PEMBELAJARAN JARINGAN SYARAF TIRUAN PROBABILISTIK BASIS RADIAL DENGAN MENGGUNAKAN ANALISIS SENSITIVITAS

Hasanuddin
Program Studi Pendidikan Matematika
Universitas Islam Negeri Sultan Syarif Kasim Riau, 28293
hasanuddin.ftk@uin-suska.ac.id

Abstrak

Makalah ini menyajikan algoritma pembelajaran bagi Radial Basis Probabilistic Neural Network berdasarkan keturunan gradien fungsi kesalahan. RBPNN terintegrasi dari jaringan saraf radial fungsi dasar (RBFNN) dan jaringan syaraf probabilistik (PNN). Langkah Kepekaan terhadap masukan atas dataset pelatihan berdasarkan turunan parsial dari model RBPNN dan aturan untuk memilih fitur berlebihan juga hadir. Analisis sensitivitas metode yang dapat meningkatkan efisiensi dan efektivitas jaringan saraf. Akhirnya, untuk mengevaluasi kinerja, pendekatan yang diusulkan kami menunjukkan melalui memberikan dua contoh kehidupan nyata kumpulan data.

Kata Kunci*: RBPNN, keturunan gradien, analisis sensitivitas, seleksi fitur, klasifikasi.*

Abstract

This paper presents a learning algorithm for Radial Basis Probabilistic Neural Network based on gradient descent of error functions. RBPNN integrates of radial basis function neural networks (RBFNN) and probabilistic neural networks (PNN). Sensitivity measures to input over training dataset based on partial derivative of the RBPNN model and rule for selecting redundant feature is also present. Sensitivity analysis of that method can improved efficiency and effectiveness of the neural networks. Finally, to evaluate the performance, our proposed approaches are demonstrated trough giving two examples of real life data set.

Keyword: *RBPNN, gradient descent, sensitivity analysis, feature selection, classification.*

1. Pendahuluan

Model jaringan syaraf tiruan *Radial Basis Probabilistic* (RBPNN), seperti yang dapat dilihat pada Gambar 1. Mengintegrasikan kelebihan dari dua jaringan syaraf tiruan yaitu Jaringan Syaraf Fungsi Basis Radial (RBFNN) dan Probabilistic Neural Networks (PNN). Konstruksi RBPNN terdiri dari empat lapisan yang berbeda: satu lapisan input, dua lapisan tersembunyi dan satu lapisan output. Lapisan tersembunyi pertama adalah lapisan pemroses non linier. Secara umum terdiri dari center-center tersembunyi yang ditentukan berdasarkan dengan melatih data input. Lapisan tersembunyi kedua secara selektif menjumlahkan keluaran dari lapisan tersembunyi pertama dan memiliki ukuran yang sama

dengan lapisan output yang ditujukan untuk masalah klasifikasi. Bobot antara lapisan tersembunyi pertama dan kedua adalah konstanta [6].

