PREDIKSI STRUKTUR SEKUNDER PROTEIN MENGGUNAKAN JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer
dalam bidang Ilmu Komputer

Oleh :
FAKHRUDDIN KHASBULLAH SALEH ATASOGE
0410960021-96

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

2009


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PREDIKSI STRUKTUR SEKUNDER PROTEIN MENGGUNAKAN JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION*

Oleh :

FAKHRUDDIN KHASBULLAH SALEH ATASOGE
0410960021-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 29 Juni 2009
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs. Achmad Ridok, M.Kom.	**Nurul Hidayat, S.Pd., M.Sc.**
NIP. 132 090 392	**NIP. 132 300 240**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, M.Sc.
NIP. 132 126 049

UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Fakhruddin Khasbullah Saleh Atasoge**
NIM	**: 0410960021**
Jurusan	**: Matematika**
Judul Skripsi Penulis	**: Prediksi Struktur Sekunder Protein Menggunakan Jaringan Syaraf Tiruan *Learning Vector Quantization***

Dengan ini menyatakan bahwa :

1. **Isi dari 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 dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 29 Juni 2009
Yang menyatakan,

(Fakhruddin Khasbullah Saleh Atasoge)
NIM. 0410960021

PREDIKSI STRUKTUR SEKUNDER PROTEIN MENGGUNAKAN JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION*

ABSTRAK

Prediksi struktur sekunder protein adalah salah satu usaha awal untuk dapat menentukan bagaimana bentuk 3 dimensi dari suatu sekuens asam amino. Struktur sekunder protein dapat ditentukan dengan metode *NMR Spectroscopy* dan *X-Ray Crystallography*. Akan tetapi membutuhkan waktu yang lama. Perlu ditemukan cara yang lebih singkat. Salah satunya dengan menggunakan jaringan syaraf tiruan. Metode jaringan syaraf tiruan yang belum banyak digunakan untuk prediksi struktur sekunder protein salah satunya adalah *Learning Vector Quantization*. Dalam mengkodekan residu asam amino dari suatu sekuens asam amino digunakan metode normalisasi data yang dihubungkan dengan bilangan ASCII. Hasil pengkodean dari residu asam amino digunakan untuk proses prediksi dengan menggunakan algoritma *Learning Vector Quantization*. Sistem prediksi struktur sekunder protein yang mengimplementasikan algoritma *Learning Vector Quantization* telah dilatih dengan batas iterasi maksimum sebesar 50, 100, 150, 200 dan sampai perubahan bobot data referensi mencapai nilai konvergen. Tingkat keakurasian sistem yang diukur dengan metode Q3* mencapai nilai terbesar dengan iterasi maksimum training sebesar 50 kali, yaitu sebesar 46,92 %

PROTEIN SECONDARY STRUCTURE PREDICTION USING LEARNING VECTOR QUANTIZATION NEURAL NETWORK

ABSTRACT

Protein secondary structure prediction is one of the early efforts to determine how the 3-dimensional shape of an amino acid sequence. Protein secondary structure can be determined with NMR Spectroscopy and X-Ray Crystallography method. But that methods requires long time. It's need to find a shorter way. One of them using the Artificial Neural Network. Artificial Neural Network method which is not widely used for prediction of protein secondary structure is Learning Vector Quantization. In encode the amino acid residues from an amino acid sequence, data normalization method is used associated with ASCII number. The result from encoding process is used to process input using the Learning Vector Quantization algorithm. Protein secondary structure prediction system with Learning Vector Quantization algorithm was trained with maximum epoch 50, 100, 150, 200 and until convergent value. Accuracy level system with maximum epoch 50 reached the largest value, that measured Q3* method, amounting to 46.92%

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahNya, Skripsi yang berjudul "PREDIKSI STRUKTUR SEKUNDER PROTEIN MENGGUNAKAN JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION* " ini dapat diselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya.

Dalam penyelesaian skripsi ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Achmad Ridok, M.Kom., Nurul Hidayat, S.Pd., M.Sc**.,** dan Yusi Tyroni M., S.Kom. terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Wayan Firdaus Mahmudy, SSi, MT selaku Penasihat Akademik dan Ketua Program Studi Ilmu Komputer Universitas Brawijaya Malang.
3. Dr. Agus Suryanto, M.Sc. selaku ketua jurusan Matematika Fakultas MIPA Universitas Brawijaya Malang.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
5. Segenap staf dan karyawan di Jurusan Matematika Fakultas MIPA Universitas Brawijaya
6. Ayah, Ibu dan Kakak. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
7. Teman-teman Ilkomers dan Matematika. Terima kasih atas senyuman, semangat, dukungan, do'a dan hari-hari kita.
8. Pihak lain yang telah membantu terselesaikannnya Skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Semoga penulisan laporan skripsi ini bermanfaat bagi pembaca sekalian. Dengan tidak lupa kodratnya sebagai manusia, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, dan mengandung banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 29 Juni 2009

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR *SOURCECODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Protein merupakan salah satu zat yang penting dalam kehidupan makhluk hidup, tidak terkecuali pada manusia. Di dalam tubuh manusia, protein terdapat pada seluruh bagian misalnya pada kulit, otot, rambut, darah, organ tubuh, mata, termasuk kuku dan tulang (Lauritzen,1992). Protein tersusun dari rangkaian asam amino. Rangkaian asam amino ini akan membentuk struktur protein. Terdapat 4 struktur protein, yaitu struktur primer, sekunder, tersier dan kwartener (Fessenden, 1997).

