_1)$ bergantung pada x sendiri dan terdefinisi dalam sebuah interval di sekitar x_1. Sehingga turunannya terhadap x di $x = x_1$ mungkin ada. Jika ada maka nilai tersebut disebut dengan turunan parsial dari $f(x, y)$ terhadap x di (x_1, y_1) dan dinotasikan $\frac{\partial f}{\partial x}(x_1, y_1)$ (Kaplan, 1962).

Definisi 2.18 (titik limit)

Titik x dikatakan sebagai titik limit dari himpunan U jika $\forall \delta > 0$ berlaku $(x - \delta, x + \delta) \cap U \setminus \{x\} \neq \varnothing$ (Lewin, 1993).

2.2 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf pada manusia, yang dapat dilatih untuk mengenali suatu obyek yang memiliki pola tertentu. Dengan pelatihan yang terstruktur, JST yang telah terlatih dapat mengenali obyek tertentu sekalipun sudah ada sedikit perubahan. Kata pelatihan disini digunakan untuk menjelaskan bahwa data diproses sedemikian sehingga sistem dapat mengenali pola data tersebut untuk mendapatkan hasil yang diinginkan.

JST dibentuk sebagai generalisasi model matematika dari jaringan syaraf pada manusia, dengan asumsi bahwa:

1. Pemrosesan informasi terjadi pada banyak elemen sederhana (neuron).
2. Sinyal dikirimkan diantara neuron-neuron melalui penghubung-penghubung.
3. Penghubung antar neuron memiliki bobot yang akan memperkuat atau memperlemah sinyal.
4. Untuk menentukan *output* (keluaran), setiap neuron menggunakan fungsi aktivasi (biasanya bukan fungsi linier) yang dikenakan pada jumlahan *input* (masukan) yang diterima. Besarnya keluaran ini selanjutnya dibandingkan dengan suatu ambang batas.

JST ditentukan oleh tiga hal, yaitu:

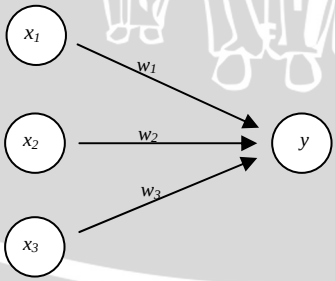
1. Pola hubungan antar neuron (arsitektur jaringan)
2. Metode untuk menentukan bobot penghubung (metode *training*/*learning*/algoritma)
3. Fungsi aktivasi

2.2.1 Struktur dasar Jaringan Syaraf Tiruan

Perhatikan Gambar 2.1, keluaran y menerima masukan dari neuron-neuron x_1, x_2 dan x_3 dengan bobot masing-masing adalah w_1, w_2 dan w_3. Kombinasi linier dari masukan dengan masing-masing bobotnya adalah

$$net = x_1 w_1 + x_2 w_2 + x_3 w_3 \quad . \tag{2.3}$$

Besarnya impuls yang akan diterima oleh y (sebagai nilai keluaran) dihitung berdasarkan fungsi aktivasi $y = f(net)$ yang dipakai (Siang, 2004).



Gambar 2.1 Struktur jaringan sederhana

2.2.2 Fungsi aktivasi, bias dan normalisasi

2.2.2.1 Fungsi aktivasi

Dalam jaringan syaraf tiruan, fungsi aktivasi dipakai untuk menentukan keluaran suatu neuron. Argumen fungsi aktivasi adalah net masukan, yaitu kombinasi linier antara masukan dengan bobotnya. Jika $net = \sum x_i w_i$, dengan x_i = nilai masukan dan w_i = nilai bobotnya, maka fungsi aktivasinya adalah $f(net) = f(\sum x_i w_i)$.

Beberapa fungsi aktivasi yang sering dipakai adalah sebagai berikut :

1. Fungsi *Threshold* (batas ambang)

$$f(x) = \begin{cases} 1 & \text{jika } x \geq a \\ 0 & \text{jika } x < a \end{cases} \quad (2.4)$$

dengan a = nilai batas ambang.

Selain itu juga ada fungsi *threshold* yang keluarannya tidak dibuat 1 atau 0, tetapi bernilai 1 atau -1 (*threshold* bipolar)

$$f(x) = \begin{cases} 1 & \text{jika } x \geq a \\ -1 & \text{jika } x < a \end{cases} \quad (2.5)$$

2. Fungsi Sigmoid

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.6)$$

Fungsi ini sering dipakai karena nilai fungsinya yang terletak antara 0 dan 1, serta dapat diturunkan dengan mudah. Hasil turunannya adalah

$$f'(x) = f(x)(1 - f(x)) \ . \quad (2.7)$$

Sama seperti fungsi *threshold*, fungsi sigmoid juga dapat bernilai antara -1 hingga 1, yang disebut fungsi sigmoid bipolar. Bentuk fungsinya adalah

$$f(x) = \frac{2}{1+e^{-x}} - 1 \ , \quad (2.8)$$

dan hasil turunannya adalah

$$f'(x) = \frac{(1+f(x))(1-f(x))}{2} \ . \quad (2.9)$$

3. Fungsi Identitas / *Purelin*

$$f(x) = x \quad (2.10)$$

Fungsi identitas dipakai apabila nilai keluaran jaringan yang diinginkan berupa sembarang bilangan riil, bukan hanya pada range [0,1] atau [-1,1] (Siang,2004).

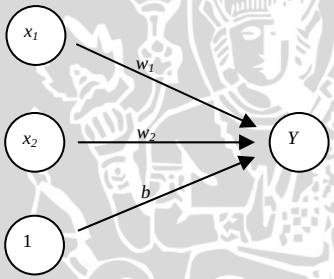
Dalam Skripsi ini fungsi aktivasi yang dipakai pada lapisan tersembunyi adalah fungsi sigmoid biner. Sedangkan pada lapisan keluaran dipakai fungsi identitas.

2.2.2.2 Bias

Dalam suatu jaringan seringkali ditambahkan sebuah unit masukan yang nilainya selalu sama dengan 1, yang disebut sebagai bias (Gambar 2.2). Sehingga, keluaran unit penjumlahnya adalah

$$net = b + \sum_i x_i w_i \qquad (2.11)$$

dengan *b* adalah bobot bias.



Gambar 2.2 Bias

2.2.2.3 Normalisasi

Normalisasi dilakukan pada data masukan dan target masukan. Tujuannya adalah untuk mempermudah perhitungan apabila nilai data yang diolah terlalu besar. Rumus normalisasi adalah :

$$x^* = \frac{x - \text{data terkecil}}{\text{data terbesar} - \text{data terkecil}}$$

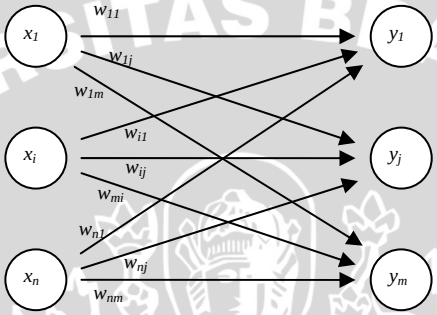
dengan x^* = data ternormalisasi dan x = data asli (Purnomo dan Kurniawan, 2006).

2.2.3 Arsitektur jaringan

Beberapa arsitektur jaringan yang sering dipakai dalam jaringan syaraf tiruan antara lain:

1. Jaringan Lapis Tunggal (*Single Layer Network*)

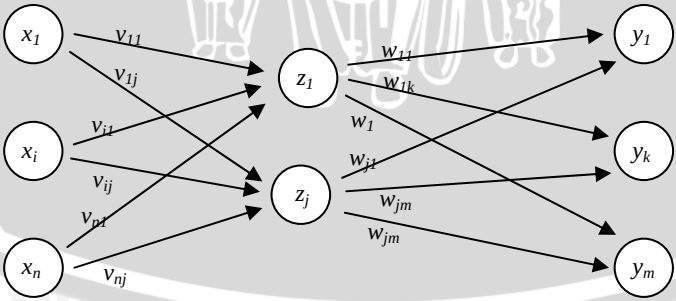
Dalam jaringan ini, sekumpulan masukan dihubungkan langsung dengan sekumpulan keluarannya (Gambar 2.3). Model arsitektur semacam ini tepat digunakan untuk pengenalan pola karena kesederhanaannya.



Gambar 2.3 Arsitektur jaringan lapis tunggal

2. Jaringan Lapis Jamak (*Multi Layer Network*)

Jaringan ini merupakan perluasan dari jaringan lapis tunggal. Diantara lapisan masukan dan keluaran ditambahkan lapisan yang lain yang disebut lapisan tersembunyi (*hidden layer*) seperti pada Gambar 2.4. Jumlah lapisan tersembunyi bisa lebih dari satu. Jaringan lapis jamak dapat menyelesaikan masalah yang lebih kompleks dibandingkan dengan lapisan tunggal, meskipun kadangkala proses pelatihannya lebih kompleks dan lama (Siang, 2004).



Gambar 2.4 Arsitektur jaringan lapis jamak

2.2.4 Proses pelatihan/pembelajaran jaringan

Proses pelatihan/pembelajaran di sini artinya data-data masukan diolah sedemikian sehingga diperoleh nilai keluaran yang diinginkan. Setelah menentukan arsitektur jaringan, proses pelatihan dapat dijalankan. Dalam jaringan syaraf tiruan, ada dua macam pelatihan dalam cara memodifikasi bobotnya, yaitu pelatihan dengan supervisi (*supervised learning*) dan tanpa supervisi (*unsupervised learning*).

