IJCCS, Vol.6, No.2, July 2012, pp.47~56
ISSN: 1978-1520

Penerapan Algoritma Optimasi Chaos pada Jaringan Ridge Polynomial untuk Prediksi Jumlah Pengangguran

Rina Pramitasari*[1], Retantyo Wardoyo[2]
[*1]Program Studi S2/S3 Ilmu Komputer, FMIPA UGM, Yogyakarta
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: [*1]**rina.pramitasari@gmail.com**, [2]rw@ugm.ac.id

Abstrak

Ridge polynomial neural network (RPNN) awalnya diusulkan oleh Shin dan Ghosh, dibangun dari jumlah peningkatan order pi-sigma neuron (PSN). RPNN mempertahankan pembelajaran cepat, pemetaan yang kuat dari layer tunggal higher order neural network (HONN) dan menghindari banyaknya bobot karena meningkatnya sejumlah input. Algoritma optimasi chaos digunakan dengan memanfaatkan persamaan logistik yang sensitif terhadap kondisi awal, sehingga pergerakan chaos dapat berubah di setiap keadaan dalam skala tertentu menurut keteraturan, ergodik dan mempertahankan keragaman solusi.

Algoritma Optimasi Chaos diterapkan pada RPNN dan digunakan untuk prediksi jumlah pengangguran di Kalimantan Barat. Proses pelatihan jaringan menggunakan ridge polynomial neural network, sedangkan pencarian nilai awal bobot dan bias jaringan menggunakan algoritma optimasi chaos. Struktur yang digunakan terdiri dari 6 neuron layer input dan 1 neuron layer output. Data diperoleh dari Badan Pusat Statistik.

Hasil dari penelitian ini menunjukkan bahwa algoritma yang diusulkan dapat digunakan untuk prediksi.

Kata kunci—*prediksi jumlah pengangguran, jaringan syaraf tiruan, algoritma optimasi chaos, ridge polynomial neural network*

Abstract

Ridge polynomial neural network was initially proposed by Shin and Ghosh, made of total increased pi-sigma neural (PSN) orders. Ridge polynomial neural network maintains quick learning, strong mapping of single layer of higher order neural network (HONN) and avoids many weights because total increased inputs. Chaos optimization algorithm is used by utilizing sensitive logistic equation to initial condition, so that chaos movement can change in each condition in specific scale according to orderliness, ergodic, and maintaining solution variety.

Chaos optimization algorithm is applied to ridge polynomial neural network and used to predict total unemployed persons in West Kalimantan. Network training process used ridge polynomial neural network; while, initial values and weights and bias of network were found using Chaos optimization algorithm. Structure used consisted of 6 input layer neurons and one output layer neuron. Data were obtained from Central Statistic Agency.

The results of research indicated that algorithm proposed could be used to predict

Keywords— *predict the number of unemployed, neural networks, chaos optimization algorithm, ridge polynomial neural network*

Received March 20th,2012; Revised June 20th, 2012; Accepted June 29th, 2012

1. PENDAHULUAN

Ridge polynomial neural network (RPNN) awalnya diusulkan oleh Shin dan Ghosh, dibangun dari jumlah peningkatan order pi-gma neural (PSN). RPNN mempertahankan pembelajaran cepat, pemetaan yang kuat dari layer tunggal higher order neural network (HONN) dan menghindari ledakan bobot karena meningkatnya sejumlah input [1]. Chaos [2] adalah gejala nonlinear secara universal dalam semua bidang ilmu, yang terjadi di dalam sistem. Variabel chaos di dalam range tertentu memiliki beberapa fitur sebagai berikut: randomness, ergodicity, keteraturan (regularity) dan sensitif terhadap kondisi awal.

RPNN tidak menjamin adanya hasil akurasi yang optimal. Salah satu kondisi yang menyebabkan ketidakmampuan hasil akurasi yang optimal adalah terjebak atau konvergensi prematur pada minimum lokal.Dan menjadi lebih rumit jika ada beberapa minimum lokal.Algoritma yang dipakai untuk penyelesaian permasalahan hasil akurasi yang optimal adalah algoritma optimasi chaos dengan memanfaatkan persamaan logistik yang sensitif terhadap kondisi awal, sehingga pergerakan chaos dapat pergi di setiap keadaan dalam skala tertentu menurut keteraturan, ergodocity dan mempertahankan keragaman solusi.Sehingga dapat diusulkan dalam perancangan optimasi untuk menemukan solusi minimal global atau menghindari optimal lokal. Hipotesanya adalah algoritma optimasi chaos yang mampu menghindari optimal local diharapkan mampu menjamin adanya hasil akurasi yang optimal dari ridge polynomial neural network