Asam-asam amino yang menyusun suatu protein menentukan bagaimana bentuk 3 dimensi suatu protein. Bentuk 3 dimensi suatu protein menentukan fungsinya. Prediksi struktur sekunder protein mempunyai tujuan untuk memprediksikan bagaimana bentuk dari 3 dimensi suatu sekuens asam amino yang belum diketahui fungsinya. Hal ini sangat membantu seperti dalam bidang pembuatan obat yang menggabungkan beberapa fungsi-fungsi khusus dari beberapa jenis protein. Peran dari prediksi struktur sekunder adalah mendapatkan gambaran awal bagaimana bentuk 3 dimensi dari sekuens asam amino yang diujicoba sehingga membantu peneliti dalam menentukan langkah lebih lanjut.

Prediksi bentuk 3 dimensi dari sebuah protein dapat dilakukan cara *X-ray crystallography* dan *NMR spectroscopy*. Jika menggunakan kedua metode ini, akan menghasilkan bentuk 3 dimensi dari protein dengan akurat, tetapi membutuhkan waktu yang lama dan proses yang sulit. Oleh karena itu, dibutuhkan metode baru yang cepat dan mudah. Salah satu metode yang dikembangkan adalah prediksi dengan menggunakan jaringan syaraf tiruan.

Bermacam-macam algoritma jaringan syaraf tiruan telah digunakan untuk memprediksi bentuk struktur sekunder protein, contohnya pada tahun 1988 Qian dan Sejnowski menerapkan algoritma *Backpropagation* untuk prediksi struktur sekunder protein. Salah satu algoritma dalam jaringan syaraf tiruan yang lain adalah *Learning Vector Quantization* atau disingkat *LVQ*. Penggunaan algoritma *LVQ* dalam memprediksikan struktur sekunder protein masih belum terlalu luas digunakan.

Dari latar belakang diatas, dikembangkan sebuah sistem prediksi struktur sekunder protein yang mengimplementasikan algoritma *LVQ*. Algoritma *LVQ* pada prediksi struktur sekunder protein mempunyai masukkan berupa sekuens asam amino penyusun sebuah protein. Kemudian, algoritma *LVQ* akan memprediksikan bentuk struktur sekunder dari sekuens asam amino yang diinputkan tersebut. Residu asam amino akan dikodekan dengan menggunakan metode normalisasi data. Metode normalisasi data akan dihubungkan dengan bilangan ASCII dari residu pada suatu rangkaian asam amino. Hasil pengkodean ini akan diimplementasikan dalam sistem prediksi struktur sekunder protein sebagai masukkan untuk algoritma *LVQ*. Kemudian, sistem prediksi struktur sekunder protein akan mengeluarkan output berupa struktur sekunder dari data referensi yang memiliki jarak *Euclidean* terkecil. Hasil yang dikeluarkan oleh sistem akan dihitung keakurasiaannya menggunakan metode Q3*, Q3, Qh, Qe dan Qc.

1.2 Rumusan Masalah

Dari latar belakang, dapat dibuat suatu rumusan masalah yaitu berapa tingkat keakurasian dari sistem prediksi struktur sekunder protein yang mengimplementasikan algoritma *LVQ* di dalamnya.

1.3 Batasan Masalah

Adapun batasan masalah dalam penulisan skripsi ini adalah:

1. Jumlah protein yang digunakan sebagai data referensi sebanyak 49 jenis protein, data pembelajaran sebanyak 50 jenis protein dan data uji coba sebanyak 10 jenis protein.
2. Pengkodean rangkaian asam amino berdasarkan bilangan ASCII dari residu asam amino dalam suatu rangkaian.
3. Jumlah residu asam amino yang dikelompokkan pada suatu kelas adalah 5 asam amino.
4. Faktor lain yang dapat mempengaruhi bentuk struktur sekunder protein diabaikan.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penulisan Skripsi ini adalah :

1. Mengetahui pengimplementasian algoritma *Learning Vector Quantization* dalam memprediksi struktur sekunder protein.

2. Mengetahui tingkat keakurasian sistem prediksi struktur sekunder protein dengan algoritma *Learning Vector Quantization* dalam memprediksi struktur sekunder protein.

1.5 Manfaat

Hasil dari penelitian ini diharapkan bermanfaat bagi pengembangan metode prediksi struktur sekunder protein dengan menggunakan metode jaringan syaraf tiruan.


