PENENTUAN KLASIFKASI JAMUR *AGARICUS LEPIOTA* MENGGUNAKAN METODE *BACK-PROPAGATION*

Muhammad Rafi Muttaqin [1] , Muhammad Asyhar Agmalaro [2]

[1] Program Studi Teknik Informatika, STT Wastukancana Purwakarta
[2] Departemen Ilmu Komputer, Institut Pertanian Bogor
[1] rafi_aqin@yahoo.com, [2] agmalao_03@yahoo.com

Abstrak

Keluarga jamur *Agaricus lepiota* merupakan salah satu pengurai yang sangat efisien. Akan tetapi beberapa *species Agaricus* mengandung racun. Oleh karena itu, dibuatlah suatu teknik kecerasan komputasional yang dapat melakukan *training* untuk membedakan antara jenis jamur yang beracun dan tidak mengandung racun (dapat dimakan). Hal ini memerlukan data *training* 7000 jenis jamur yang berbeda berdasarkan 22 properti yang ada. Dengan menggunakan jaringan syaraf tiruan *back propagation* maka dapat diklasifikasikan jenis jamur yang beracun dan jamur yang tidak beracun(dapat dimakan). Program yang dibuat menggunakan fungsi *tansig* dan *purelin*. Hidden layer pada program di-set sebanyak 22 dan 1 *output* layer. Data training yang digunakan sebayak 1000 dengan proporsi 500 untuk kelas edible dan 500 untuk kelas poisonous. Hasil yang didapatkan dari mempunyai keakuratan sebesar 99.99%. Kompleksitas dari program ini adalah O (n.t) dengan n adalah jumlah data set, satu data set terdiri dari satu buah kelas tidak beracun dan kelas beracun dan t adalah waktu eksekusi satu buah data set. Karena dalam uji coba program ini menggunakan 500 data set, maka kompleksitas program ini sebesar 500t.

Kata kunci : ***Agaricus Lepiota, Backpropagation, tansig, purelin***

1. Pendahuluan

Keluarga jamur *Agaricus lepiota* merupakan salah satu pengurai yang sangat efisien. Keluarga jamur jenis ini dapat menguraikan bahan yang sulit diuraikan oleh bakteri. Jamur tersebut memegang peranan penting dalam mendaur ulang karbon yang dihasilkan oleh bahan organik yang berasal dari tanaman. Beberapa *species Agaricus* mengandung racun.

Oleh karena itu, dibuatlah suatu jaringan yang dapat melakukan *training* untuk membedakan antara jenis jamur yang beracun dan tidak mengandung racun (dapat dimakan). Hal ini memerlukan data *training* 7000 jenis jamur yang berbeda berdasarkan 22 properti yang ada. Dengan menggunakan jaringan syaraf tiruan *back propagation* maka dapat diklasifikasikan jenis jamur yang beracun dan jamur yang tidak beracun (dapat dimakan).

1.1 Ruang Lingkup

Ruang lingkup penelitian ini adalah sebagai berikut :

1. Proses pelatihan untuk jaringan syaraf tiruan menggunakan metode *back propagation*,
2. Atribut informasi yang digunakan pada penelitian ini sebanyak 22 atribut, yaitu :
   1) *cap-shapes: bell=b, conical=c, covex=x, flat=f, knobbed=k, sunken=s*
   2) *cap-surface: fibrous=f, grooves=g, scally=y, smooth=s*
   3) *cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y*
   4) *bruises: bruises=t, no=f*
   5) *odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s*
   6) *gill-attachment: attached=a, descending=d, free=f, notched=n*
   7) *gill-spacing: close=c, crowded=w, distant=d*
   8) *gill-size: broad=b, narrow=n*
   9) *gill-collor: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y*
   10) *stalk-shape: enlarging=e, tapering=t*
   11) *stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?*
   12) *stalk-surface-above-ring: ibrous=f, scaly=y, silky=k, smooth=s*
   13) *stalk-surface-below-ring: ibrous=f, scaly=y, silky=k, smooth=s*

14) *stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y*
15) *stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y*
16) *veil-type: partial=p, universal=u*
17) *veil-color: brown=n, orange=o, white=w, yellow=y*
18) *ring-number: none=n, one=o, two=t*
19) *ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z*
20) *spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y*
21) *population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y*
22) *habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d*

Jenis keluaran yang dihasilkan yaitu dapat dimakan (*edible*) atau beracun (*poisonous*).