Setiap hari sebagian pekerja kehilangan atau keluar dari pekerjaannya, dan sebagian lagi yang menganggur diterima bekerja.Sehingga tingkat pengangguran memiliki sifat normalisasi antara 0 sampai 1 bukan tak terhingga, karena terikat oleh batasan angkatan kerja.Salah satu sifat persamaan logistik adalah normalisasi antara 0 sampai 1 bukan tak terhingga, karena terikat oleh batasan fisik dari lingkungannya.Sehingga hipotesa bahwa persamaan logistik dari chaos diharapkan mampu untuk memprediksi jumlah pengangguran berdasarkan kesamaan diatas.

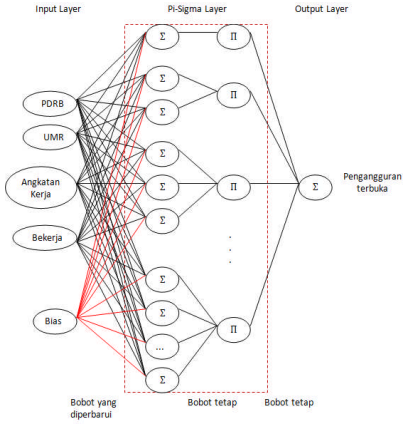
Algoritma optimasi chaos diterapkan pada RPNN dan digunakan untuk prediksi jumlah pengangguran di Kalimantan Barat.Sehingga untuk memprediksi jumlah pengangguran digunakan RPNN sebagai pelatihan jaringan syaraf tiruan.Sedangkan untuk mengatasi terjebak ke dalam minimum lokal, maka diusulkan algoritma optimasi chaos (COA) sebagai nilai awal bobot dan bias jaringan syaraf tiruan.

2. METODE PENELITIAN

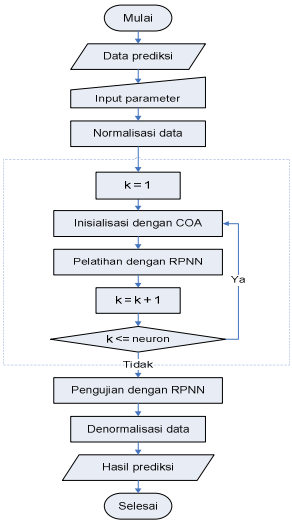
Diagram dari metode yang digunakan untuk melakukan prediksi jumlah penganggran ditunjukkan pada Gambar 2. Berikut adalah Gambar 1 arsitektur jaringan untuk prediksi jumlah pengangguran.

2.1Proses input parameter

Merupakan proses dimana memasukkan parameter awal. Pelatihan jaringan menggunakan parameter di jaringan ridge polynomial hasil rekomendasi dari [3] dan di algoritma optimasi chaos hasil rekomendasi dari [4] dan [5]. Parameter di jaringan ridge polynomial yaitu learning rate dimana ketika setiap kali PSN ditambahkan learning rate menurun dengan pembagi 1.7, error pada PSN awal = 0.00007 ketika setiap kali PSN ditambahkan menurun dengan pembagi 10. Parameter di algoritma optimasi chaos yaitu maksimum iterasi pencarian pertama N_1 = 5000, maksimum iterasi pencarian kedua N_2 = 10000 dan rentang penyempit p adalah 0.1. Interval percarian pertama = [-50, +50] dan interval pencarian kedua = [-2, +2].



Gambar 1 Arsitektur jaringan Syaraf Tiruan



Gambar 1 Diagram alir sistem secara umum

2.2 Proses standarisasi data

Merupakan proses dimana seluruh data dinormalkan agar berada pada rentang nilai [0, 1].

Rumus yang digunakan pada proses ini menurut [6]adalah :

$$N' = (N - Nmin)/(Nmax - Nmin) \quad (1)$$

Dimana

N' = Data yang sudah normalisasi

N = Data yang akan dinormalisasi

Nmax = Data terbesar dari sekumpulan data

Nmin = Data terkecil dari sekumpulan data

2.3Proses inisialisasi

Merupakan proses inisialisasi bobot dan bias pada pelatihan jaringan syaraf tiruan, setiap kali neuron layer PSN ditambahkan. Algoritma yang digunakan untuk menetukan nilai bobot dan bias awal adalah algoritma optimasi chaos (COA).