1. Pelatihan dengan supervisi

Dalam pelatihan ini terdapat sejumlah pasangan data, yaitu data masukan dan target keluaran yang dipakai untuk melatih jaringan hingga diperoleh bobot yang diinginkan. Pasangan data tersebut akan memberikan informasi yang jelas tentang bagaimana sistem harus mengubah dirinya untuk memperoleh bentuk yang terbaik dalam kerjanya. Dalam setiap pelatihannya, masukan yang diberikan akan diproses untuk menghasilkan suatu keluaran. Selisih antara keluaran jaringan dengan target keluaran yang diinginkan merupakan kesalahan yang terjadi. Jaringan akan memodifikasi bobot sesuai dengan kesalahan tersebut.

2. Pelatihan tanpa supervisi

Dalam pelatihan tanpa supervisi, tidak ada pasangan data yang dijadikan sebagai “pelatih” yang mengarahkan proses pelatihan. Perubahan bobot jaringan dilakukan berdasarkan parameter tertentu dan jaringan dimodifikasi menurut ukuran parameter tersebut (Siang, 2004).

Dalam Skripsi ini pelatihan yang digunakan adalah pelatihan dengan supervisi, yang salah satu modelnya adalah *backpropagation*, yang akan dibahas pada subbab selanjutnya. Model ini akan mencari nilai minimum dari kesalahan yang terjadi pada proses pelatihan data (yaitu selisih antara nilai target yang diharapkan dengan nilai keluaran aktual dalam jaringan) dengan cara mengkoreksi nilai kesalahan tersebut, yang selanjutnya akan dipropagasikan kembali ke dalam lapisan sebelumnya dalam jaringan. Proses propagasi balik tersebut nantinya akan merubah nilai bobot yang digunakan, sampai akhirnya diperoleh nilai kesalahan yang paling minimum dengan menggunakan bobot yang baru tersebut melalui proses iterasi. Ada beberapa metode numerik yang dapat digunakan dalam pencarian bobot baru tersebut, salah satu diantaranya adalah metode *gradient-descent*. Metode inilah yang dipakai pada penulisan Skripsi ini, dan

dibandingkan dengan metode *gradient-descent* yang dimodifikasi dengan menambahkan momentum pada proses perubahan bobotnya.

Parameter yang dipakai untuk mendapatkan bobot baru adalah *Sum Square Error* (SSE) yang didefinisikan sebagai :

$$E_i = \frac{1}{2}\sum_{k=1}^{m}\left(t_{ik} - y_{ik}\right)^2 \qquad (2.12)$$

Fungsi tersebut dipilih karena dapat diturunkan dan dapat diolah untuk meminimalkan selisih antara nilai target dan keluaran (kesalahan). Nilai pengali 0,5 di atas dipakai untuk mempermudah perhitungan (Patterson, 1995). Sedangkan parameter yang akan dijadikan sebagai kriteria penghentian iterasi adalah *Mean Square Error* (MSE) yang didefinisikan sebagai

$$MSE_r = \frac{\sum_{i=1}^{s} E_i}{s} \qquad (2.13)$$

dimana MSE_r = kuadrat rata-rata kesalahan pada iterasi ke-*r*

E_i = parameter kesalahan pada data ke-*i*

t_{ik} = target ke-*k* pada data ke-*i*

y_{ik} = keluaran aktual ke-*k* pada data ke-*i*

s = jumlah data

m = jumlah unit keluaran

2.3 *Backpropagation*

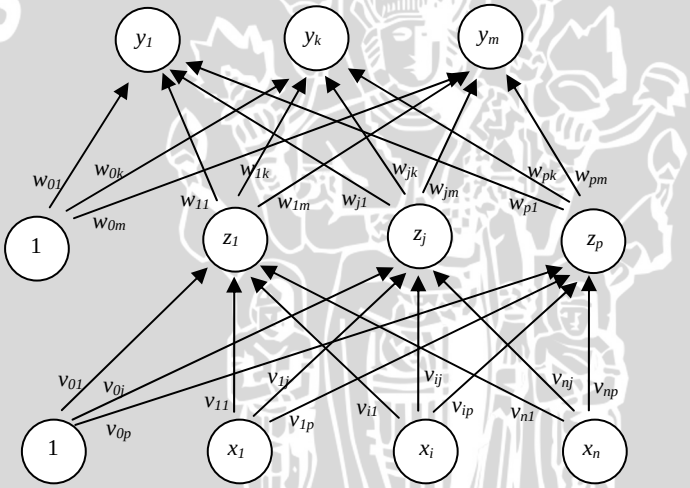
Keterbatasan yang dimiliki oleh jaringan lapis tunggal, yaitu hanya dapat mengenali pola yang sederhana, membuat perkembangan jaringan syaraf tiruan menjadi terhenti pada sekitar tahun 1970-an. Dengan ditemukannya *backpropagation* yang memiliki lapisan jamak dan telah berhasil diaplikasikan pada berbagai bidang, membuat jaringan syaraf tiruan kembali diminati. Kelemahan utama dalam jaringan lapis tunggal adalah sedikitnya jumlah lapisan yang dipakai, sehingga hanya dapat mengenali pola yang sederhana. Kelemahan ini dapat ditanggulangi dengan menambahkan satu atau beberapa lapisan, yang disebut lapisan tersembunyi (*hidden layer*) diantara lapisan masukan dan lapisan keluaran.

Backpropagation melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan dalam 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.

Arsitektur *backpropagation* ditunjukkan dalam Gambar 2.5. Arsitektur *backpropagation* tersebut memiliki *n* unit masukan ditambah dengan sebuah bias, sebuah lapisan tersembunyi terdiri dari *p* unit ditambah dengan sebuah bias, serta *m* unit keluaran.

v_{ij} merupakan bobot garis dari unit masukan x_i ke unit lapisan tersembunyi z_j. v_{0j} merupakan bobot garis dari bias di unit masukan ke unit lapisan tersembunyi z_j. w_{jk} merupakan bobot dari unit lapisan tersembunyi z_j ke unit keluaran y_k. w_{0k} merupakan bobot dari bias di lapisan tersembunyi ke unit keluaran y_k.



Gambar 2.5 Arsitektur *backpropagation* dengan satu lapisan tersembunyi

Pada *backpropagation* ada tiga tahap dalam proses pelatihannya. Tahap yang pertama adalah propagasi maju, kedua adalah propagasi mundur, dan yang ketiga adalah tahap perubahan bobot. Berikut akan dijelaskan ketiga tahap tersebut dalam satu kali proses pelatihan untuk satu data:

1. Propagasi Maju (*feed forward*)

Dalam tahapan ini, terdapat pasangan data masukan dan target keluaran yang akan dilatih. Misalkan diberikan vektor masukan $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ dan vektor bobot $\boldsymbol{v} = (v_{11}, v_{12}, \ldots, v_{ij})$ yang berada di antara lapisan masukan dan lapisan tersembunyi. Kombinasi linier dari $\boldsymbol{x}$ dan $\boldsymbol{v}$, yaitu

$$z\_net_j = \sum_{i=1}^{n} x_i v_{ij} \tag{2.14}$$

akan diaktifkan dengan fungsi aktivasi yang telah ditentukan, dituliskan dengan

$$z_j = f(z\_net_j) \tag{2.15}$$

dimana z_net_j = nilai masukan neuron ke-*j* pada lapisan tersembunyi

z_j = nilai keluaran neuron ke-*j* pada lapisan tersembunyi

Selanjutnya nilai dari z_j dikombinasikan lagi dengan bobot yang berada di antara lapisan tersembunyi dengan lapisan keluaran, yaitu vektor bobot $\boldsymbol{w} = (w_1, w_2, \ldots, w_p)$ menghasilkan

$$y\_net_k = \sum_{j=1}^{p} z_j w_{jk} \,. \tag{2.16}$$

Nilai tersebut diaktifkan oleh fungsi aktivasi, dituliskan dengan

$$y_k = f(y\_net_k) \tag{2.17}$$

dimana y_net_k = nilai masukan neuron ke-*k* pada lapisan keluaran

y_k = nilai keluaran neuron ke-*k* pada lapisan keluaran

Apabila ditambahkan suatu nilai bias dalam kedua lapisan, yaitu pada lapisan masukan dan lapisan tersembunyi, maka nilai kombinasi linier pada persamaan (2.14) menjadi

$$z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \tag{2.18}$$

dan persamaan (2.16) menjadi

$$y\_net_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \,. \tag{2.19}$$

2. Propagasi mundur (*backpropagation*)

Pada tahap ini proses awalnya adalah membandingkan keluaran JST dengan target keluaran. Galat yang diperoleh digunakan untuk memperbaiki bobot sehingga nilai kesalahan yang terjadi dapat diminimalkan. Parameter kesalahan yang dipakai adalah *Sum Square Error*, seperti yang telah dituliskan dalam persamaan (2.12). Dalam hal ini, diasumsikan proses perhitungan hanya dilakukan pada satu pasangan data, yaitu satu data masukan dan data target. Sehingga persamaan (2.12) dapat ditulis kembali menjadi

$$E = \frac{1}{2}\sum_{k=1}^{m}\left(t_k - y_k\right)^2 \tag{2.20}$$

yang dapat diuraikan menjadi

$$\begin{aligned} E &= \frac{1}{2}\sum_{k=1}^{m}(t_k - y_k)^2 \\ &= \frac{1}{2}\sum_{k=1}^{m}(t_k - f(y\_net_k))^2 \\ &= \frac{1}{2}\sum_{k=1}^{m}(t_k - f(\sum_{j=1}^{p} z_j w_{jk}))^2 \end{aligned} \tag{2.21}$$

dimana E = nilai parameter kesalahan

t_k = nilai target keluaran neuron ke-*k* pada lapisan keluaran

y_k = nilai keluaran aktual neuron ke-*k* pada lapisan keluaran

z_j = nilai keluaran neuron ke-*j* pada lapisan tersembunyi

w_{jk} = bobot antara neuron ke-*j* pada lapisan tersembunyi dan neuron ke-*k* pada lapisan keluaran