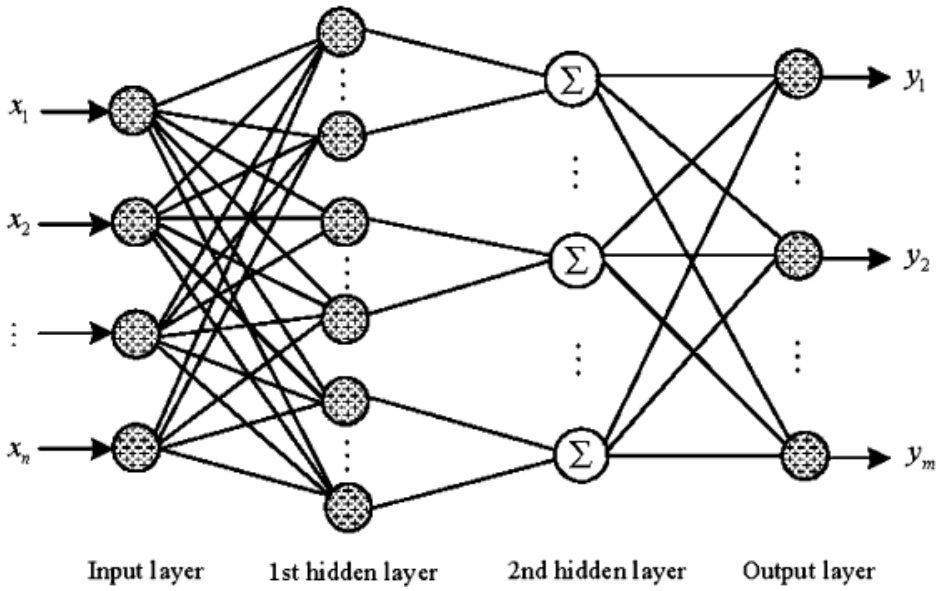
Kinerja RBPNN tergantung pada jumlah dan posisi center, spread, dan metode yang digunakan untuk pembelajaran pemetaan input-output. Strategi pembelajaran untuk RBPNN dapat diklasifikasi sebagai berikut: 1) strategi untuk pemilihan secara random dari data training 2) Strategi tanpa supervisi untuk pemilihan center 3) Strategi prosedur tersupervisi untuk pemilihihan center [7]. Ada banyak metode pembelajaran alternative untuk jaringan syaraf. Sebagai contoh, pada kasus jaringan dengan multi lapis algoritma sukses pertama yaitu Backpropogasi klasik[1]. Sejumlah periset telah menawarkan penyelesaian maslah simplikasi jaringan, seperti meningkatkan algoritma pembelajaran, sesnsitivitas berdasarkan pemilihan fitur dan sensitivitas analisis dari center-center neuron tersembunyi. Sensitivitas berdasarkan simplifikasi jaringan salah satu teknik yang cukup efektif dan lebih diutamakan[9].

Analisis sensitivitas fitur merupakan isu fundamental dalam desain jaringan syaraf tiruan. Hal ini dapat digunakan untuk pemilihan fitur, penyederhanaan jaringan syaraf dan meeningkatkan kinerja secara umum[9].

Makalah ini disusun sebagai berikut. Bagian 2 menyajikan algoritma pembelajaran tersupervisi untuk RBPNN berdasarkan gradient descent. Bagian 3 menyajikan ukuran sensitivitas untuk input terhadap data pelatihan dan aturan pemilihan fitur redundan. Bagian 4 evaluasi kinerja RBPNN dan pengujian berdasarkan algoritma yang ditawarkan dan bagian 5 terdiri dari kesimpulan.

1. Metode *Gradient Descent* untuk Pembelajaran Bobot

Model Jaringan Syaraf Probabilistik Basis Radial atau *Radial Basis Probabilistic Neural Network* (RBPNN) seperti yang dapat dilihat pada Gambar [1] menggabungkan beberapa keunggulan RBFNN dan PNN.



Gambar 2.1 Arsitektur RBPNN [6]

Konstruksi RBPNN terdiri dari empat lapisan, yaitu: lapisan input, dua lapisan tersembunyi dan lapisan output. Lapisan tersembunyi pertama merupakan lapisan dengan proses tak linier, yang secara umum terdiri dari *center* tersembunyi yang ditentukan dengan himpunan sampel pelatihan input. Lapisan tersembunyi

kedua merupakan penjumlahan dari output lapisan pertama, dan secara umum memiliki ukuran yang sama dengan lapisan output.

Bobot antara lapisan tersembunyi pertama dan lapisan tersembunyi kedua, merupakan bobot konstan. Artinya, bobot disetting tetap sehingga tidak diperlukan pembelajaran (Huang dkk., 2005).