Masalah fungsi optimasi menurut [5] dapat dirumuskan sebagai berikut :

$$fo = \min f(X),\ X = [x_1, x_2, \ldots, x_n] \quad (2)$$

$$x_i \square [a_i, b_i],\ i = 1, 2, \ldots, n$$

Dimana *fo* adalah fungsi objektif, dan *x* adalah keputusan vektor berisikan *n* variabel.

Langkah-langkah algortima optimisasi chaos sebagai berikut

Pada pencarian pertama

Langkah 1 : Inisialisasi

r = 0, *r* adalah variabel tanda untuk iterasi variabel chaos. Tetapkan jumlah iterasi maksimum *r max*. Mengisi variabel chaos awal dengan random, $A_i^0 \in (0,1)$, dimana *i* = 1, 2, ..., *n* adalah jumlah bobot diantara layer input dan layer hidden ditambah bobot bias. Pastikan $A_i^0 \neq (0, 0.25, 0.5, 0.75, 1)$.Pada penelitian ini menetapkan interval untuk $a^1 = -50$ dan $b^1 = 50$.

Tetapkan nilai $A^* = 0$, $X^* = 0$ dan F^* dengan nilai besar. Dimana A^* adalah nilai terbaik dari variabel chaos, X^* adalah nilai terbaik yang akan dijadikan nilai bobot dan bias pada jaringan syaraf tiruan dan F^* adalah fungsi terbaik yang diperoleh dari error jaringan syaraf tiruan.

Langkah 2 : Memetakan variabel chaos A_i^r ke dalam rentang variabel optimisasi $X_i^r \in [a^1, b^1]$ dengan persamaan

$$X_i^r = a^1 + A_i^r (b^1 - a^1), \quad (3)$$

dimana :

X_i^radalah variabel optimisasi

A_i^radalah variabel chaos

r adalah variabel tanda untuk iterasi variabel chaos pada pencarian pertama

i = 1, 2, ...,*n* adalah banyaknya jumlah variabel

a^1adalah interval paling bawah pada pencarian pertama

b^1adalah interval paling atas pada pencarian pertama

Langkah 3 : Menghitung fungsi objektif $F(X^r)$ dengan persamaan MSE

Jika $F(X^r) < F^*$ maka $A^* = A^r$, $X^* = X^r$ dan $F^* = F(X^r)$.

Jika $F(X^r) \geq F^*$ maka tinggalkan X^r

Langkah 4 : Menghitung variabel chaos dengan persamaan

$$A_i^{r+1} = \mu A_i^r (1 - A_i^r) \quad (4)$$

dan*r* = *r* + 1

dimana :

A_i^{r+1}adalah variabel chaos iterasi ke-*r+1*

A_i^radalah variabel chaos iterasi ke-*r*

r adalah variabel tanda untuk iterasi variabel chaos pada pencarian pertama

i = 1, 2, ...,*n* adalah banyaknya jumlah variabel

Langkah 5 :

Jika *r* belum mencapai r max, maka ulangi langkah 2 – 4.Jika sudah, maka lanjut ke langkah 6.

Pada Pencarian kedua

h adalah variabel tanda untuk iterasi variabel chaos. Tetapkan jumlah iterasi maksimum *h max*. Pada penelitian ini menetapkan interval untuk $a^2 = -10$ dan $b^2 = 10$.

Langkah 6 : Memetakan variabel chaos A_i^h ke dalam rentang variabel optimisasi $X_i^h \in [a^2, b^2]$ dengan persamaan

$$X_i^h = X_i^* + \omega (2 * A_i^h - 1), \; i = 1, 2, \ldots, n \quad (5)$$

$$\text{Dan } \omega = \rho (b^2 - a^2), \; \rho \in (0, 0.5)$$

dimana :

X_i^hadalah variabel optimisasi

A_i^hadalah variabel chaos

h adalah variabel tanda untuk iterasi variabel chaos pada pencarian kedua

i = 1, 2, ...,*n* adalah banyaknya jumlah variabel

a^2adalah interval paling bawah pada pencarian kedua

b^2adalah interval paling atas pada pencarian kedua

ρadalah faktor ruang penyempit pada pencarian kedua, pada penelitian ini menetapkan ρ = 0,4

Langkah 7 : Menghitung fungsi objektif $F(X^h)$ dengan persamaan MSE

Jika $F(X^h) < F^*$ maka $A^* = A^h$, $X^* = X^h$ dan $F^* = F(X^h)$.