Untuk memperbaiki nilai bobot, akan diminimalkan nilai galat/kesalahan dari persamaan (2.21). Untuk itu akan dicari turunan parsial dari fungsi kesalahan tersebut dengan aturan rantai sehingga didapatkan suatu nilai δ yang merupakan unit kesalahan yang akan dipakai dalam perubahan bobot pada lapisan sebelumnya. Dari sini diperoleh nilai

$$\Delta w_{jk} = \alpha \delta_k z_j \tag{2.22}$$

yang merupakan suku perubahan bobot w_{jk} dengan α = laju pembelajaran (*learning rate*) dan δ_k = suku perubahan bobot pada neuron ke-*k* dari lapisan keluaran. Nilai α merupakan bilangan positif yang lebih besar dari nol, dapat dicari dengan metode *line-search* yang akan dibahas pada bab pembahasan. Selanjutnya nilai dari δ_k dipakai sebagai nilai keluaran dari masing-masing neuron ke-*k* pada lapisan keluaran yang akan dikombinasikan dengan bobot-bobot sebelumnya (w_{jk}) untuk memperoleh nilai masukan bagi neuron-neuron pada lapisan sebelumnya (lapisan tersembunyi). Nilai masukan tersebut diberi simbol γ_net pada masing-masing neuron di lapisan tersembunyi, dan γ setelah diaktivasikan. Selanjutnya, seperti pada proses sebelumnya, nilai γ ini akan dikombinasikan dengan bobot-bobot sebelumnya pada masing-masing lapisan tersembunyi. Karena dalam Skripsi ini hanya digunakan satu unit lapisan tersembunyi, maka hanya akan dicari nilai γ pada satu lapisan tersembunyi saja, yaitu γ_j dengan *j* menunjukkan nilai γ pada neuron ke-*j*. Dari sini didapat suku perubahan bobot v_{ij}, yaitu

$$\Delta v_{ij} = \alpha \gamma_j x_i . \qquad (2.23)$$

Suku perubahan bobot yang telah ditulis dalam persamaan (2.22) dan (2.23) dicari penurunan rumusnya dalam bab pembahasan.

3. Tahap perubahan bobot

Tahap berikutnya adalah menggantikan bobot yang lama dengan bobot yang baru didapat dari perhitungan di atas pada masing-masing lapisan. Nilai bobot yang baru adalah

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk} \qquad (2.24)$$

di antara lapisan tersembunyi dan lapisan keluaran, dan

$$v_{ij}(t+1) = v_{ij}(t) + \Delta v_{ij} \qquad (2.25)$$

di antara lapisan masukan dan lapisan tersembunyi, dengan *t* menunjukkan waktu (proses iterasi) yang ke-*t*.
Bobot baru inilah yang akan digunakan pada perhitungan data selanjutnya.

Untuk memudahkan pemahaman tentang perhitungan di atas, berikut dituliskan algoritma *backpropagation* (Kusumadewi, 2004):

1. Inisialisasi bobot dengan bilangan acak yang cukup kecil.

2. Tetapkan : iterasi maksimum, *MSE* maksimum dan laju percepatan/*learning rate* (α).
3. Inisialisasi : iterasi = 0, *MSE* = 1
4. Kerjakan langkah-langkah berikut selama iterasi < iterasi maksimum atau *MSE* > *MSE* maksimum.

a. iterasi = iterasi + 1

b. Untuk tiap-tiap pasangan elemen yang akan dilatih kerjakan:

Feedforward:

Tiap-tiap unit masukan (x_i, $i = 1, 2, ..., n$) menerima sinyal x_i dan meneruskannya ke unit lapisan tersembunyi berikutnya.

i. Hitung jumlah sinyal-sinyal terbobot masukan di unit tersembunyi (z_j, $j = 1, 2, \ldots, p$):

$$z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \tag{2.26}$$

dan gunakan fungsi aktivasi untuk menghitung sinyal keluarannya :

$$z_j = f(z\_net_j) \tag{2.27}$$

ii. Hitung jumlah sinyal-sinyal terbobot masukan di unit keluaran ($y_k, k = 1, 2, \ldots, m$) :

$$y\_net_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \tag{2.28}$$

dan gunakan fungsi aktivasi untuk menghitung sinyal keluarannya :

$$y_k = f(y\_net_k) \tag{2.29}$$

Backpropagation:

i. Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran (y_k, $k = 1, 2, ..., m$):

$$\delta_k = (t_k - y_k) f'(y\_net_k) \tag{2.30}$$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar di bawahnya. Kemudian hitung suku perubahan bobot w_{jk} (yang akan dipakai untuk merubah bobot w_{jk}) dengan laju percepatan (*learning rate*) α :

$$\Delta w_{jk} = \alpha \delta_k z_j \tag{2.31}$$

dengan $k = 1, 2, ..., m$ dan $j = 1, ..., p$.

Sedangkan suku perubahan bobot bias w_{0k} :

$$\Delta w_{0k} = \alpha \delta_k \tag{2.32}$$

ii. Tiap-tiap unit tersembunyi (z_j, $j = 1, 2, ..., p$) menjumlahkan delta masukannya dari setiap unit yang berada di lapisan berikutnya:

$$\gamma\_net_j = \sum_{k=1}^{m} \delta_k w_{jk} \tag{2.33}$$

Untuk menghitung informasi kesalahannya, kalikan dengan turunan fungsi aktivasinya:

$$\gamma_j = \gamma\_net_j f'(z\_net_j) \tag{2.34}$$

Hitung suku perubahan bobot v_{ij} yang akan dipakai untuk merubah bobot v_{ij} :

$$\Delta v_{ij} = \alpha \gamma_j x_i \tag{2.35}$$

dengan $j = 1, 2, ..., p$ dan $i = 1, ..., n$.

Sedangkan suku perubahan bobot bias v_{0j} :

$$\Delta v_{0j} = \alpha \gamma_j \tag{2.36}$$

Perubahan Bobot:

Perubahan bobot garis yang menuju ke unit keluaran:

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}(t) \tag{2.37}$$

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

Perubahan bobot garis yang menuju ke unit tersembunyi:

$$v_{ij}(t+1) = v_{ij}(t) + \Delta v_{ij}(t) \tag{2.38}$$

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

c. Hitung MSE (*Mean Square Error*)

2.3.1 Penambahan momentum pada model *backpropagation*

Pada kenyataannya, metode *gradient-descent* secara umum hanya akan mencapai nilai maksimum atau minimum lokal saja. Untuk lebih mengoptimalkan hasil perhitungan, pada proses pencarian bobot baru dalam model *backpropagation* pada penulisan ini, akan ditambahkan momentum pada perhitungan perubahan bobot pada tahap *backpropagation*. Modifikasi ini didasarkan pada arah

gradien pola terakhir dan pola sebelumnya, yang disebut dengan momentum. Jadi perubahan bobot tidak hanya berdasarkan pola terakhir saja yang diperhitungkan. Selain digunakan untuk menghindari terjadinya optimum lokal pada tahap pelatihannya, modifikasi ini juga dapat mempercepat laju kekonvergenan nilai bobot yang diperoleh, sehingga nilai kesalahan yang terjadi akan dapat diminimumkan dengan cepat. Bobot baru pada proses ke-(t+1) diperoleh berdasarkan bobot pada proses ke-t dan ke-(t-1). Disini perlu ditambahkan parameter baru (μ) yang mencatat besarnya momentum (Siang, 2004).

Persamaan (2.2) yang merupakan persamaan metode *gradient-descent* apabila ditambahkan dengan momentum menjadi

$$w_{k+1} = w_k - \alpha_k \nabla E(w_k) + \mu\left(w_k - w_{k-1}\right) \tag{2.39}$$

atau dapat ditulis

$$\Delta w_k = -\alpha_k \nabla E(w_k) + \mu\left(\Delta w_{k-1}\right) \tag{2.40}$$

Maka dalam model *backpropagation*, bobot baru antara lapisan tersembunyi dan lapisan keluaran dihitung berdasarkan persamaan berikut :

$$w_{jk}(t+1) = w_{jk}(t) + \alpha\delta_k z_j + \mu(w_{jk}(t) - w_{jk}(t-1)) \tag{2.41}$$

atau

$$\Delta w_{jk}(t) = \alpha\delta_k z_j + \mu(\Delta w_{jk}(t-1)) \tag{2.42}$$

dan untuk perubahan bobot antara lapisan masukan dan lapisan tersembunyi

$$v_{ij}(t+1) = v_{ij}(t) + \alpha\gamma_j x_i + \mu(v_{ij}(t) - v_{ij}(t-1)) \tag{2.43}$$

atau

$$\Delta v_{ij}(t) = \alpha\gamma_j x_i + \mu(\Delta v_{ij}(t-1)) \tag{2.44}$$

2.4 Pasar Modal

Pasar modal didefinisikan sebagai pasar untuk berbagai instrumen keuangan (sekuritas) jangka panjang yang bisa diperjualbelikan, baik dalam bentuk hutang ataupun modal sendiri, yang dapat diterbitkan baik oleh pemerintah, *public autorities*, maupun swasta.