Misalkan untuk vektor input ***x***, nilai aktual dari neuron output RBPNN ke-*i* y_i^a, secara matematika dapat disederhanakan menjadi persamaan berikut

$$y_i^a = \sum_{k=1}^{M} w_{ik} h_k(x),\ i = 1, 2, ..., M \tag{1}$$

$$h_k(x) = \sum_{i=1}^{n_k} \phi_i(x, c_{ki}) = \sum_{i=1}^{n_k} \phi_i\left(\|x - c_{ki}\|_2\right),\ k = 1, ..., M, \tag{2}$$

dengan:

$h_k(x)$: nilai output ke-*k* dari lapisan tersembunyi RBPNN,
w_{ik} : bobot sinaptik antara lapisan tersembunyi kedua dan lapisan output RBPNN,
c_{ki} : vektor center tersembunyi untuk kelas pola ke *k*,
n_k : jumlah vektor center tersembunyi untuk kelas pola ke *k*,
$\|\cdot\|_2$: norm euclid,
M : jumlah neuron pada lapisan output,
$\phi_i(\cdot)$: fungsi kernel, biasanya menggunakan fungsi gauss.

Sebelum menentukan bobot RBPNN, terlebih dahului harus ditentukan kernel Gaussnya. Beberapa algoritma Clustering dapat digunakan untuk menentukan vektor-vektor pusat. Cara termudah yaitu dengan memilih secara random dari data pelatihan. Akan tetapi, metode ini haruslah menggunakan center yang banyak, tujuannya untuk mencover seluru domain pola input. Salah satu metode yang cukup populer yaitu *K-means Clustering*, metode ini telah diterima secara luas karena kemampuannya untuk memberikan hasil yang cukup baik. Ide dasar dari algoritma ini yaitu mengumpulkan data ke dalam beberapa kelompok dan memilih *center* berdasarkan ukuran dari *center-center* dengan menggunakan jarak Euclid

Salah satu parameter penting lainnya dari RBPNN yiatu parameter spread dari fungsi kernel gauss. *Spread* umum dapat dipilih dengan menggunakan rata-rata dari semua jarak Euclid antara *center* ke-*i* dengan lingkungan terdekatnya. Parameter-parameter spread dari fungsi-fungsi kernel RBPNN dapat dilihat persamaan berikut[2]

$$\sigma = \frac{d_{\max}}{\sqrt{K}}, \tag{3}$$

dengan:

$d_{\max}$: Jarak Euclid maksimal dari sehimpunan pelatihan,
K : Jumlah total sehimpunan pelatihan.

RBPNN dilatih untuk memetakan vector $\mathbf{x}(l) \in R^N$ ke vector $\mathbf{y}(l) \in R^M$, dengan pasangan terurut $(\mathbf{x}(l), \mathbf{y}(l))\ 1 \le l \le L$ berasala dari data pelatihan. RBPNN candilatih dengan memiminasi error

$$E = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{M}\left(e_{i,l}\right)^2, \tag{4}$$

Dengan $e_{i,l}$ merupakan error approxsimasi dari neuron output ke-*i*dan sampel pelatihan ke-*l*. $e_{i,l} = y_{i,l} - \hat{y}_{i,l}$.

Persamaan (4) digunakan untuk menentukan bobot optimal. Bobot pelatihan kemudian ditentukan dengan menurunkan persamaan (4) terhadap bobot w_p sebagai berikut

$$\frac{\partial E}{\partial \mathbf{w}_p} = -\sum_{l=1}^{L}\sum_{i=1}^{M}\left(y_{i,l} - \hat{y}_{i,l}\right)\cdot\frac{\partial\left(\hat{y}_{i,l}\right)}{\partial \mathbf{w}_p}, \tag{5}$$

dengan

$$\frac{\partial\left(\hat{y}_{i,l}\right)}{\partial \mathbf{w}_p} = \mathbf{h}_l\delta_{ip} \tag{6}$$

δ_{ip} adalah Kronecker delta, jika $i = p$ maka $\delta_{ip} = 1$, $\delta_{ip} = 0$ untuk yang lainnya.

Selanjutnya subtitusi persamaan (6) ke dalam persamaan (5) diperoleh

$$\frac{\partial E}{\partial \mathbf{w}_p} = -\sum_{l=1}^{L}\varepsilon_{p,l}^{o}\mathbf{h}_l \tag{7}$$

dengan $\varepsilon_{p,l}^{o} = y_{p,l} - \hat{y}_{p,l}$.