1.2 Tujuan

Tujuan dari penelitian ini yaitu untuk menentukan suatu jenis jamur hasil pengujian dapat dimakan atau beracun dengan menggunakan metode *backpropagation*.

1.3 Manfaat

Manfaat dari penelitian ini adalah dapat mengklasifikasikan suatu jamur dari keluarga *Agraricus* yang dapat dimakan (*edible*) dan beracun (*poisonous*).

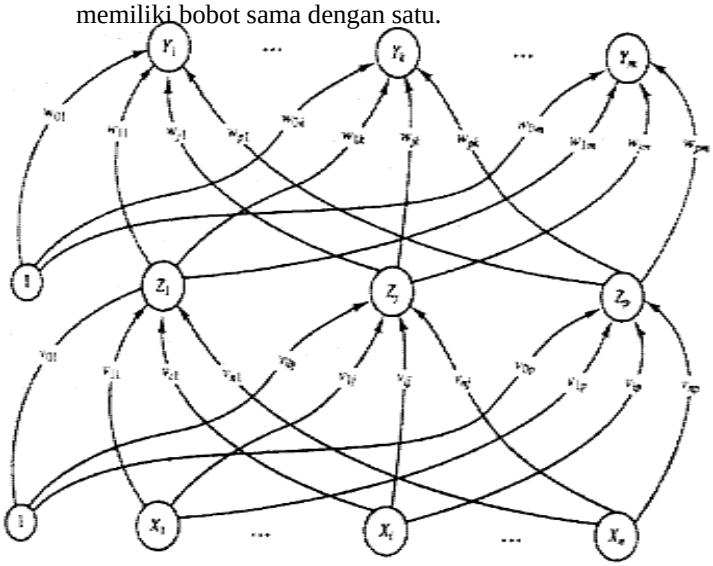
2. Tinjauan Pustaka

2.1 *Agaricus Lepiota*

Keluarga jamur *Agaricus lepiota* merupakan salah satu pengurai yang sangat efisien. Keluarga jamur jenis ini dapat menguraikan bahan yang sulit diuraikan oleh bakteri. Jamur tersebut memegang peranan penting dalam mendaur ulang karbon yang dihasilkan oleh bahan organik yang berasal dari tanaman. Lepiota memililki ciri-ciri yaitu terdapat topi berdaging/pileur pada permukaan atasnya, kemudian batang/stipe yang menopang pileus sehingga membentuk seperti payung. Antara batang dan pileus terdapat cincin dan memiliki spora bewarna putih hingga kecoklat-coklatan, dan termasuk dalam genus Agaricus. Walau banyak dibudidayakan dan mempunyai khasiat sebagai pengurai zat atau bahan, namun banyak dari jamur ini yang sangat berbahaya untuk dikonsumsi karena menggandund zat amanitins dan sangat beracun sehingga dapat menyebabkan kematian apabila dimakan.
(http://wikipedia.co.id/lepiota)

2.2 Jaringan Saraf Tiruam (JST) Propagasi Balik

JST Propagasi balik merupakan JST yang mempunyai topologi *multilayer* (multi-lapis) dengan lapis *input* (lapis *X*) , satu atau lebih lapisan *hidden* (lapis *Z*) dan satu lapis *output* (lapis *Y*) (Siang, 2005). Setiap lapis memiliki neuron-neuron yang dimodelkan dengan lingkaran (lihat Gambar 1). Di antara neuron pada satu lapis dengan neuron pada lapis berikutnya dihubungkan dengan koneksi yang memiliki bobot-bobot (*weights*), **w** dan **v**. Lapis tersembunyi dapat memiliki *bias*, yang memiliki bobot sama dengan satu.