Pasar modal memiliki peran besar bagi perekonomian suatu negara, karena pasar modal menjalankan dua fungsi sekaligus, yaitu sebagai fungsi ekonomi dan fungsi keuangan (Milagefira, 2006). Salah satu kegiatan pasar modal adalah transaksi saham. Pada dasarnya ada dua

keuntungan yang diperoleh pemodal dengan membeli atau memiliki saham, yaitu *dividen* (pembagian keuntungan yang diberikan perusahaan penerbit saham) dan *capital gain* (selisih antara harga beli dan harga jual). Sedangkan resikonya adalah tidak mendapat *dividen* dan *capital loss*, jika harga jual lebih rendah daripada harga beli.

Ada dua model analisis yang digunakan oleh pemain bursa untuk memperkirakan harga saham, yaitu:

1. Analisis fundamental

   Model ini memperkirakan harga saham di masa mendatang dengan mengestimasi nilai-nilai faktor fundamental, seperti penjualan, pertumbuhan penjualan, biaya, kebijakan *dividen* dan sebagainya yang mempengaruhi harga saham di masa mendatang, dan menerapkan hubungan variabel-variabel tersebut sehingga diperoleh taksiran harga saham.
2. Pendekatan teknikal

   Model ini merupakan upaya untuk memperkirakan harga saham denga mengamati perubahan harga di waktu yang lalu. Karenanya perubahan harga saham diperkirakan akan mempunyai pola tertentu yang berulang (Milagefira, 2006).

Menurut Yao,et al (1999) dan Bodis (2004) dalam Milagefira (2006), terdapat dua strategi dalam perdagangan berdasarkan peramalan harga saham, yaitu:

a. Strategi 1 : Jika $(x^*_{t+1} - x^*_t) > 0$ maka saham dibeli, tetapi jika $(x^*_{t+1} - x^*_t) < 0$ maka saham dijual.
b. Strategi 2 : Jika $(x^*_{t+1} - x_t) > 0$ maka saham dibeli, tetapi jika $(x^*_{t+1} - x_t) < 0$ maka saham dijual.

dimana : x_t = harga penutupan saham hari ini

x^*_t = prediksi harga penutupan saham hari ini

x^*_{t+1} = prediksi harga penutupan saham pada keesokan hari

BAB III
HASIL DAN PEMBAHASAN

Dalam bab ini dibahas mengenai implementasi metode *gradient-descent* dan pencarian nilai laju pembelajaran yang mengoptimalkan fungsi kesalahan. Selain itu juga akan dianalisa secara numerik mengenai perbandingan antara model *backpropagation* standar dengan yang telah dimodifikasi dengan penambahan momentum pada data harga perdagangan saham harian Astra International, Tbk.

3.1 Implementasi Metode *Gradient-Descent* Pada Model *Backpropagation*

Seperti yang telah dibahas pada subbab 2.3 mengenai *backpropagation*, ada tiga tahap dalam proses pelatihannya. Dalam pembahasan ini lebih dititik-beratkan pada penurunan rumus-rumus dalam tahap propagasi mundur (*backpropagation*). Pada tahap ini, fungsi yang dioptimalkan adalah *Sum Square Error* yang telah diuraikan pada persamaan (2.21), yaitu

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - f(\sum_{j=1}^{p} z_j w_{jk}))^2 \qquad (3.1)$$

Dari persamaan (3.1), akan diturunkan secara parsial dengan aturan rantai untuk memperoleh nilai suku perubahan bobot Δv_{ij} dan Δw_{jk}, yang mana tujuannya adalah untuk mencari nilai bobot $\boldsymbol{v}$ dan $\boldsymbol{w}$ yang meminimumkan fungsi kesalahan pada persamaan (3.1). Proses pencarian nilai Δv_{ij} dan Δw_{jk} menggunakan metode *gradient-descent* (*steepest-descent*), yang telah dituliskan pada persamaan (2.2), yaitu $w_{k+1} = w_k - \alpha_k \nabla E(w_k)$. Persamaan (2.2) dapat ditulis ulang menjadi

$$w_{jk}(t+1) = w_{jk}(t) - \alpha_k \nabla E(w_{jk}(t)) \qquad (3.2)$$

dimana : $\nabla E\big(w(t)\big)$ = gradien dari fungsi E terhadap bobot w_{jk}

w_{jk} = bobot antara neuron ke-j pada lapisan tersembunyi dan neuron ke-k pada lapisan keluaran

t = urutan proses perhitungan pasangan data

Kedua ruas pada persamaan (3.2) dikurangkan dengan $w_{jk}(t)$ diperoleh

$$w_{jk}(t+1) - w_{jk}(t) = -\alpha \nabla E(w(t)) \tag{3.3}$$

Nilai $w_{jk}(t+1) - w_{jk}(t)$ merupakan selisih nilai bobot pada pasangan data ke-(t+1) dan ke-t, yaitu $\Delta w_{jk}(t)$, sehingga dapat ditulis kembali menjadi

$$\Delta w_{jk}(t) = -\alpha \nabla E(w_{jk}(t)) \tag{3.4}$$

Perhitungan tersebut dalam tahap *backpropagation* didapat melalui penurunan parsial fungsi kesalahan E terhadap bobot w_{jk} dengan aturan rantai, yaitu

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial y\_net_k} \frac{\partial y\_net_k}{\partial w_{jk}} \tag{3.5}$$

$$= \frac{\partial}{\partial y_k}\left(\frac{1}{2}\sum_{k=1}^{m}\left(t_k - y_k\right)^2\right) \frac{\partial}{\partial y\_net_k}\left(f\left(y\_net_k\right)\right)$$

$$\frac{\partial}{\partial w_{jk}}\left(\sum_{j=1}^{p} z_j w_{jk}\right)$$

$$= -\left(t_k - y_k\right) f'(y\_net_k) z_j \tag{3.6}$$

Apabila didefinisikan nilai $\delta_k = (t_k - y_k) f'(y\_net_k)$, maka persamaan (3.6) menjadi

$$\frac{\partial E}{\partial w_{jk}} = -\delta_k z_j \tag{3.7}$$

Untuk memperoleh nilai kesalahan yang minimum, maka arah pencarian dari gradien pada persamaan (3.7) adalah negatif, sehingga perubahan bobot ***w*** adalah

$$\Delta w_{jk} = \alpha\left(-\frac{\partial E}{\partial w_{jk}}\right) = \alpha \delta_k z_j \tag{3.8}$$

dengan α = laju pembelajaran (*learning rate*), $j = 1, 2, \ldots, p$ dan $k = 1, 2, \ldots, m$. Sedangkan perubahan bobot pada bias, karena nilai inputnya sama dengan satu ($z_0 = 1$), maka persamaan perubahan bobotnya adalah

$$\Delta w_{0k} = \alpha\left(-\frac{\partial E}{\partial w_{0k}}\right) = \alpha \delta_k \tag{3.9}$$

Untuk memperoleh suku perubahan bobot ***v*** dilakukan hal yang sama seperti pada perolehan suku perubahan bobot ***w***, tetapi variabel penurunan terhadap fungsi kesalahannya adalah bobot v_{ij}. Penguraian fungsi kesalahan E pada persamaan (3.1) adalah sebagai berikut

$$E = \frac{1}{2}\sum_{k=1}^{m}\left(t_k - f\left(\sum_{j=1}^{p} z_j w_{jk}\right)\right)^2$$

$$= \frac{1}{2}\sum_{k=1}^{m}\left(t_k - f\left(\sum_{j=1}^{p} f\left(z\_net_j\right) w_{jk}\right)\right)^2$$

$$= \frac{1}{2}\sum_{k=1}^{m}\left(t_k - f\left(\sum_{j=1}^{p} f\left(\sum_{n=1}^{n} x_i v_{ij}\right) w_{jk}\right)\right)^2 \qquad (3.10)$$

Karena nilai target keluaran pada lapisan tersembunyi tidak diketahui (tidak ada nilai koreksi untuk perbandingan), maka dengan mempertimbangkan pengaruh dari neuron-neuron dalam lapisan tersembunyi terhadap nilai masukan bagi neuron-neuron pada lapisan keluaran dan dengan menerapkan aturan rantai, penurunan fungsi kesalahan E terhadap nilai z_j dijabarkan seperti berikut :

$$\frac{\partial E}{\partial z_j} = \sum_{k=1}^{m}\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial y\_net_k}\frac{\partial y\_net_k}{\partial z_j}$$

$$= \sum_{k=1}^{m}\frac{\partial}{\partial y_k}\left(\frac{1}{2}\sum_{k=1}^{m}\left(t_k - y_k\right)^2\right)\frac{\partial}{\partial y\_net_k}\left(f\left(y\_net_k\right)\right)$$

$$\frac{\partial}{\partial z_j}\left(\sum_{j=1}^{p} z_j w_{jk}\right)$$

$$= -\sum_{k=1}^{m}\left(t_k - y_k\right) f'\left(y\_net_k\right) w_{jk} \qquad (3.11)$$