Jadi, persamaan untuk vector-vektor bobotjaringan dapat diperoleh dengan menggunakan gradient descent

$$\begin{aligned}\Delta w_p &= -\eta\frac{\partial E}{\partial \mathbf{w}_p}\\ &= \eta\sum_{l=1}^{L}\varepsilon_{p,l}^{o}\mathbf{h}_l\end{aligned} \tag{8}$$

dengan η is *tingkat pembelajaran* and $\varepsilon_{p,l}^{o}$ is eror output.

Algoritma ini dapat disimpulkan sebagai berikut:

1) pilih η dan ε; inisialisasi $\{w_{ij}\}$ dengan nilai nol;
2) tentukanpusat berdasarkan *K-Mean algorithm*
3) tentukan *spread* dengan menggunakan persamaan (3)
4) Hitung Respon awal
   * $h_k(\mathbf{x}) = \sum_{i=1}^{n_k}\phi_i(\mathbf{x}, \mathbf{c}_{ki}) = \sum_{i=1}^{n_k}\phi_i\left(\left\|\mathbf{x} - \mathbf{c}_{ki}\right\|_2\right)$
   * $\hat{y}_i = \sum_{k=1}^{M} w_{ik}h_k(\mathbf{x})$
5) Hitung $E = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{M}\left(y_{i,l} - \hat{y}_{i,l}\right)^2$
6) Set *Eold* =*E*;
7) Update bobot
   * $\varepsilon_{p,l}^{o} = y_{p,l} - \hat{y}_{p,l}$

* $\mathbf{w} = \mathbf{w} + \eta \sum_{l=1}^{L} \varepsilon_{p,l}^{o} \mathbf{h}_l$

8) Hitung Respon langsung

* $h_k(\mathbf{x}) = \sum_{i=1}^{n_k} \phi_i(\mathbf{x}, \mathbf{c}_{ki}) = \sum_{i=1}^{n_k} \phi_i\left(\|\mathbf{x} - \mathbf{c}_{ki}\|_2\right)$

* $\hat{y}_i = \sum_{k=1}^{M} w_{ik} h_k(\mathbf{x})$

9) Hitung $E = \frac{1}{2} \sum_{l=1}^{L} \sum_{i=1}^{M} \left(y_{i,l} - \hat{y}_{i,l}\right)^2$

10) jika (*Eold-E*)> ε; maka, kembali ke langkah (5) lainnya berhenti.

2. Analisis Sensitivitas

Sensitivitas respon fungsi *f* yang mengacu kepada parameter desain himpunan *N* dapat ditentukan dari gradiennya. Misalakan $f(x_1, x_2, \cdots, x_N)$ fungsi yang dapat didifferensialkan dengan N variabel. Seperti yang diasumsikan pada [9], jaringan dapat dipandang sebagai pemetaan taklinier. Sensitivitasnya pada titik input dapat dipandang sebagai vektor

$$\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_N} \right), \tag{9}$$

Sensitivitas jaringan pada himpunan data pelatihan yang dihasilkan pada pada (9) dapat dipertimbangkan sebagai koefisien sensitivitas. Khususnya, sensitivitas y_m yang mengacu ke x_n sebagai berikut

$$s_{x_n}^{y_m} = \frac{\partial y_m}{\partial x_n}, \tag{10}$$

Yang kemudian dapat ditulis

$s_{mn} = s_{x_n}^{y_m}$,

2.1 Pengukuran sensitivitas untuk input pada data pelatihan

Definisi 1 [10]. *Sensitivitas mean square average (MSA), S_{mn} terhadap himpunan χ dapat dihitung sebagai berikut*

$$S_{mn} = \sqrt{\frac{\sum_{l=1}^{L} \left(S_{mn}^{(l)}\right)^2}{N}}, \tag{11}$$

dengan $s_{mn}^{(l)}$ merupakan sensitivitas yang mengacu vector input ke -l.