Gambar 1. Jaringan saraf tiruan propagasi balik dengan satu lapis tersembunyi.

2.3 Algoritma Pelatihan JST Propagasi Balik

Algoritma pelatihan JST propagasi balik pada dasarnya dapat dibagi menjadi dua langkah, yaitu: langkah maju (*feedforward*) dan propagasi balik (*back propagation*) (Siang, 2005). Pada langkah maju, perhitungan bobot-bobot neuron hanya didasarkan pada vektor masukan, sedangkan pada propagasi balik, bobot-bobot diperhalus dengan cara memperhitungkan nilai dari target atau keluaran. Algoritma ini dapat dilihat pada Gambar 2.

Nilai *mean square error* (MSE) pada suatu siklus pelatihan (langkah 2 – 10, dimana seluruh rekord dipresentasikan satu kali) adalah nilai kesalahan (*error* = nilai keluaran - nilai masukan) rata-rata dari seluruh rekord (*tupple*) yang dipresentasikan ke JST dan dirumuskan sebagai berikut (Vaisla & Bhatt, 2010):

$$MSE = \left(\frac{\sum error^2}{jumlah\_record} \right)$$

Semakin kecil nilai MSE maka semakin kecil nilai kesalahan pada JST dalam memprediksi kelas dari rekord yang baru. Dengan demikian, pelatihan JST ditujukan untuk memperkecil MSE dari satu siklus ke siklus berikutnya sampai selisih nilai MSE pada siklus ini dengan siklus sebelumnya lebih kecil atau sama dengan batas minimal yang diberikan (*epsilon*).

2.4 Klasifikasi Data

Klasifikasi data ini dilakukan dengan langkah *feedforward* dan hasilnya adalah berupa kelas yang diprediksi JST untuk *record* yang baru ini