Selanjutnya, turunan parsial fungsi kesalahan E terhadap bobot v_{ij} adalah sebagai berikut :

$$\frac{\partial E}{\partial v_{ij}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial y\_net_k}\frac{\partial y\_net_k}{\partial z_j}\frac{\partial z_j}{\partial z\_net_j}\frac{\partial z\_net_j}{\partial v_{ij}}$$

$$= \frac{\partial}{\partial y_k}\left(\frac{1}{2}\sum_{k=1}^{m}\left(t_k - y_k\right)^2\right)\frac{\partial}{\partial y\_net_k}\left(f\left(y\_net_k\right)\right)$$

$$\frac{\partial}{\partial z_j}\left(\sum_{j=1}^{p} z_j w_{jk}\right)\frac{\partial}{\partial z\_net_j}\left(f\left(z\_net_j\right)\right)\frac{\partial}{\partial v_{ij}}\left(\sum_{i=1}^{n} x_i v_{ij}\right)$$

$$= -\sum_{k=1}^{m}\left(t_k - y_k\right) f'(y\_net_k) w_{jk} f'(z\_net_j) x_i$$

$$= -\sum_{k=1}^{m} \delta_k w_{jk} f'(z\_net_j) x_i \qquad (3.12)$$

Apabila didefinisikan $\gamma_j = \sum_{k=1}^{m}\left(\delta_k w_{jk}\right) f'(z\_net_j)$, maka persamaan (3.12) menjadi

$$\frac{\partial E}{\partial v_{ij}} = -\gamma_j x_i \qquad (3.13)$$

Seperti pada perubahan bobot ***w***, arah gradien menurun, sehingga nilai dari gradien fungsi kesalahan di atas negatif. Maka perubahan bobot v_{ij} adalah

$$\Delta v_{ij} = \alpha\left(-\frac{\partial E}{\partial v_{ij}}\right) = \alpha\gamma_j x_i \qquad (3.14)$$

dan suku perubahan bobot biasnya adalah

$$\Delta v_{0j} = \alpha\left(-\frac{\partial E}{\partial v_{0j}}\right) = \alpha\gamma_j \qquad (3.15)$$

Selanjutnya hasil dari penurunan di atas, yaitu persamaan (3.8), persamaan (3.9), persamaan (3.14) dan persamaan (3.15) dijumlahkan dengan bobot sebelumnya, sehingga diperoleh bobot baru yang digunakan untuk proses perhitungan pada data berikutnya.

3.2 Mencari Nilai Laju Pembelajaran / *Learning Rate* (α) dengan Metode *Line-search*

Nilai laju pembelajaran α dapat dicari dengan menggunakan metode *line-search*. Fungsi aktivasi yang digunakan pada lapisan keluaran adalah fungsi identitas/*purelin*, dimana nilai keluaran dari lapisan tersembunyi dijadikan sebagai nilai masukan untuk lapisan keluaran. Karena fungsi aktivasi pada lapisan keluaran adalah fungsi

identitas, maka $y_k = w_{0k} + \sum_{k=1}^{m} z_j w_{jk}$, sehingga bentuk fungsi kesalahan *E* pada persamaan (2.20) menjadi

$$E = \frac{1}{2}\sum_{k=1}^{\mathrm{m}}(t_k - (w_{0k} + \sum_{j=1}^{p} z_j w_{jk}))^2 \qquad (3.16)$$

Sesuai dengan aturan pada langkah ke-3 dalam algoritma *steepest-descent* pada sub bab 2.1.3, akan dicari nilai ukuran langkah (*step-size*) α yang meminimumkan fungsi objektif $E(\boldsymbol{w} - \alpha\nabla_{w_j} E(\boldsymbol{w}))$ dengan metode *line-search*, sehingga fungsi objektif pada persamaan (3.16) diubah menjadi

$$E^* = \frac{1}{2}\sum_{k=1}^{m}(t_k - [(w_{0k} - \alpha\nabla_{w_{0k}} E) + \sum_{j=1}^{p} z_j (w_{jk} - \alpha\nabla_{w_{jk}} E)])^2 \qquad (3.17)$$

Selanjutnya, parameter α dijadikan sebagai variabel dalam fungsi yang baru tersebut.

$$\begin{aligned} E^* &= \frac{1}{2}\sum_{k=1}^{m}(t_k - [(w_{0k} - \alpha\nabla_{w_{0k}} E) + \sum_{j=1}^{p} z_j (w_{jk} - \alpha\nabla_{w_{jk}} E)])^2 \\ &= \frac{1}{2}\sum_{k=1}^{m}(t_k - w_{0k} + \alpha\nabla_{w_{0k}} E - \sum_{j=1}^{p} z_j w_{jk} + \alpha\sum_{j=1}^{p} z_j \nabla_{w_{jk}} E)^2 \\ &= \frac{1}{2}\sum_{k=1}^{m}(t_k - w_{0k} - \sum_{j=1}^{p} z_j w_{jk} + \alpha(\nabla_{w_{0k}} E + \sum_{j=1}^{p} z_j \nabla_{w_{jk}} E))^2 \qquad (3.18) \end{aligned}$$

Untuk mencari nilai α yang meminimumkan persamaan (3.18), dapat dilakukan dengan membuat turunan dari persamaan (3.18) sama dengan nol, yaitu

$$\frac{dE^*}{d\alpha} = 0$$

$$\frac{d}{d\alpha}\left(\frac{1}{2}\sum_{k=1}^{m}(t_k - w_{0k} - \sum_{j=1}^{p} z_j w_{jk} + \alpha(\nabla_{w_{0k}} E + \sum_{j=1}^{p} z_j \nabla_{w_{jk}} E))^2\right) = 0$$

$$\frac{1}{2}\sum_{k=1}^{m}\frac{d}{d\alpha}\left((t_k - w_{0k} - \sum_{j=1}^{p} z_j w_{jk} + \alpha(\nabla_{w_{0k}} E + \sum_{j=1}^{p} z_j \nabla_{w_{jk}} E))^2\right) = 0$$

$$\sum_{k=1}^{m}(t_k - w_{0k} - \sum_{j=1}^{p} z_j w_{jk} + \alpha(\nabla_{w_{0k}} E + \sum_{j=1}^{p} z_j \nabla_{w_{jk}} E))(\nabla_{w_{0k}} E$$

$$+\sum_{j=1}^{p} z_j \nabla_{w_{jk}} E) = 0$$

$$\alpha = \frac{w_{0k} + \sum_{j=1}^{p} z_j w_{jk} - t_k}{\nabla_{w_{0k}} E + \sum_{j=1}^{p} z_j \nabla_{w_{jk}} E} \qquad (3.19)$$

Nilai α tersebut selanjutnya berubah-ubah pada setiap perhitungan pasangan data seiring dengan perubahan nilai variabel dalam persamaan (3.18), dalam arti nilai α bersifat dinamis.

Algoritma *backpropagation* dengan nilai laju pembelajaran yang dinamis dengan penambahan momentum adalah sebagai berikut :

1. Tetapkan : konstanta momentum (jika ada), iterasi maksimum, *MSE* maksimum, jumlah unit lapisan masukan (*n* unit), tersembunyi (*p* unit) dan keluaran (*m* unit), jumlah data (*s* data).
2. Masukkan data *input* dan *target.*
3. Normalisasi data *input* dan *target* dengan rumus :

$$x^* = \frac{x - \text{data terkecil}}{\text{data terbesar} - \text{data terkecil}}$$

4. Inisialisasi bobot awal dengan bilangan acak yang cukup kecil.
5. Inisialisasi : iterasi = 0.
6. Kerjakan langkah-langkah berikut selama iterasi < iterasi maksimum atau *MSE* > *MSE* maksimum :
   a. iterasi = iterasi + 1.
   b. Untuk setiap pasangan data yang akan dilatih kerjakan :
      Feedforward :
      Tiap-tiap unit masukan (x_i, $i = 1,2,...,n$) menerima sinyal dan meneruskannya ke unit pada lapisan berikutnya.
      i. Hitung jumlah sinyal-sinyal terbobot masukan di unit tersembunyi (z_j, $j = 1,2,\ldots,p$) :

$$z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

dan gunakan fungsi aktivasi untuk menghitung sinyal keluarannya :

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

ii. Hitung jumlah sinyal-sinyal terbobot masukan di unit keluaran sebanyak lapisan tersembunyi sampai dengan lapisan keluaran ($y_k, k = 1,2,\ldots,m$) :

$$y\_net_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

dan gunakan fungsi aktivasi untuk menghitung sinyal keluarannya :

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

Backpropagation :

i. Hitung gradien dari fungsi kesalahan E :

$$\nabla_{w_{jk}} E = (t_k - y_k)(-z_j)$$

ii. Hitung nilai *learning-rate* (α) :

$$\alpha = \frac{w_{0k} + \sum_{j=1}^{p} z_j w_{jk} - t_k}{\nabla_{w_{0k}} E + \sum_{j=1}^{p} z_j \nabla_{w_{jk}} E}$$

iii. Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran (y_k, $k = 1,2,...,m$) :

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

Kemudian hitung suku perubahan bobot w_{jk} (yang akan dipakai untuk merubah bobot w_{jk}) dengan ketentuan :

Pada iterasi ke-1 hitung :

$$\Delta w_{jk} = \alpha \delta_k z_j$$

dengan $k = 1,2,...,m$ dan $j = 1,...,p$.

Untuk suku perubahan bobot bias w_{0k} :

$$\Delta w_{0k} = \alpha \delta_k$$

Pada iterasi ke-2 dan selanjutnya hitung :

Tanpa momentum : $\Delta w_{jk} = \alpha \delta_k z_j$

Dengan momentum : $\Delta w_{jk} = \alpha \delta_k z_j + \mu(\Delta w_{jk}$ sebelumnya)

Untuk suku perubahan bobot bias w_{0k} :

Tanpa momentum : $\Delta w_{0k} = \alpha \delta_k$

Dengan momentum : $\Delta w_{0k} = \alpha \delta_k + \mu(\Delta w_{0k}$ sebelumnya)

iv. Tiap-tiap unit tersembunyi (z_j, $j = 1, 2, ..., p$) menjumlahkan delta masukannya dari setiap unit yang berada pada lapisan sesudahnya :

$$\gamma\_net_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

Untuk menghitung informasi kesalahannya, kalikan dengan turunan fungsi aktivasinya :

$$\gamma_j = \gamma\_net_j \, f'(z\_net_j)$$

Hitung suku perubahan bobot v_{ij} yang akan dipakai untuk merubah bobot v_{ij} dengan ketentuan :

Pada iterasi ke-1 hitung :

$$\Delta v_{ij} = \alpha \gamma_j x_i$$

dengan $j = 1, 2, ..., p$ dan $i = 1, ..., n$.