Jika $F(X^h) \geq F^*$ maka tinggalkan X^h
Langkah 8 : Menghitung variabel chaos dengan persamaan

$$A_i^{h+1} = \mu A_i^h (1 - A_i^h) \quad (6)$$

dan$h = h + 1$
dimana :
A_i^{h+1}adalah variabel chaos iterasi ke-$h+1$
A_i^hadalah variabel chaos iterasi ke-h
h adalah variabel tanda untuk iterasi variabel chaos pada pencarian kedua
$i = 1, 2, ...,n$ adalah banyaknya jumlah variabel
Langkah 9 :
Jika h belum mencapaih_{max}, maka ulangi langkah 6 – 8 Jika sudah, maka COA diakhiri dan X^* adalah solusinya.

2.4Proses pelatihan

Merupakan proses untuk mendapatkan nilai bobot dan bias yang optimal menggunakan metode ridge polynomial neural network (RPNN). Arsitektur yang akan dilatih menggunakan jumlah neuron pi-sigma neural (PSN) antara 1 sampai 7.

Neuron PSN memiliki neuron hidden PSN yang berbeda-beda tetapi berurutan.Misalkan, neuron PSN memiliki jumlah neuron hidden di layer hidden adalah 1, maka disebut PSN order 1. Neuron PSN memiliki jumlah neuron hidden di layer hidden adalah 2, maka disebut PSN order 2, dan seterusnya. Saat proses pelatihan jaringan syaraf tiruan, pertama jaringan memiliki PSN order 1. Jaringan tersebut menginisialisasi bobot dan bias awal dengan algoritma optimasi chaos dan dilatih dengan algoritma PSN, setelah tercapai kondisi rasio yang diinginkan, kemudian hasil pelatihan bobot dan bias PSN order 1 dibekukan atau disimpan. Kemudian PSN order 2 ditambahkan, menginisialisasi bobot dan bias awal dengan algoritma optimasi chaos dan dilatih dengan algoritma PSN. kemudian hasil pelatihan bobot dan bias PSN order 2 dibekukan atau disimpan, dan seterusnya. Berikutnya setiap hasil pelatihan bobot dan bias masing-masing PSN order tersebut, ditambahkan ke neuron RPNN dan menjadi output RPNN.

1. *Jaringan Pi-Sigma (PSN)*

Jaringan pi-sigma adalah jaringan feedforward yang menghasilkan perkalian dari penjumlahan komponen-komponen input. Jaringan pi-sigma terdiri dari lapisan input, lapisan single hidden berupa unit penjumlahan dan lapisan output berupa unit perkalian. Bobot yang menghubungkan neuron input ke neuron dari lapisan single hidden dimana diperbarui selama proses pelatihan. Sedangkan, bobot yang menghubungkan neuron dari lapisan single hidden ke neuron dari lapisan output adalah tetap (*fixed*).Derajat jaringan pi-sigma adalah jumlah unit dari lapisan hidden.

Penerapan jaringan pi-sigma didasari karena pembelajaran yang cepat, kemampuan pemetaan yang kuat dari jaringan syaraf higher-order dan menghindari jumlah bobot yang diperlukan meningkat secara kombinatorial.Jaringan pi-sigma memiliki struktur higher-order yang teratur dan memerlukan jumlah bobot lebih sedikit daripada jaringan syaraf higher-order (HONN) [7].

Persamaan output jaringan pi-sigmamengacu dari [8] dapat dijelaskan sebagai berikut

$$y = f\left(\prod_{i=1}^{N}\left[\sum_{j=1}^{n} w_{ij}x_j + w_i\right]\right) \quad (7)$$

dimanaw_{ij} adalah bobot yang akan diperbarui. $w_i$adalah threshold yang disesuaikan ke neuron hidden ke-j. $f(x)$dinotasikan fungsi aktivasi nonlinear. Total jumlah bobot untuk jaringn pi-sigma order N dengan n input adalah $(n+1)N$.

2. *Jaringan Ridge Polynomial (RPNN)*

Menurut [3] PSN hanya memberikan aproksimasi yang terbatas.Karena terpotongnya kemampuan aproksimasi, PSN tidak dapat secara merata mengaproksimasi semua fungsi

multivariate continu yang didefinisikan pada himpunan compact.Bagaimanapun, aproksimasi menyeluruh dapat dicapai dengan menjumlahkan output dari beberapa PSN yang berbeda order.Hasil penggabungan jaringan dari PSN disebut *jaringan ridge polynomial*.

Jaringan ridge polynomial adalah generalisasi dari jaringan pi-sigma yang menggunakan bentuk khusus dari *ridge polynomial*.