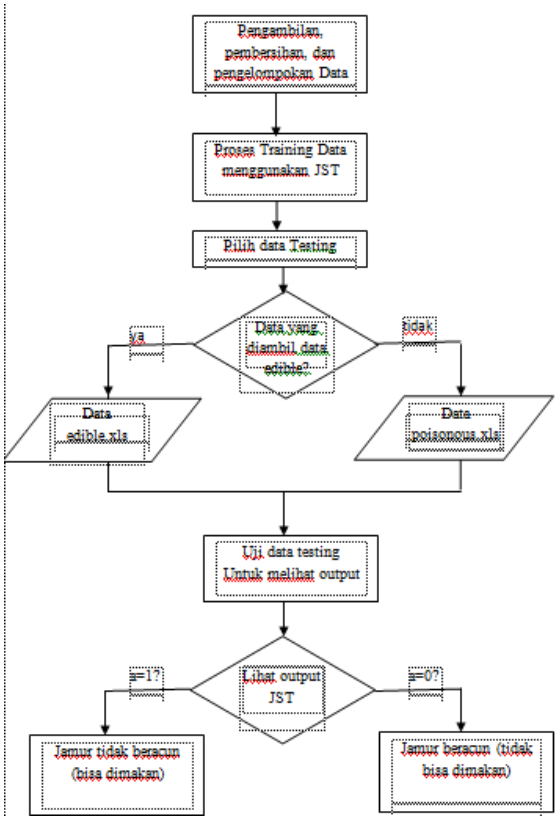
Narasi: Melatih JST dengan data pelatihan (berupa vektor atau rekord-rekord tabel) yang diberikan sampai bobot-bobot tidak berubah lagi (atau dicapai kondisi konvergen).
Input: Set data pelatihan , jumlah lapis, jumlah neuron, *learning rate*, epsilon
Output: Model JST yang siap untuk mengklasifikasi data (vektor) baru.
Algoritma:
(1) Menginisialisasi bobot awal dengan nilai acak yang sangat kecil, hitung MSE dan ΔMSE inisialisasi.
(2) Selama ΔMSE > epsilon lakukan:
(3) Untuk setiap *tupple* pada set data pelatihan lakukan
Feedforward:
(4) Setiap unit masukan (X_i, i=1..n) menerima vektor masukan X_i dan mengirimkan vektor ini ke seluruh unit pada lapis diatasnya (*hidden layer*).
(5) Setiap unit *hidden* (Z_j, j=1..p) menjumlahkan bobot dari vektor masukan: $z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$ Hitung keluaran fungsi aktivasi: $z_j = f(z\_in_j)$
Kirimkan vektor ini ke unit-unit pada *lapis* diatasnya (lapis keluaran)
(6) Setiap unit keluaran (Y_k, k=1..m) menjumlahkan vektor masukan:. $y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$
Hitung keluaran dari fungsi aktivasi: $y_k = f(y\_in_k)$
Propagasi balik dari *error*:
(7) Setiap unit keluaran (Y_k, k=1..m) menerima vektor hasil yang diinginkan (t_k) untuk data masukan tersebut, hitung *error*-nya $(t_k - y_k)$: $\delta_k = (t_k - y_k) f'(y\_in_k)$
Hitung nilai koreksi bobotnya dengan α sebagai *learning rate*nya: $\Delta w_{jk} = \alpha \delta_k z_j$
Hitung nilai koreksi *bias*nya: $\Delta w_{0k} = \alpha \delta_k$
Kirimkan δ_k ke unit pada *lapis* dibawahnya.
(8) Setiap unit *hidden* (Z_j, j=1..p) menjumlahkan delta masukannya (dari unit-unit pada lapis diatasnya): $\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$
Kalikan dengan turunan dari fungsi aktivasinya untuk menghitung *error*nya: $\delta_j = \delta\_in_j f'(z\_in_j)$ Hitung nilai koreksi bobotnya: $\Delta v_{ij} = \alpha \delta_j x_i$ Hitung nilai koreksi *bias*nya: $\Delta v_{0j} = \alpha \delta_j$
Perbaharui bobot dan *bias*:
(9) Setiap unit keluaran (Y_k, k=1..m) memperbaharui *bias* dan bobotnya (j=0..p).
$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$
Setiap unit tersembunyi (Z_j, j=1..p) memperbaharui *bias* dan bobotnya (i=0..n).
$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$
MSEold = MSE. Hitung MSE, ΔMSE = MSE – MSEold.
(10) Uji kondisi berhentinya

Gambar 2. Algoritma pelatihan pada JST propagasi balik

Keterangan Notasi untuk Algoritma 1: x = vektor masukan = (x_1,…, x_i,…, x_n), t = vektor keluaran = (t_1,…, t_k,…, t_m). δ_k = nilai koreksi bobot *error* untuk w_{jk} yang disebabkan oleh *error* pada unit keluaran Y_k . δ_j = nilai koreksi bobot *error* untuk v_{ij} yang disebabkan oleh informasi propagasi balik dari *error* pada *lapis* keluaran ke unit tersembunyi Z_j . α = konstanta laju pembelajaran (*learning rate*). X_i = unit masukan i. v_{0j} = bias pada unit tersembunyi j. Z_j = unit tersembunyi j. w_{0k} = bias pada unit keluaran k. Y_k = unit keluaran k. Fungsi aktivasi yang dapat digunakan pada propagasi balik antara lain adalah: **Binary sigmoid:** $f(x) = \frac{1}{1+e^{-x}}$, arc tangen: $f(x) = \frac{2}{\pi}\arctan(x)$, radial basis: $f(x) = e^{-\sqrt{x}}$ (Siang, 2005).

3. Metode Penelitian

Metode yang digunakan dalam penelitian ini dapat dilihat pada Gambar 3.



Gambar 3. Metodologi penelitian JST untuk klasifikasi Jamur *Agaricus Lepiota*.