Untuk suku perubahan bobot bias v_{0j} :

$$\Delta v_{0j} = \alpha \gamma_j$$

Pada iterasi ke-2 dan selanjutnya hitung :

Tanpa momentum : $\Delta v_{ij} = \alpha \gamma_j x_i$

Dengan momentum : $\Delta v_{ij} = \alpha \gamma_j x_i + \mu(\Delta v_{ij}$ sebelumnya)

Untuk suku perubahan bobot bias v_{0j} :

Tanpa momentum : $\Delta v_{0j} = \alpha \gamma_j$

Dengan momentum : $\Delta v_{0j} = \alpha \gamma_j + \mu(\Delta v_{0j}$ sebelumnya)

v. Perubahan Bobot :

Perubahan bobot garis yang menuju ke unit keluaran :

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}(t)$$

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

Perubahan bobot garis yang menuju ke unit tersembunyi :

$$v_{ij}(t+1) = v_{ij}(t) + \Delta v_{ij}(t)$$

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

c. Hitung MSE (*Mean Square Error*).

3.3 Pengolahan Data

Seperti telah dijelaskan sebelumnya bahwa data yang akan digunakan sebagai aplikasi dari model *backpropagation* di sini adalah data perdagangan saham harian Astra International, Tbk di Bursa Efek Jakarta (data pada Lampiran 1). Variabel-variabel yang dipakai sebagai data masukan adalah data harga sebelum, harga tertinggi, harga terendah, volume penjualan dan *stock index* pada hari ini. Sedangkan variabel yang dipakai sebagai data keluaran adalah data harga penutupan keesokan harinya. Jumlah total data, yaitu data *training* dan data *testing* sebanyak 209 data, masing-masing sebanyak 189 untuk data *training* dan 20 untuk data *testing*. *Input* pasangan data diacak urutannya (tidak urut berdasarkan tanggal) agar pola yang terbentuk tidak teratur atau acak sehingga diharapkan sistem akan mengenali pola data tersebut dengan baik walaupun dalam kondisi acak. Untuk pengolahan data ini digunakan program yang telah dibuat sebelumnya dengan software Delphi 7.

3.3.1 Pemilihan arsitektur jaringan

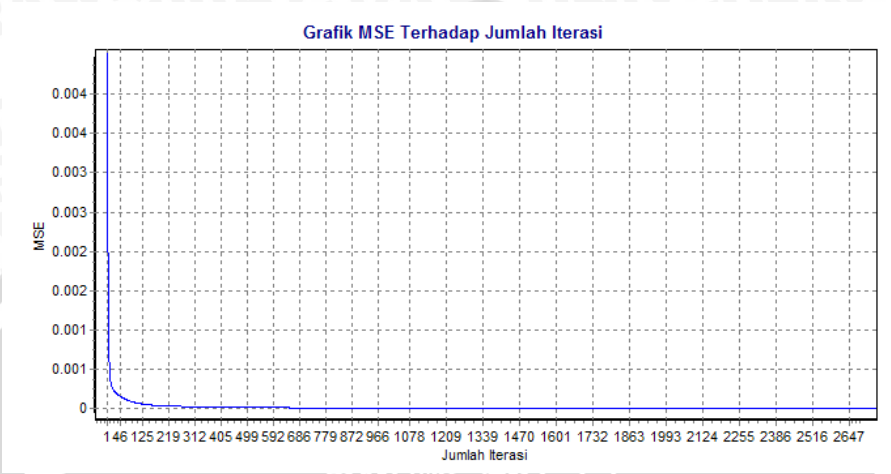
Dalam aplikasi model *backpropagation*, pemilihan arsitektur jaringan merupakan salah satu faktor penting yang mempengaruhi keberhasilan pendekatan atau keakuratan hasil, yang dalam hal ini adalah keakuratan peramalan/prediksi nilai dari harga penutupan saham Astra International, Tbk di Bursa Efek Jakarta untuk keesokan harinya. Pemilihan arsitektur jaringan dapat dilakukan dengan uji coba (*trial and error*) sebagai langkah awal dalam pengolahan data. Dalam Tugas Akhir ini, jumlah variabel yang dipakai sebagai data masukan sebanyak 5 variabel (5 neuron dalam 1 lapisan masukan) dan 1 variabel keluaran (1 neuron dalam 1 lapisan keluaran). Untuk jumlah neuron dalam lapisan tersembunyi, akan dicoba sebanyak 8 kali dengan jumlah yang berbeda, yaitu 5, 10, 15, 20, 25, 30, 35 dan 40 neuron. Penulisan arsitektur jaringan untuk selanjutnya adalah urutan dari banyaknya neuron pada lapisan masukan, lapisan tersembunyi dan lapisan keluaran, misalnya 5-10-1 artinya jumlah neuron pada lapisan masukan adalah 5, 10 neuron pada lapisan tersembunyi dan 1 neuron pada lapisan keluaran. Untuk jumlah iterasi maksimal ditetapkan sebanyak 15000 iterasi, dan nilai *MSE*

maksimal adalah sebesar 1E-7 atau 1.10^{-7}. Nilai bobot awal dipilih secara acak dengan bilangan yang cukup kecil, yaitu dalam selang [-0,5;0,5]. Berikut disajikan tabel hasil dari pengolahan data dengan 8 macam arsitektur jaringan yang berbeda dengan menggunakan program Delphi yang telah dibuat sebelumnya, dengan nilai bobot awal dituliskan pada Lampiran 2:

Tabel 3.1 Perbandingan nilai *MSE* pada tahap *training* dan *testing* dengan arsitektur jaringan yang berbeda

Training				***Testing***
Arsitektur Jaringan	**Waktu (detik)**	**Jml Iterasi**	***MSE***	***MSE***
5-5-1	66	15000	9.6800E-7	6.8251
5-10-1	60	15000	3.2986E-7	1.7918
5-15-1	70	15000	2.6680E-7	1.5893
5-20-1	62	15000	1.9900E-7	1.9488
5-25-1	68	15000	2.3500E-7	1.1514
5-30-1	14	2789	9.9999E-8	0.3590
5-35-1	40	9464	9.9999E-8	0.3522
5-40-1	65	13824	9.9999E-8	0.6506

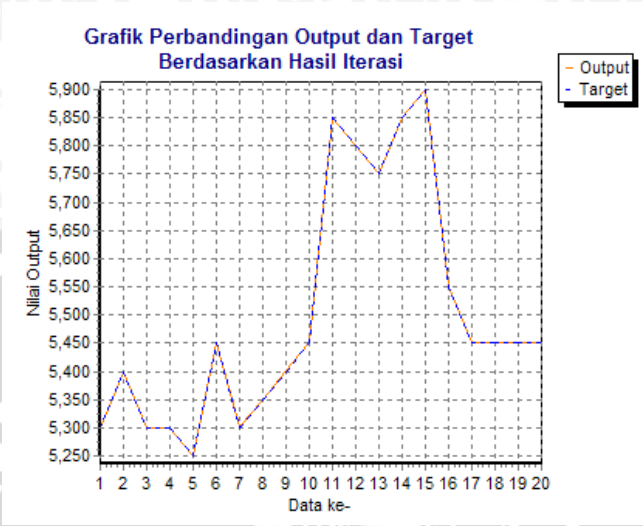
Dalam Tabel 3.1 dapat disimpulkan bahwa nilai *MSE* maksimal sebesar 1E-7 pada tahap pelatihan/*training* tercapai pada percobaan dengan arsitektur 5-30-1, 5-35-1 dan arsitektur 5-40-1. Waktu tercepat yang diperlukan dalam proses perhitungan adalah pada arsitektur 5-30-1, yaitu selama 14 detik pada iterasi ke-2789. Untuk arsitektur yang lainnya, nilai *MSE* belum tercapai sampai pada batas iterasi yang telah ditetapkan. Nilai bobot akhir yang diperoleh dari tahap pelatihan dengan arsitektur jaringan 5-30-1 dapat dilihat pada Lampiran 3. Gambar 3.1 menampilkan grafik kesalahan (*MSE*) pada tahap *training* jaringan dengan arsitektur 5-30-1.



Gambar 3.1 Grafik nilai *MSE* terhadap jumlah iterasi tahap pelatihan arsitektur 5-30-1

Dari Gambar 3.1 dapat dilihat bahwa nilai kesalahan monoton turun. Pada awal iterasi penurunan nilai kesalahan sangat drastis. Dan untuk berikutnya penurunan tidak terlalu signifikan. Apabila diinginkan agar waktu perhitungannya tidak terlalu lama, proses pelatihan dapat dihentikan pada saat perubahan nilai *MSE*-nya dianggap cukup kecil.

Pada tahap *testing*, nilai *MSE* jaringan dengan arsitektur 5-30-1 adalah sebesar 0,3590. Nilai tersebut ternyata masih sedikit lebih besar dari nilai *MSE* pada arsitektur 5-35-1 sebesar 0,3522. Tetapi melihat waktu pelatihan yang digunakan cukup jauh lebih cepat pada arsitektur 5-30-1 dibandingkan dengan arsitektur 5-35-1, maka pemilihan arsitektur terbaik pada percobaan di atas adalah 5-30-1. Grafik yang menunjukkan selisih atau perbandingan antara data keluaran jaringan dengan nilai data sebenarnya ditunjukkan pada Gambar 3.2. Sedangkan selisih antara nilai keluaran jaringan dengan nilai data sebenarnya, dalam hal ini adalah besar harga penutupan saham, ditunjukkan dalam Tabel 3.2.