Andaikan RBPNN memiliki *M* output, maka vector output dari jaringan dapat dituliskan $(y_1, y_2, \cdots, y_M)$. sehingga, setiapoutput dapat dihitung dengan

$$y_i = \sum_{k=1}^{M} w_{ik} \sum_{i_k=1}^{n_k} \phi_k\left(\|\mathbf{x} - \mathbf{c}_{ki_k}\|_2\right),\ i = 1,2,\cdots,M, \tag{12}$$

Turunan parsial dari jaringan yang mengacu kepada input x_n, dapat diperoleh dengan

$$s_{x_n}^{y_i} = \sum_{k=1}^{M} w_{ik} \sum_{i_k=1}^{n_k} \exp\left(\frac{\sum_{t=1}^{N} \left(x_t - c_{i_k t}\right)^2}{-2\sigma_{i_k}^2} \right) \left(\frac{x_n - c_{i_k n}}{-\sigma_{i_k}^2} \right), \quad (13)$$

dengan $n = 1, 2, ..., N$.

Formula sensitivitas pada persamaan (13) dapat digunakan untuk menghitung sensitivitas fitur input ke-*n* terhadap output ke-m.

3.2 Aturan pemilihan Fitur redundan

Matriks sensitivitas ***S*** dapat digunakan untuk mendeteksi fitur redundan. Jika sensitivitas satu atau lebih fitur input relative kecil, dimensi input dari RBPNN dapat disederhanakan.

$$\hat{x}_n^{(l)} = \frac{x_n^{(l)} - \left(\left(\max_{n=1,...,N} \left\{ x_i^{(n)} \right\} + \min_{n=1,...,N} \left\{ x_i^{(n)} \right\} \right) \Big/ 2 \right)}{\left(\max_{n=1,...,N} \left\{ x_i^{(n)} \right\} - \min_{n=1,...,N} \left\{ x_i^{(n)} \right\} \right)}, \quad (14)$$

dengan^ melambangkan variable ternormalisasi, or parameter.

Jika penskalaan terhadap input (14) telah dilakukan sebelum melatih jaringan, tidak perlu lagi operasi tambahan pada $s_{mn,\,\mathrm{is}}$ dan berarti

$$\hat{s}_{mn} = s_{mn}, \quad (15)$$

Untuk mengukur sensitivitas jarignan yang mengacu kepada input menggunakan bilangan riil bukan vector. Pengukuran dilakukan dengan menggunakan

$$S_n = \max_{m=1,...,M} \{\hat{s}_{nm}\}, n = 1, \cdots, N, \quad (16)$$

Dengan $\hat{s}_{nm}$ sensitivitas yang mengacu ke input ke-*n dan output ke-m.*

Ketika pembelajaran jaringan selesai, dipih criteria untuk menyederhanakan input yang mirip dengan dengan criteria pada [9] dan [10]. Sensitivitas dapat dihitung dengan persamaan (16)lalu keemudia di sorit dengan criteria sebagai berikut

$$S_{n_{u+1}} \le S_{n_u}, u = 1,..., N-1, \quad (17)$$

dengan n_u barisan input yang disortir.

Gap sensitivitas ditentukan berdasarkan perbandingan dua sensitivitas yang berdekatan, sebagai berikut

$$g_u = \frac{S_{n_u}}{S_{n_{u+1}}}, u = 1,..., N-1, \quad (18)$$

Kemudian Gap terbesar dapat ditentukan dengan

$$g_{\max} = \max_u \{g_u\}, \; u = 1,2,..., N-1, \quad (19)$$

Indeks $g_{\max}$ dapat digunakan untuk menentukan fitur-fitur efektif yang dapat member pengaruh yang signifikan ke RBPNN.

$$u_{\mathrm{cut}} = u \quad (20)$$

gap terbesar kedua dapat ditentukan dengan menggunakan criteria berikut

$$g_{\max\, II} = \max_{u \ne u_{cut}} \{g_u\}, \; u = 1,2,..., N-1 \quad (21)$$

jika $g_{\max}$ lebih besar dibanding $g_{\max\, II}$, i.e., $C\, g_{\max} > g_{\max\, II}$, dengan C = 0.5, maka fitur input yang sensitiflebih kecil $S_{n_{u_{cut}}}$ dapat di sederhanakan.