Untuk $x=[x_1 \cdots x_n]^T$ dan $w=[w_1 \cdots w_n]^T \in R^n$, diberikan

$$\langle x, w\rangle = \sum_{i=1}^{n} w_i x_i \tag{8}$$

$\langle x, w\rangle$didefinisikan sebagai inner product antara dua vektor

Definisi 3.2 Diberikan himpunan compact $K \subset R^d$*, semua fungsi didefinisikan pada K dengan bentuk,*

$$f(\langle ., w\rangle): K \mapsto R$$

dimana $w \subset R^d$ *dan* $f(.): R \mapsto R$ *adalah kontinu, disebutfungsi ridge*

Ridge polynomial adalah fungsi ridge yang dapat digambarkan sebagai :

$$\sum_{i=0}^{N}\sum_{j=j}^{M} a_{ij} \langle x, w_{ij}\rangle^j$$

untuk beberapa $a_{ij} \in R$ dan $\langle x, w_{ij}\rangle \in R^d$

Sebuah teorema ditampilkan menyatakan setiap polynomial multivariate dapat digambarkan dalam istilah ridge polynomial, dan dapat direalisasikan dengan jaringan ridge polynomial yang disesuai kan (Teorema 3.1)

Teorema 3.1 Setiap polynomial multivariate dapat digambarkan sebagai ridge polynomial.

$$p(x) = \sum_{j=0}^{k}\sum_{m=1}^{n_j} c_{jm} x^{jm} \leftrightarrow p(x) = \sum_{j=1}^{N}\prod_{i=1}^{j}\left(\langle x, w_{ji}\rangle + w_{ji}\right)$$

Jaringan ridge polynomial memiliki kemampuan pemetaan yang bagus didalam artinya bahwa setiap fungsi kontinu pada himpunan *compact* di R^d dapat diaproksimasi secara merata dengan jaringan (Teorema 3.2).

Teorema 3.2 Setiap fungsi kontinu pada himpunan compact di R^d *dapat diaproksimasi secara merata dengan jaringan ridge polynomial.*

Jaringan ridge polynomial adalah efisien dalam artian bahwa memanfaatkan polynomial univariate yang mudah ditangani, berbeda dengan jaringan higher-order lainnya yang menggunakan polynomial multivariate yang menyebabkan ledakan jumlah bobot.Selain itu, jaringan ridge polynomial mengarah ke struktur yang teratur dibandingkan dengan jaringan higher-order biasa, dan mempertahankan pembelajaran yang cepat.

Dari teorema di atas dapat diturunkan bahwa jaringan ridge polynomial dapat dibentuk dengan penambahan jaringan pi-sigma yang berbeda derajat dan struktur baru ini memiliki kemampuan aproksimasi yang sama dengan polynomial multivariate biasa. Jadi, sebuah fungsi yang tidak diketahui f didefinisikan pada himpunan compact $k \subset R^d$ dapat diaproksimasi dengan jaringan ridge polynomial seperti berikut :

$$f(x) \cong (\langle x, w_{11}\rangle + w_{11}) + (\langle x, w_{21}\rangle + w_{21}) \cdot (\langle x, w_{22}\rangle + w_{22}) + \cdots + (\langle x, w_{N1}\rangle + w_{N1}) \cdots (\langle x, w_{NN}\rangle + w_{NN})$$

dimana setiap perkalian diperoleh sebagai output dari jaringan pi-sigma dengan unit output.

Persamaan output jaringan ridge polynomial yang mengacu dari [8] dapat dijelaskan sebagai berikut

$$y = \varphi\left(\sum_{j=1}^{N}\prod_{i=1}^{j}\left(\langle x, w_{ij}\rangle + \theta_{ji}\right)\right)$$

Atau

$$y = f\left(\sum_{j=1}^{N}\prod_{i=1}^{j}\left(\sum_{k=1}^{n} w_{ijk} x_k + \theta_{ji}\right)\right) \tag{9}$$

Dimana :
j adalah jumlah PSN dari 1 sampai *N*,
i adalah banyaknya order di PSN dari *i* sampai *j*,
k adalah jumlah input dari 1 sampai *n*,
w_{ijk} adalah bobot yang diperbarui dari input x_k ke unit PSN order ke-*i* dari PSN ke-*j*.
$f(x)$ adalah fungsi aktivasi nonlinear.
Total jumlah bobot yang terlibat dalam struktur adalah $N(N+1)(n+1)/2$.