Gambar 3.2 Grafik perbandingan nilai keluaran jaringan dengan nilai keluaran sebenarnya (harga penutupan) arsitektur 5-30-1

Tabel 3.2 Selisih nilai keluaran jaringan dengan nilai data sebenarnya

Data ke-	Harga Penutupan Sebenarnya	Harga Penutupan Keluaran Jaringan	Selisih
1.	5300	5300,5	0,5
2.	5400	5400,1	0,1
3.	5300	5299,6	-0,4
4.	5300	5300,4	0,4
5.	5250	5250,2	0,2
6.	5450	5450,6	0,6
7.	5300	5301	1
8.	5350	5350,4	0,4
9.	5400	5400,4	0,4
10.	5450	5451,1	1,1
11.	5850	5850,8	0,8
12.	5800	5801	1
13.	5750	5750,6	0,6
14.	5850	5851	1
15.	5900	5900,1	0,1
16.	5550	5550,7	0,7

Data ke-	Harga Penutupan Sebenarnya	Harga Penutupan Keluaran Jaringan	Selisih
17.	5450	5450,2	0,2
18.	5450	5449,9	-0,1
19.	5450	5449,9	-0,1
20.	5450	5450,2	0,2
MSE			**0,3590**

Dari Tabel 3.2, dapat dilihat bahwa selisih nilai keluaran jaringan dengan data sebenarnya cukup kecil. Ini menunjukkan bahwa pendekatan dari hasil simulasi dengan arsitektur jaringan 5-30-1 sudah cukup baik. Sehingga untuk selanjutnya arsitektur jaringan 5-30-1 akan digunakan pada pengolahan data dengan model *backpropagation* yang dimodifikasi dengan penambahan momentum pada proses perubahan bobotnya.

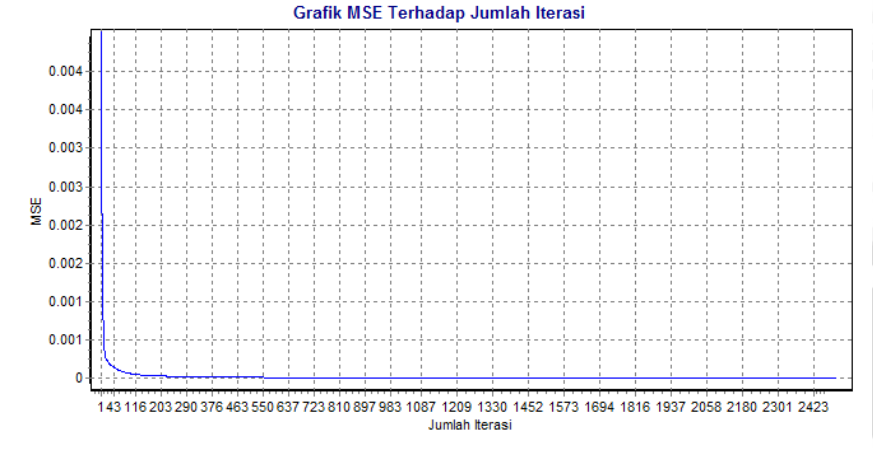
3.3.2 Pengolahan data dengan penambahan konstanta momentum pada perubahan bobot dalam sistem

Tahap selanjutnya setelah arsitektur jaringan ditentukan, data *training* diolah lagi dengan model *backpropagation* yang telah dimodifikasi dengan menambahkan konstanta momentum μ pada proses perubahan bobotnya. Dalam tahap ini juga perlu dilakukan uji coba menggunakan nilai konstanta momentum yang berbeda. Akan dicoba dengan nilai μ = 0,1; 0,2; 0,3;…; 1. Nilai bobot awal yang digunakan pada tahap pelatihan di sini adalah bobot awal yang juga digunakan dalam tahap pelatihan dengan arsitektur jaringan 5-30-1 sebelumnya. Dan bobot yang digunakan dalam tahap pengujian nantinya adalah bobot akhir yang diperoleh dari tahap pelatihan yang dapat dilihat pada Lampiran 4. Hasil dari pelatihan dan pengujian dengan arsitektur 5-30-1 dapat dilihat dalam Tabel 3.3:

Tabel 3.3 Perbandingan nilai *MSE* dan jumlah iterasi dengan penambahan konstanta momentum yang berbeda

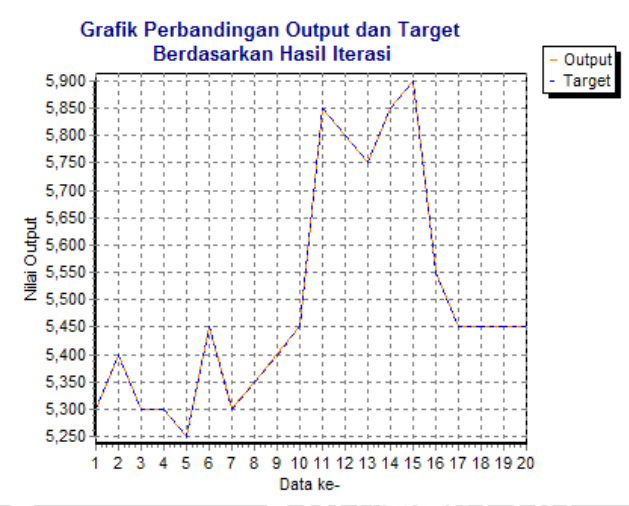
Training					*Testing*
μ	**Arsitektur Jaringan**	**Waktu (detik)**	**Jml Iterasi**	**MSE**	**MSE**
0.1	5-30-1	10	2589	1E-7	0.2729
0.2	5-30-1	10	2503	9.997E-8	0.2258
0.3	5-30-1	11	2844	9.999E-8	0.2377
0.4	5-30-1	19	4720	9.999E-8	0.4689
0.5	5-30-1	5	500	7.264E+13	6.980E+19
0.6	5-30-1	5	500	1.192E+14	1.117E+20
0.7	5-30-1	5	500	2.993E+14	6.533E+21
0.8	5-30-1	5	500	7.901E+14	1.724E+22
0.9	5-30-1	5	500	1,299E+16	2.836E+23
1	5-30-1	5	500	4.372E+36	9.589E+43

Dari Tabel 3.3 dapat disimpulkan bahwa dengan menambahkan konstanta momentum yang berbeda dalam model *backpropagation*, hasil yang diperoleh juga berbeda. Pada penambahan konstanta momentum sebesar 0,2 jaringan dapat melatih dan mengenali pola data dengan cukup baik. Nilai *MSE* yang dicapai pada tahap pelatihan sebesar 9,997E-8, yaitu pada iterasi yang ke-2503 dalam waktu 10 detik. Nilai *MSE* pada tahap pengujian sebesar 0,2258. Hasil ini lebih baik dibandingkan dengan pengolahan data dengan model *backpropagation* standar sebelumnya (tanpa penambahan momentum) dengan nilai *MSE* yang sama pada tahap pelatihan dan nilai *MSE* sebesar 0,3590 pada tahap pengujian pada iterasi ke-2789. Grafik kesalahan (*MSE*) terhadap jumlah iterasi pada tahap pelatihan dengan penambahan μ = 0,2 dapat dilihat pada Gambar 3.3.



Gambar 3.3 Grafik nilai *MSE* terhadap jumlah iterasi tahap pelatihan arsitektur 5-30-1 dengan μ = 0,2

Sama seperti pada proses pelatihan arsitektur 5-30-1 tanpa penambahan momentum, nilai kesalahannya monoton turun. Pada awal iterasi penurunannya sangat drastis, dan berikutnya perubahan nilai kesalahan tidak terlalu signifikan, sehingga proses dapat dihentikan pada saat perubahan *MSE*-nya dianggap cukup kecil apabila diinginkan waktu pelatihan yang cepat. Grafik perbandingan antara nilai keluaran jaringan dengan nilai dari harga penutupan sebenarnya dapat dilihat pada Gambar 3.4.



Gambar 3.4 Grafik perbandingan nilai keluaran jaringan dengan nilai harga penutupan sebenarnya, arsitektur 5-30-1 dengan μ = 0,2

Selisih nilai keluaran jaringan dengan nilai yang sebenarnya disajikan dalam Tabel 3.4 berikut:

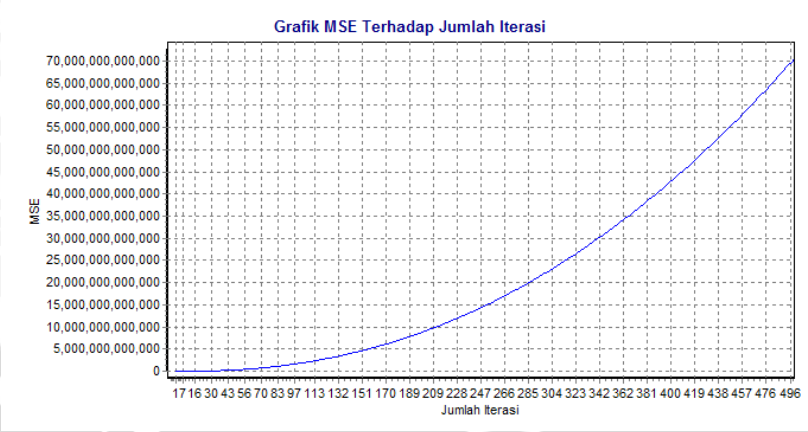
Tabel 3.4 Selisih nilai keluaran jaringan dengan nilai data sebenarnya pada arsitektur 5-30-1 dengan μ = 0,2