3.1 Pengambilan Data

Data diambil dari sebuah website (http://uci.com). Data yang diambil merupakan data *Agaricus Lepiota*. Data ini terdiri dari 8124 data, dibagi menjadi 2 kelas, yaitu kelas *edible* dan *poisonous*. Kelas *edible* sebanyak 4208 buah, dah kelas *poisonous* sebanyak 3916 buah.

3.2 Pembersihan Data

Setelah data terkumpul, ternyata terdapat beberapa data yang tidak mempunyai nilai pada beberapa atribut. Karena hal demikian, dilakukan proses pembersihan data dengan menghilangkan

data yang tidak mempunyai nilai atribut lengkap. Data yang dihilangkan sebanyak 2480 buah. Jadi total data setelah dibersihkan sebanyak 5644 buah.

3.3 Pengelompokan Data

Setelah data dibersihkan, maka data dikelompokan menjadi 2 buah berdasarkan kelas targetnya. Kelompok yang pertama berisi 3488 buah data yang masuk ke dalam kelas *edible*, dan kelompok yang kedua berisisi 2156 buah data yang masuk ke dalam kelas *poisonous*.

3.4 Proses *Training* Data

Proses *training* dilakukan dengan menggunakan metode *feed-fordward backpropagation*. Data yang dilatih sebanyak 1000 buah data, terdiri dari 500 buah data kelas *edible* dan 500 buah data kelas *poisonous*. Proses training tidak dilakukan menggunakan keseluruhan data disebabkan terdapat data ribuan sehingga menyebabkan proses training membutuhkan waktu yang lama, sehingga diambil 1000 data yang dianggap mewakili keseluruhan data.

3.5 Pengujian / Testing

Proses pengujian dilakukan dengan memasukan data test secara manual di dalam barisan kode program yang dijalankan. Jika hasil yang didapatkan sesuai dengan target dari data testing tersebut maka dapat dikatakan proram tersebut berhasil, jika tidak sesuai, dilakukan kembali proses *training* dengan merubah beberapa parameter, seperti jumlah hidden layer atau jumlah data yang akan dilatih.

4. Hasil Dan Pembahasan

Dalam pembuatan program ini, seluruh atribut yang digunakan dikonversi terlebih dahulu kedalam angka. Hasil konversi dapat dilihat pada Tabel 1.

Tabel 1. Konversi Atribut Menjadi Angka

Nama Variabel	Nama Atribut	Variabel Konversi
Cap-Shapes	Bell	0
	Conical	1
	Covex	2
	Flat	3
	knobbed	4
	sunken	5
Cap-Surface	Fibrous	0
	grooves	1
	scally	2
	Smooth	3
Cap-Color	Brown	0
	Buff	1
	Cinnamon	2
	Gray	3
	Green	4
	Pink	5
	Purple	6
	Red	7
	White	8
	Yellow	9
Bruises	Bruises	1
	No	0
Odor	Almond	0
	Anise	1
	Creosote	2
	Fishy	3
	Foul	4
	Musty	5
	None	6
	Pungent	7
	Spicy	8
Gill-Attachment	Attached	0
	Descending	1
	Free	2
	Notched	3
Gill-Spacing	Close	0

	Crowded	1
	Distant	2
Gill-Size	Broad	0
	Narrow	1
Gill-Collor	Black	0
	Brown	1
	Buff	2
	Chocolate	3
	Gray	4
	Green	5
	Orange	6
	Pink	7
	Purple	8
	Red	9
	White	10
	Yellow	11
Stalk-Shape	Enlarging	0
	Tapering	1
Stalk-Root	Bulbous	0
	Club	1
	Cup	2
	Equal	3
	Rhizomorphs	4
	Rooted	5
	Missing	6
Stalk-Surface-Above-Ring	Ibrous	0
	Scaly	1
	Silky	2
	Smooth	3