Algortima pelatihan jaringan ridge polynomial pada dasarnya terdiri dari 5 tahapan yaitu:

1. Mulai dengan RPNN order 1, yang mana memilki satu unit PSN order pertama
2. Melakukan pelatihan dan update bobot secara asinkronus setelah setiap pola pelatihan.
3. Ketika error dari PSN yang diamati berubah jatuh dibawah error standart r, yaitu $\left|\frac{e_c - e_p}{e_p}\right|$, maka PSN order lebih tinggi ditambahkan. Catatan bahwa adalah MSE untuk iterasi saat ini dan adalah MSE untuk iterasi sebelumnya.
4. Error target r dan learning rate n dibagi dengan faktor dec_r dan dec_n.
5. Jaringan diperbarui melakukan siklus pembejaran (ulangi langkah 2 sampai 4) sampai jumlah unit PSN yang diinginkan tercapai atau jumlah iterasi maksimum tercapai.

2.5Proses pengujian

Proses tersebut digunakan untuk menguji sistem untuk mendapatkan arsitektur terbaik dengan memakai hasil bobot dan bias dari hasil pelatihan. Arsitektur terbaik didasarkan pada kriteria informasi pada nilai MSE.
Fungsi objektif yang digunakan yaitu :

$$\text{Mean Square Error (MSE)} = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2 \quad (10)$$

dimana*i* merupakan indeks pola data dari pelatihan, t_i adalah target dari jaringan untuk pola ke-*i* dan y_i adalah output aktual dari jaringan untuk pola ke-*i*.

2.5 Proses denormalisasi

Proses denormalisasi adalah hasil output jaringan dari proses pengujian berupa range nilai antara 0 sampai 1 akan di ubah kedalam angka sebenarnya.

3. HASIL DAN PEMBAHASAN

3.1 Pelatihan Jaringan Syaraf tiruan

Proses pelatihan dengan melakukan 7 kali percobaan. Dimana Setiap kali percobaan dengan arsitektur tertentu, learning rate ditentukan antara 0,1 sampai 0.9. Kesimpulan hasil pelatihan dari percobaan 1 sampai 7 dapat dilihat pada Tabel 1. Dari Tabel 1 didapat hasil MSE terkecil pada percobaan ke-2, yaitu: Learning rate = 0.3, arsitektur = 6-2-1, maksimum iterasi = 451, MSE = 0.021954.

Tabel 1 Kesimpulan hasil pelatihan dari percobaan 1 sampai 7

Percobaan	Arsitektur	Learning rate	Iterasi ke-	MSE
1	6-1-1	0.3	393	0.022099
2	6-2-1	0.3	451	0.021954
3	6-3-1	0.4	519	0.022058
4	6-4-1	0.4	1020	0.022045
5	6-5-1	0.2	2198	0.022096
6	6-6-1	0.1	3155	0.022093
7	6-7-1	0.3	4999	0.022119

Pengaruh parameter yang dipelajari dalam penelitian ini dikelompokkan menjadi 2 bagian yaitu:

3.1.1 Pengaruh parameter interval pencarian

Interval pencarian mentukan proses pencarian solusi yang optimal. Solusi tersebut mampu mencapai optimal global, dengan persamaan variabel chaos yaitu logistik map, yang memiliki sensitif terhadap kondisi awal. Jika nilai awal sudah ditentukan pada hasil perhitungan pertama, tidak akan pernah sama dengan nilai awal pada hasil perhitungan kedua yang hampir mendekati nilai awal pertama. Sehingga ketika menggunakan persamaan logistik map, nilai tersebut mampu mencari nilai optimal tanpa ada perulangan nilai. Pada pembahasan ini akan diamati pengaruh interval pencarian pada sistem dengan memperluas atau menyempit interval pencarian.

Tabel 2 Pengaruh parameter interval pencarian pada pencarian pertama

Pencarian pertama	Pencarian kedua	Iterasi ke-	MSE
[-100, 100]	[-2, 2]	445	0.022138
[-90, 90]	[-2, 2]	580	0.022142
[-80, 80]	[-2, 2]	427	0.022011
[-70, 70]	[-2, 2]	452	0.022269
[-60, 60]	[-2, 2]	478	0.022129
[-50, 50]	[-2, 2]	451	0.022037
[-40, 40]	[-2, 2]	408	0.022189
[-30, 30]	[-2, 2]	420	0.022101
[-20, 20]	[-2, 2]	415	0.022029
[-10, 10]	[-2, 2]	383	0.022145