3. Eksperimen dan Hasil

Data penelitian yang kami sajikan pada bab ini telah digunakan untuk mengevaluasi kinerja jaringan RBPNN. Penelitian difokuskan pada tingkat akurasi klasifikasi dan analisis sensitivitas fitur input RBPNN. Data yang digunakan sebanyak tiga dataset, yaitu: *Iris* dan E-Coli. Ketiga data ini diperoleh dari website UCI Machine Learning.

3.1 Iris Dataset

Kinerja RBPNN-GD degenerate berdasarkan pada pendekatan yang dibuat dan juga telah dibandingkan dengan algoritma RBPNN konvensional dengan menggunakan data Iris. Data Iris telah digunakan secara luas untuk evaluasi kinerja algoritma clustering dan klasifikasi. Data iris terdiri dari 150 titik data, empat atribut yang terdiri dari 3 kelas. Setiap kelas terdiri dari 50 titik data. Ketida kelas tersebut dibagi secara random menjadi 2 bagian, 70% digunakakan untuk pembelajaran dan 30% digunakan untuk pengujian. RBPNN diuji pada terdiri dari empat input dan tiga output linier. Output akan terkait dengan kelas secara fisik.

Kunci utama optimisasi RBPNN yaitu menyeleksi hidden center untuk lapisan tersembunyi pertama. Beberapa algoritma clustering dapat digunakan untuk langkat lebih lanjut untuk menentukan center dari RBPNN. Center merepresentasikan lokasi kernel RBF pada ruang input yang ditentukan dengan *K-means algorithms*. Parameter penting dari RBPNN ditentukan berdasarkan persamaan (3). RBPNN dilatih menggunakan algoritma yang dikaji dengan rate pembelajaran $\eta = 10^{-3}$.

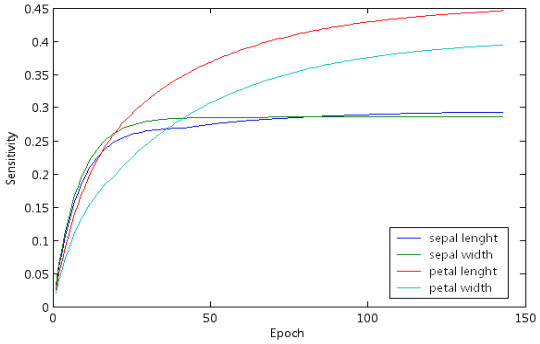
Table 1 shows the percentage of classification errors produced on the training and testing dataset formed from the Iris data by RBPNN-LS [5], RBPNN-OLS [6] and RBPNN-GD. According to the Table 1, each RBPNN algorithm produced almost the same numbers of classification errors on the training set and testing set.

Pada Tabel 1 dapat dilihat persentase eror klasifikasi yang dihasilkan pelatihan dan pengujian data yang didapat dari Data Iris berdasarkan RBPNN-LS [5], RBPNN-OLS [6] and RBPNN-GD. Berdasarkan Tabel 1, setiap algoritma yg dihasilkan hamper memiliki kemampuan yang sama dalam hal error baik pada data pelatihan maupun pengujian.

Tabel 4.1 Percentage of Classification Errors produced on The Training Dataset and Testing Dataset Formed from the Iris Data.

Data	**Classification Errors**		
	RBPNN-LS	**RBPNN-OLS**	**RBPNN-GD**
Training	2.86%	2.86%	2.86%
Testing	2.22%	2.22%	2.22%

Perubahan signifikansi input selama pembelajaran dapat dilihat pada gambar 2. Sensitivitas pada awalnya sangat rendah. Selama pembelajaran sensitivitasnya semakin meningkat.