Data ke-	Harga Penutupan Sebenarnya	Harga Penutupan Keluaran Jaringan	Selisih
1.	5300	5300,1	0,1
2.	5400	5400	0
3.	5300	5299,7	-0,3
4.	5300	5300,1	0,1
5.	5250	5249,9	-0,1
6.	5450	5450,4	0,4
7.	5300	5300,9	0,9
8.	5350	5350,1	0,1
9.	5400	5400,2	0,2
10.	5450	5450,6	0,6
11.	5850	5850,5	0,5
12.	5800	5800,9	0,9
13.	5750	5750,8	0,8

Data ke-	Harga Penutupan Sebenarnya	Harga Penutupan Keluaran Jaringan	Selisih
14.	5850	5850,6	0,6
15.	5900	5899,8	-0,2
16.	5550	5550,7	0,7
17.	5450	5450,3	0,3
18.	5450	5450	0
19.	5450	5450,1	0,1
20.	5450	5450,2	0,2
MSE			**0,2258**

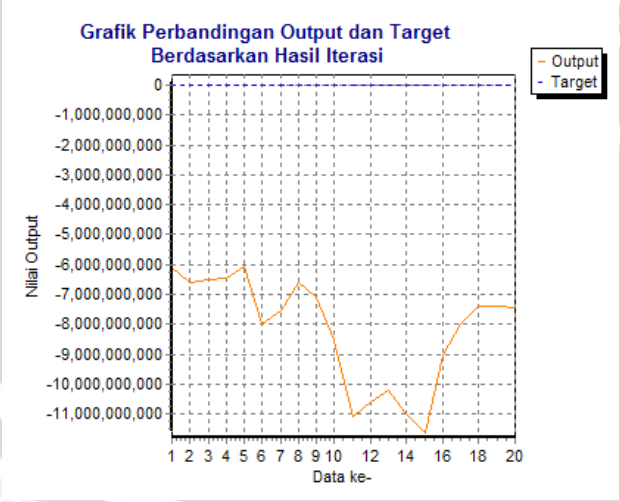
Dalam percobaan lainnya dengan menambahkan konstanta momentum sebesar 0,1 dan 0,3, nilai *MSE* yang dicapai juga lebih baik dibandingkan dengan model *backpropagation* standar dengan arsitektur yang sama. Pada tahap pelatihan, nilai *MSE* yang diperoleh masing-masing sebesar 1E-7 dan 9,999E-8, dan pada tahap pengujian masing-masing sebesar 0,2729 dan 0,2377. Namun disini perolehan nilai tersebut terjadi pada iterasi yang ke-2589 dalam waktu 10 detik dan ke-2844 dalam waktu 11 detik, yang berarti sedikit lebih lama dibandingkan dengan pelatihan pada percobaan dengan konstanta momentum 0,2. Hal tersebut dapat dijadikan sebagai alternatif. Dengan perolehan nilai *MSE* yang lebih baik, sistem juga mengenali pola data lebih baik atau akurat. Karena selisih nilai *MSE* tidak terlalu jauh dan waktu yang dibutuhkan dalam pelatihan hanya terpaut 1 detik saja, maka penambahan μ = 0,3 juga dapat dijadikan sebagai pilihan.

Pada pemilihan konstanta momentum $\mu \geq 0,5$ dari Tabel 3.3 dapat dilihat bahwa nilai *MSE* yang diperoleh sangat besar, yang berarti bahwa selisih atau kesalahan hasil yang dicapai sangat besar. Tentunya hasil peramalan/prediksi yang akan dilakukan nantinya juga akan sangat buruk, karena sistem tidak mampu mengenali pola data dengan baik. Maka dalam kasus ini pemilihan konstanta momentum $\mu \geq 0,5$ sangat tidak dianjurkan. Gambar 3.6 memperlihatkan grafik perubahan nilai *MSE* terhadap jumlah iterasi pada tahap pelatihan model *backpropagation* dengan arsitektur 5-30-1 dengan konstanta momentum μ = 0,5.



Gambar 3.5 Grafik nilai *MSE* terhadap jumlah iterasi tahap pelatihan arsitektur 5-30-1 dengan μ = 0,5

Dari Gambar 3.5, dapat dilihat bahwa nilai *MSE* semakin besar seiring dengan semakin banyaknya jumlah iterasi (monoton naik). Berarti pengambilan nilai konstanta momentum μ = 0,5 tidak baik. Dengan nilai kesalahan yang besar, tentu saja hasil peramalan yang akan dilakukan nantinya sangat tidak akurat. Hal ini dapat dilihat pada Gambar 3.6 yang menampilkan grafik perbandingan antara nilai keluaran jaringan dengan nilai dari harga penutupan sebenarnya.



Gambar 3.6 Grafik perbandingan nilai keluaran jaringan dengan nilai harga penutupan sebenarnya, arsitektur 5-30-1 dengan μ = 0,5

BAB IV
KESIMPULAN DAN SARAN

4.1 Kesimpulan

Dari hasil pembahasan dapat ditarik beberapa kesimpulan, yaitu :

1. Pada model *backpropagation*, metode *gradient-descent* digunakan pada proses perubahan bobot, yaitu pada tahap *backpropagation*. Persamaan metode *gradient-descent* adalah $w_{k+1} = w_k - \alpha_k \nabla E(w_k)$. Pada tahap *backpropagation* dihasilkan suatu rumusan untuk mencari nilai suku perubahan bobot yang nantinya digunakan untuk memperbarui nilai bobot yang lama, yaitu $\Delta w_{jk} = \alpha \delta_k z_j$ dan $\Delta v_{ij} = \alpha \gamma_j x_i$ yang diturunkan dari metode *gradient-descent*.
2. Nilai laju pembelajaran (*learning rate*) pada metode *gradient-descent* dicari dengan metode *line-search* untuk mengoptimalkan proses perubahan bobot. Nilai laju pembelajaran α yang diperoleh adalah

$$\alpha = \frac{w_{0k} + \sum_{j=1}^{p} z_j w_{jk} - t_k}{\nabla_{w_{0k}} E + \sum_{j=1}^{p} z_j \nabla_{w_{jk}} E}$$

3. Penambahan konstanta momentum $\mu = 0,2$ memperbaiki kinerja model *backpropagation* standar dengan arsitektur 5-30-1, dengan hasil sebagai berikut:
   a. Dalam model *backpropagation* standar, pada proses pelatihan didapat nilai *MSE* sebesar 9,9999E-8, tercapai pada iterasi ke-2789 dalam waktu 14 detik. Pada proses pengujian didapat *MSE* sebesar 0,3590.
   b. Dalam model *backpropagation* dengan momentum ($\mu = 0,2$), pada proses pelatihan didapat nilai *MSE* sebesar 9,997E-8, tercapai pada iterasi ke-2503 dalam waktu 10 detik. Pada proses pengujian didapat *MSE* sebesar 0,2258.

   Penambahan $\mu \geq 0,5$ tidak memperbaiki kinerja *backpropagation* standar.

4.2 Saran

Untuk penulisan berikutnya penulis memberikan saran-saran sebagai berikut:

1. Untuk mendapatkan hasil peramalan yang lebih akurat dapat digunakan metode optimasi yang lebih baik, misalnya metode *Conjugate Gradient-Descent, Quasi-Newton* dan lain sebagainya.
2. Untuk pencarian nilai laju pembelajaran α dapat digunakan metode lain yang disesuaikan dengan fungsi aktivasi pada lapisan keluaran.
3. Dalam proses pemilihan arsitektur jaringan disarankan untuk mencoba terlebih dahulu setiap kombinasi dari variabel data masukan.

DAFTAR PUSTAKA

Anonymous. 2007. *Mean Squared Error*. Http://www.wikipedia.org, tanggal akses : 27 November 2007.

Anonymous. 2007. *Nonlinear System*. Http://www.wikipedia.org, tanggal akses : 16 Desember 2007.

Bose, N. K. dan Liang, P. 1996. *Neural Network Fundamentals With Graphs, Algorithms, and Applications.* McGraw-Hill, Inc. Singapore.

Boyd, S. dan Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press. UK.

Freund, R. 2004. *The Steepest Descent Algorithm for Unconstrained Optimization*. Massachusetts Institute of Technology.

Kaplan, W. 1962. *Advance Calculus.* First Edition. Addison-Wesley Publishing Company, Inc. USA.

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

Leon, S. J. 1998. *Aljabar Linier dan Aplikasinya*. Erlangga. Jakarta.

Lewin, J dan Lewin, M. 1993. *An Introduction to Mathematical Analysis*. McGraw-Hill, Inc. Singapore.

Luknanto, D. 2000. *Pengantar Optimasi Nonlinier*. Jurusan Teknik Sipil Fakultas Teknik Universitas Gadjah Mada. Yogyakarta.

Mathews, J. H. 2004. *Modul for Steepest Descent or Gradient Method* . Http://math.fullerton.edu, tanggal akses : 31 Juli 2006.

Milagefira, F. M. 2006. *Penerapan Artificial Neural Network dengan Arsitektur Multilayer Feedforward dan Algoritma Pelatihan Backpropagation pada Peramalan Harga Saham di Bursa Efek Jakarta*. Skripsi. Jurusan Statistika. Fakultas MIPA. Universitas Brawijaya. Malang.

Purnomo, M dan Kurniawan, A. 2006. *Supervised Neural Networks dan Aplikasinya.* Graha Ilmu. Yogyakarta.

Schalkoff, R. 1997. *Artificial Neural Network.* McGraw-Hill.Singapore.

Siang, J. J. 2004. *Jaringan Syaraf Tiruan & Pemrogramannya Menggunakan Matlab.* Andi. Yogyakarta.

Stewart, J. 2001. *Kalkulus*. Jilid 1. Erlangga. Jakarta.