Tabel 3 Pengaruh parameter interval pencarian pada pencarian kedua

Pencarian pertama	Pencarian kedua	Iterasi ke-	MSE
[-80, 80]	[-1,1]	563	0.022100
[-80, 80]	[-2, 2]	465	0.022084
[-80, 80]	[-3, 3]	423	0.022201
[-80, 80]	[-4, 4]	416	0.022013
[-80, 80]	[-5, 5]	428	0.022097
[-80, 80]	[-6, 6]	604	0.022123
[-80, 80]	[-7, 7]	463	0.022159
[-80, 80]	[-8, 8]	442	0.021979
[-80, 80]	[-9, 9]	519	0.022079
[-80, 80]	[-10, 10]	413	0.022024

3.1.2 Pengaruh parameter maksimum iterasi

Jumlah iterasi akan menentukan proses pencarian solusi lebih baik. Lebih banyak jumlah iterasi, proses pencarian lebih mungkin untuk mendapatkan solusi yang diinginkan. Algoritma optimasi chaos bekerja sesuai pada percobaan sebelumnya, selanjutnya untuk jumlah iterasi ditentukan oleh pengguna. Pada pembahasan ini akan diamati pengaruh jumlah itrasi terhadap nilai MSE yang terkecil.

Tabel 4 Pengaruh Parameter maksimum iterasi pada pencarian pertama

Pencarian pertama	Pencarian kedua	Iterasi ke-	MSE
1000	10.000	428	0.022324
2000	10.000	440	0.022640
3000	10.000	437	0.022104
4000	10.000	423	0.022069
5000	10.000	406	0.022188
6000	10.000	382	0.022650
7000	10.000	435	0.022253
8000	10.000	493	0.022141
9000	10.000	450	0.021943
10.000	10.000	436	0.021844

Tabel 5 Pengaruh parameter maksimum iterasi pada pencarian kedua

Pencarian pertama	Pencarian kedua	Iterasi ke-	MSE
10.000	1.000	420	0.022033

10.000	2.000	443	0.022093
10.000	3.000	409	0.022125
10.000	4.000	394	0.022186
10.000	5.000	586	0.022134
10.000	6.000	461	0.022094
10.000	7.000	425	0.022196
10.000	8.000	419	0.022462
10.000	9.000	523	0.022035
10.000	10.000	448	0.022112

3.2 Pengujian Jaringan Syaraf Tiruan

Untuk dapat melihat jaringan syaraf tiruan yang telah optimal, untuk dapat melihat hasilnya, dilakukan perbandingan hasil pengujian dan pelatihan jaringan syaraf tiruan. Perbandingan hasil pengujian dengan pelatihan dapat dilihat pada Tabel 6

Tabel 6Perbandingan hasil pengujian dengan pelatihan JST

Pembahasan	MSE
Pelatihan jaringan syaraf tiruan	0.021954
Pengujian jaringan syaraf tiruan	0.429706

3.2.1 Perbandingan antara jaringan ridge polynomial dan algoritma optimasi chaos dengan jaringan ridge polynomial saja.

Parameter pengujian jaringan ridge polynomial dan algoritma optimasi chaos sama dengan proses pelatihan diatas. Parameter di jaringan ridge polynomial yaitu learning rate dimana ketika setiap kali PSN ditambahkan learning rate menurun dengan pembagi 1.7, error pada PSN awal = 0.0001 ketika setiap kali PSN ditambahkan menurun dengan pembagi 10. Learning rate awal = 0.04, arsitektur =6-2-1, maksimum iterasi = 5000, error target = 0.0001. Hasil yang didapat ditampilkan dengan Tabel 7

Tabel 7 Perbandingan antara jaringan ridge polynomial dan algoritma optimasi chaos dengan jaringan ridge polynomial saja

Metode	Pelatihan	Pengujian	Output	Selisih
Jaringan ridge polynomial dan algoritma optimasi chaos	0.021954	0.429706	119353	6853
Jaringan ridge polynomial	0.022120	0.427137	119348	6848

3.2.2 Perbandingan antara jaringan ridge polynomial dan algoritma optimasi chaos dengan backpropagation

Parameter pengujian jaringan ridge polynomial dan algoritma optimasi chaos sama dengan proses pelatihan diatas. Parameter di Backpropagation yaitu leraning rate = 0.04, jumlah neuron hidden = 5, maksimum iterasi = 5000, error target = 0.0001. Hasil yang didapat ditampilkan dengan Tabel 8

Tabel 8 Perbandingan antara jaringan ridge polynomial dan algoritma optimasi chaos dengan backpropagation

Metode	Pelatihan	Pengujian	Target data	Output	Selisih
Jaringan ridge polynomial dan algoritma optimasi chaos	0.021954	0.429706	112500	119353	6853
Backpropagation	0.041854	0.540908	112500	116987	4487

4. KESIMPULAN

Kesimpulan yang diperoleh dalam penelitian ini adalah.