Stalk-Surface-Below-Ring	Ibrous	0
	Scaly	1
	Silky	2
	Smooth	3
Stalk-color-above-ring	Brown	0
	Buff	1
	Cinnamon	2
	Gray	3
	Orange	4
	Pink	5
	Red	6
	White	7
	Yellow	8
Stalk-color-below-ring	Brown	0
	Buff	1
	Cinnamon	2
	Gray	3
	Orange	4
	Pink	5
	Red	6
	White	7
	Yellow	8
Veil-type	Partial	0
	Universal	1
Veil-color	Brown	0
	Orange	1
	White	2
	Yellow	3
Ring-number	None	0

	One	1
	Two	2
Ring-type	Cobwebby	0
	Evanescent	1
	Flaring	2
	Large	3
	None	4
	Pendant	5
	Sheathing	6
	Zone	7
Spore-print-color	Black	0
	Brown	1
	Buff	2
	Chocolate	3
	Green	4
	Orange	5
	Purple	6
	White	7
	Yellow	8
Population	Abundant	0
	Clustered	1
	Numerous	2
	Scattered	3
	Several	4
	Solitary	5
Habitat	Grasses	0
	Leaves	1
	Meadows	2
	Paths	3
	Urban	4
	Waste	5
	woods	6

Listing programnya adalah sebagai berikut :

```
function agaricus
clc
clear all
%dengan newff
rangeinput=[0 5;0 3;0 9;0 1;0 8;0 3;0 2;0 1;0
11;0 1;0 6;0 3;0 3;0 8;0 8;0 1;0 3;0 2;0 7;0 8;0
5;0 6];
%net =newff(rangeinput,[22 2],{'tansig'
'purelin'});
net =newff(rangeinput,[22 1],{'tansig'
'logsig'});

load('edible.mat');
load('poisonous.mat');
% 1 nilai untuk edible (jamur yang tidak
beracun)
% 0 nilai untuk poisonous (jamur yang beracun)
v=[1 0];
p1=xlsread('edible.xls',1,'A1:V10');
p1=p1';
p2=xlsread('poisonous.xls',1,'A1:V10');
p2=p2';
p=[p1 p2];
%net =newff(rangeinput,[22 1],{'tansig'
'logsig'});
t1=ones(1,10);
t2=zeros(1,10);
t=[t1 t2];

net.trainParam.epochs = 100;
net=train(net,p,t);
%melihat semua nilai semua bobot dari lapisan
input ke layer 1
disp('net.IW{1,1}-->');
net.IW{1,1}%input ke hiden layer
%melihat semua nilai semua bobot dari layer 1 ke
layer 2
disp('net.LW{2,1}-->');
net.LW{2,1}%w->dari hiden ke output
%melihat bobot nilai bias pada layer 1
disp('net.b{1}-->');
net.b{1}
%melihat bobot nilai bias pada layer 2
disp('net.b{2}-->');
net.b{2}

%datatesting=datatesting';
%datatesting
=[3;3;8;0;6;2;1;0;1;1;3;3;0;6;6;0;2;1;1;0;3;0];%
kelas edible 1
datatesting
=[3;0;9;0;4;2;0;0;7;0;0;2;2;0;5;0;2;1;3;3;5;0];
%kelas poisonous 0
%datatesting=datatesting'
a = round(sim(net,datatesting))
disp ('untuk datatesting berikut:');
datatesting=datatesting'
if a==v(1,1)
disp ('Hasilnya adalah Jamur Tidak Beracun');
else disp ('Hasilnya adalah Jamur Beracun');
end
```

Program yang dibuat menggunakan fungsi tansig dan purelin. Tansig digunakan sebagai fungsi aktivasi pada *output layer*. Tansig lebih akurat dan direkomendasikan untuk aplikasi yang membutuhkan tangen hiperbolik. Tansig memanggil fungsi newff untuk membuat *network* sederhana algoritma tansig adalah tansig (N) : n = 2/(1+exp(-2*n))-1 secara matematika akan equivalen dengan tanh (N). Fungsi Purelin digunakan sebagai fungsi aktivasi pada *hidden layer*. Purelin digunakan untuk membuat *network* sederhana dengan memanggil fungsi newlin atau newlind algoritmanya adalah purelin (n) = n.