Gambar 4.1 Input Significance Changes during Training for the Training Dataset.

Table 4.2 Feature Sensitivities using sensitivity analysis

Feature	Sensitivity	Gap
3	0.44636	*1.131917*
4	0.39434	**1.34399**
1	0.29341	1.023904
2	0.28656	0
g_{max}	1.34399	
g_{max} ii	1.131917	
g_{max} ii/g_{max}	0.842206	>C

Nilai akhir sensitivitas dapat dilihat pada Tabel 2. Parameter C pada eksperimen ini di set untuk C=0.5. Bilangan yang ditebalkan adalah Gap terbesar dan Bilangan yang dimiringkan merupakan gap terbesar kedua. Selanjutnya diperoleh g_{max} ii/g_{max} = 0.842206 > C. jadi tidak ada fitu yang boleh di hapus.

4. E-Coli

Kinerja RBPNN-GD didasarkan kepada lagoritma pendekatan yang diajukan juga uji dengan menggunakan Data E-Coli. Data ini terditi dari 336 titik data dan yang terbagi kedalam 8 kelas.

Centernya merepresentasikan lokasi kernel RBF pada ruang input yang ditentukan dengan menggunakan *K-means algorithms*. Parameter penting dari RBPNN yaitu spread ditentukan dengan menggunakan persamaan (3). Learning ratenya η = 0.1.

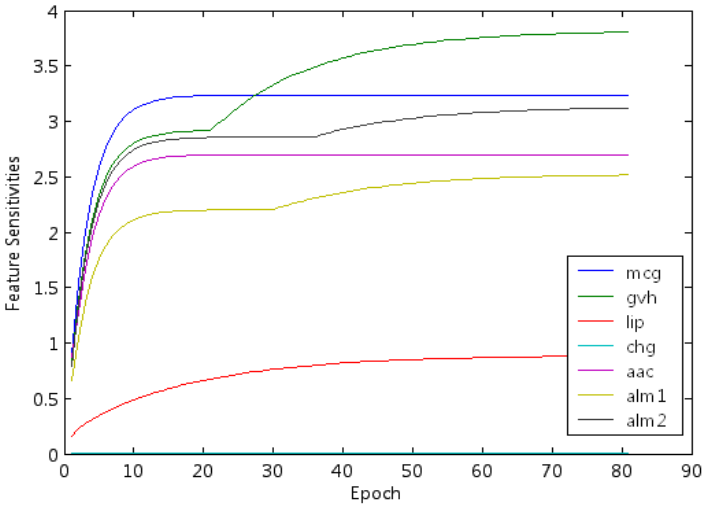
Pada Tabel 3 dapat dilihat persentase error klasifikasi yang dihasilkan dengan menggunakan Data E-Coli. Berdasarkan Tabel 3, RBPNN yang dilatih dengan menggunakan Gradient Descent menampilkan eror yang lebih kecil jika dibandingkan dengan yang lainnya.

Tabel 5.1 Percentage of Classification Errors produced on the Training Dataset and Testing Dataset Formed from the E-coli Data.

Data	Classification Errors		
	RBPNN-LS	RBPNN-OLS	RBPNN-GD
Training	17,21%	17,21%	15,57%
Testing	20,83%	20,83%	19,79%

Signifikansi input selama pelatihan data E-Coli dapat ditunjukkan pada gambar 3. Selama pelatihan beberapa sensifitas meningkat, ketika yang lain konvergen

menuju nilai terendah. Nilai akhir sensitivitas dari output memberikan petunjuk untuk penghapusan input fitur yang memiliki sensitivitas rendah.