1. Jaringan ridge polynomial memiliki kemampuan yang baik dalam memprediksi jumlah pengangguran, menggunakan arsitektur unit neuron jaringan pi-sigma 2.
2. Penggunaan interval pencarian pertama dan pencarian kedua pada algoritma optimasi chaos tidak mempunyai pengaruh yang signifikan pada proses pelatihan.

3. Penggunaan maksimum iterasi pencarian pertama dan interval pencarian kedua pada algoritma optimasi chaos tidak mempunyai pengaruh yang signifikan pada proses pelatihan.

Dari hasil pengujian dengan metode lain diperoleh:

1. Perbandingan hasil pengujian jaringan ridge polynomial dan algoritma optimasi chaos dengan pengujian jaringan ridge polynomial saja tidak berbeda jauh nilai MSE nya.
2. Perbandingan hasil pengujian jaringan ridge polynomial dan algoritma optimasi chaos lebih baik daripada pengujian backpropagation terhadap nilai MSE nya.

5. SARAN

Adapun saran-saran terhadap penelitian ini adalah.

1. Metode jaringan syaraf tiruan dan algoritma optimasi chaos ini lebih baik digunakan untuk studi kasus yang membutuhkan data lebih banyak serta dapat ditambahkan data variabel yang dapat mempengaruhi jumlah pengangguran untuk mendukung proses memprediksi jumlah pengangguran.
2. Pada penelitian lebih lanjut diharapkan melakukan riset untuk optimasi fungsi nonlinear pada algoritma optimasi chaos disarankan menggunakan metode Broyden, Fletcher, Goldfarb dan Shanno (BFGS).
3. Dan lebih lanjut diharapkan melakukan riset pada algoritma optimasi bobot yang lain pada Ridge Polynomial Neural Network seperti Bacterial Chemotaxis Optimization (BCO), Differential Evolution Algorithm atau Particle Swarm Optimization (PSO).

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada kedua orang tua yang telah memberi dukungan doa dan financial terhadap penelitian ini.

DAFTAR PUSTAKA

[1]Hacib, T., Bihan, Y. L., Mekideche, M. R., Ferkha, N., 2010, *Ridge Polynomial Neural Network for Non-destructive Eddy Current Evaluation, Computational Methods for the Innovative Design of Electrical Devices, (Studies in Computational Intelligence 327)*, Springer, Verlag Berlin Heidelberg.

[2]Zhang, R., Zheng, X., 2010, A New Flatness Pattern Recognition Model Based on Variable Metric Chaos Optimization Neural Network, *R. Zhu et al. (Eds.): ICICA 2010, LNCS 6377, pp. 357-364, Springer-Verlag Berlin Heidelberg 2010.*

[3]Karnavas, Y. L., Papadopoulos, D. P., 2004, Excitation Control of a Synchronous Machine Using Polynomial Neural Networks, *Journal of ELECTRICAL ENGINEERING, VOL. 55, NO. 7-8. 2004, 169-179, ISSN 1335-3632.*

[4]Velasquez, J. D., 2010, An Enhanced Hybrid Chaotic Algorithm using Cyclic Coordinate Search and Gradient Techniques,*revista de ingenieria. Universidad de los Andes. Bogota, Colombia. rev.ing. ISSN. 0121-4993. Julio – Diciembre de 2010, pp.45-53.*

[5]Khoa T. Q. D. dan Nakagawa M., 2007, Neural Network Learning based on Chaos, *International Journal of Computer and Information Engineering 1:2 2007*

[6]Samarasinghe, S., 2007, *Neural Networks for Applied Sciences and Engineering, From Fundamentals to Complex Pattern Recognition,* Auerbach Publicationss, New York.

[7]Epitropakis, M. G., Vrahatis, M. N., 2005, Root finding and approximation approaches through neural networks, *ACM SIGSAM Bulletin, Vol 39, No. 4, Desember 2005.*

[8]Gupta, M. M., Jin, L., Homma, N., 2003, *Static and Dynamic Neural Networks, From Fundamentals to Advanced Theory,* John Wiley & Sons, Inc., Hoboken, New Jersey