Pada program Matlab di atas menggunakan jaringan saraf tiruan feed-forward propagation. Algoritma newff adalah jaringan feed-forward terdiri dari N1 layer menggunakan fungsi bobot dotprod, fungsi *input* jaringan netsum, dan fungsi transfer yang spesifik. Layer pertama mempunyai bobot yang berasal dari *input*. Masing-masing layer subsequent mempunyai bobot berasal dari layer sebelumnya semua layer mempunyai bias. Layer terakhir adalah *output* dari jaringan. Masing-masing bobot layer dan bias diinisialisasi dengan initnw (initial number of weight). Adaptasi dilakukan dengan trains, yang meng-*update* bobot dengan fungsi learning spesifik. Training dilakukan dengan fungsi learning spesifik. Performa diukur berdasarkan fungsi fungsi performa spesifik.

Hidden layer pada program di-set sebanyak 22 dan 1 *output* layer. Data training yang digunakan sebayak 1000 dengan proporsi 500 untuk kelas edible dan 500 untuk kelas poisonous. Hasil *output* program dari data testing yang diberikan yaitu:

```
a =

     0
untuk datatesting berikut:

datatesting =

  Columns 1 through 12
     3     0     9     0     4     2
0     0     7     0     0     2
  Columns 13 through 22
     2     0     5     0     2     1
3     3     5     0
Hasilnya adalah Jamur Beracun
```

Dari output program, dapat diketahui bahwa dari hasil dari tes data testing dikenali sebagai jenis jamur *poisonous* atau tidak dapat dimakan dimana layer output a bernilai bol. Hasil a didapatkan dari hasil pembulatan ke bawah dan mendekati nol dan menghasilkan klasifikasi datatsting termasuk dalam kelas *poisonous* (p) dengan keakuratan sebesar 99.99%. Kompleksitas dari program ini adalah O (n.t) dengan n adalah jumlah data set, satu data set terdiri dari satu buah kelas ‘e’ dan kelas ‘p’ dan t adalah waktu eksekusi satu buah data set. Karena dalam uji coba program ini menggunakan 500 data set, maka kompleksitas program ini sebesar 500t.

5. Kesimpulan dan Saran

Kesimpulan

Agaricus-lepiota mempunyai dua jenis kelas, yaitu beracun (*poisonous*) dan tidak beracun (*edible*). Klasifikasi jamur ini dapat dilakukan dengan metode *back propagation* dengan hasil yang akurat. Arsitektur *bak propagation* yang digunakan memiliki 22 *node* pada *input layer*, 22 *node* pada *hidden layer* dan 1 *node* pada *output layer*. Penentuan jumlah *node* pada *input layer* berdasarkan banyaknya properti yang tersedia pada data. Penentuan jumlah *node* pada *output layer* berdasarkan jumlah kelas. Akan tetapi, penentuan jumlah *node* pada *hidden layer* masih dilakukan dengan cara coba-coba.

Saran

Walaupun keakuratan program ini tinggi, kami menyarankan untuk tidak langsung percaya akan hasil klasifikasi program ini. Hal ini dikarenakan adanya kemungkinan properti-properti lain yang mungkin belum dimasukkan pada data *input*.

Daftar Pustaka

Siang Jong Jek. 2005. *Jaringan Syaraf Tiruan & Pemrograman Mengunakan MATLAB.* Andi : Yogyakarta.

Vaisla, Kunwar Singh & Bhatt, Ashutosh Kumar. 2010. *An Analysis Of The Performance Of Artificial Neural Network Technique For Stock Market Forecasting*. International Journal of Computer Science and Engineering, Vol. 02, 2104-2109.

http://uci.com tanggal akses 10 mei 2010

http://wikipedia.co.id tanggal akses 17 mei 2010