Gambar 5.1 Input Significance during training for the E-Coli dataset

Tabel 5.2 Feature sensitivities for the E-Coli Dataset

Feature	Sensitivity	Gap
2	3.8052	1.175460274
1	3.2372	1.037431099
7	3.1204	1.155062003
5	2.7015	1.073770818
6	2.5159	*2.846040724*
3	0.884	**27977339620.85**
4	3.16×10^{-11}	0
g_{max}		27977339620.85
g_{max} ii		2.846040724
g_{max}/g_{max} ii	1.01727×10^{-10}	$< C = 0.5$

Tabel 4 merupakan simpulan numeris.. bilangan yang di tebalkan merupakan gap terbesar bilangan yang dimiringkan gap terbesar kedua. Selanjuntya berdasarkan perhitungan diperoleh g_{max} ii/g_{max} = $1.01727 \times 10^{-10} < C$. Parameter C pada eksperiment ini di setuntuk nilai C = 0.5. semua fitur yang berada dibawah fitur input yang memiliki level yang sama dengan gap terbesar dapat dihapus. Sensitivitas x_4 atau fitur *chg* sangat rendah is 3.16×10^{-11}.

Tabel 5 .3 Percentage of Classification Errors before and after Pruning Feature of the E-coli dataset

Data	Before Pruning	After Pruning
Training	15,57%	15,57%
Testing	19,79%	19,79%

Pada Tabel 5. Dapat dilihat persentase error klasifikasi yang hasilkan selama pelatihan dan pengujian. Hasil eksperimen menunjukkan bahwa eror klasifikasi tidak bertambah setelah dilakukan penghapusan atribut yg memiliki sesnitifitas yang rendah.

5. Kesimpulan

Bobot konvensional RBPNN dapat diperoleh dengan menggunakan matriks pseudo invers atau dekomposisi QR. Makalah ini menawarkan alternatif pendekatan untuk melatih RBPNN yaitu dengan gradient descent. Hasil experiment menunjukkan hasil yang lebih baik atau sama dengan algoritma yang lain. Analisis sensitivitas dari input merupakan suatu langkah yang efisien untuk menyederhanakan struktur jaringan. Sensitivitas setiap input dievaluasi berdasarkan kepada data pelatihan dan data pengujian. Hasil eksperimen menunjukkan bahwa persentase error kalsisfikasi tidak meningkat setelah penghapusan atribut yang memeiliki sensitifitas yang rendah.

Daftar Pustaka

Castillo, E., Guijarro-Berdinas, B., Fontenla-Romero, O., dan Alonso-Betanzos, A., *A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis*, Journal of Machine Learning Research, 7 (2006), 1159–1182.

Du, K.-L. and Swamy, M.N.S., *Neural Networks in a Softcomputing Framework*, Springer-Verlag, London, 2006

Faussett, L., *Fundamentals of Neural Networks*: *Architectures, Algorithms, and Application*, Prentice-Hall, Inc., New Jersey, 1994

Gupta, M.M., Jin, L. and Homma, N., *Static and Dynamic Neural Networks*: *From Fundamental to Advanced Theory*, John Wiley & Sons, New York, USA, 2003

Huang, D.S. and Zhao, W.B., *Determining the Center of Radial Basis Probabilistic Neural Networks by Recursive Orthogonal Least Square Algorithm*, Applied Mathematics and Computation, 162 (2005), 461-473.

Karayiannis, N.B., *Reformulated Radial Basis Neural Networks by Gradient Descent*, IEEE Transaction on Neural Network, 10 (1999) , No. 3, 657-671.

Shi, D., Yeung, D.S. and Gao, J., *Sensitivity Analysis Applied to the Construction of Radial Basis Function Networks*, Journal of Neural Networks, 18 (2005), hal. 951–957.

Wang, X.-Z., Li, C.-G., Yeung, D.S., Song, S.J. and Feng, H.M., *A Definition of Partial Derivative of Random Functions and Its Application to RBFNN Sensitivity Analysis*, Journal of Neurocomputing, 71 (2008), 1515–1526.

Zhao, W.B. and Huang, D.S., *Radial Basis Probabilistic Neural Networks of Genetic Optimization of Full Structure*, J. Infrared Millim. Waves, 23 (2004) no. 2, 113-118.

Zurada, J..M., Malinowski, A. and Usui, S., *Pertubation for Deleting Redundant Inputs of Perceptron Networks*, *Neurocomputing*, 14 (1996), 177-193.