UNIVERSITAS BRAWIJAYA

BAB II
TINJAUAN PUSTAKA

2.1 Protein

2.1.1 Pengertian Protein

Menurut Harper (1963), protein adalah bahan organik. Bila protein dididihkan dalam asam (basa encer) atau bila dikenai kerja enzim-enzim spesifik dalam pencernaan, molekul-molekulnya dihidrolisis menjadi asam-asam amino. Dapat disimpulkan bahwa protein merupakan bahan organik yang tersusun dari rangkaian asam-asam amino. Hal ini didukung oleh Mazur A. dan Benjamin Harrow (1971), bahwa protein tersusun atas kira-kira 20 jenis asam-asam amino berbeda yang terhubung satu sama lain dalam jumlah yang besar. Asam-asam amino saling terhubung dengan ikatan peptida. Sesuai dengan pendapat Campbell (1995), bahwa protein adalah rantai panjang dari asam-asam amino yang saling berkaitan menggunakan ikatan-ikatan peptida dengan sebuah grup amino yang berisi nitrogen bermuatan positif disuatu ujung dan grup karboksil yang bermuatan negatif diujung yang lain. Jika disimpulkan protein adalah bahan organik yang terbentuk dari rangkaian panjang asam-asam amino di mana asam-asam amino tersebut dihubungkan oleh ikatan peptida dengan sebuah grup amino yang berisi nitrogen bermuatan positif disuatu ujung dan grup karboksil yang bermuatan negatif diujung yang lain pada ikatan peptida tersebut.

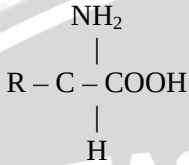
2.1.2 Struktur Molekul Protein

Molekul protein sederhana adalah molekul berantai panjang yang terbentuk oleh penggabungan ratusan atau ribuan molekul asam amino. Penggabungan itu berkat ikatan-ikatan yang masing-masing dapat dianggap berasal dari suatu gugus $–NH_2$ dan $–CO_2H$ dengan tereliminasinya sebuah molekul air. Ikatan-ikatan penting ini disebut ikatan peptida.

2.1.3 Pengertian Asam-Asam Amino

Asam-asam amino merupakan gabungan bahan-bahan organik yang berisi amino serta grup karboksil sehingga memiliki properti-properti dasar dan bersifat asam. Asam-asam amino adalah blok-blok

pembangun dasar dari protein (Sawhney, 2005). Secara umum, formula dari asam amino ditunjukkan pada gambar 2.1.



Gambar 2.1 Formula Umum dari Sebuah Asam Amino

Pada tabel 2.1, ditampilkan 20 jenis asam amino yang menyusun protein, lengkap dengan singkatannya.

Tabel 2.1 20 Jenis asam amino yang menyusun protein
(Abdalla dan Deris, 2005)

Jenis Asam Amino	Singkatan
Alanine	A
Arginine	R
Asparagine	N
Aspartic acid	D
Cysteine	C
Glutamic acid	E
Glutamine	Q
Glycine	G
Histidine	H
Isoleucine	I
Leucine	L
Lysine	K
Methionine	M
Phenylalanine	FV
Proline	P
Serine	S
Threonine	T
Tryptophan	W
Tyrosine	Y
Valine	V

2.1.4 Struktur Protein

Terdapat 4 jenis struktur protein, yaitu struktur primer, sekunder, tersier dan kwarterner. Pada penelitian ini, struktur yang digunakan adalah struktur primer dan struktur sekunder protein.

1. Struktur Primer

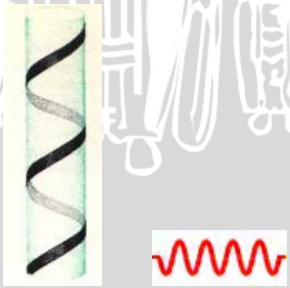
Struktur primer adalah urutan atau orde dari asam amino dalam rantai protein (Fessenden, 1997). Struktur primer berada pada tingkatan pertama dari struktur protein.

2. Struktur Sekunder

Struktur sekunder adalah tingkatan kedua dari struktur protein. Struktur sekunder adalah bentuk dari rantai hidrogen yang panjang yang dijadikan satu oleh ikatan hidrogen antara proton dari amida dan gugus karbonil amida (Fessenden, 1997). Struktur sekunder protein pada umumnya terbagi menjadi 3 macam, yaitu alfa heliks, *beta sheet* dan *coil*.

a. Alfa Heliks

Struktur sekunder protein pada kebanyakan adalah suatu alfa heliks, yaitu spiral tangan kanan *(right handed spiral)* dengan rantai yang berputar arah jarum jam. Bila dilihat dari sumbu bawah, D-asam amino membentuk heliks tangan kiri *(left-handed spiral)* atau yang berputar berlawanan arah jarum jam (Fessenden, 1997). Alfa heliks akan dinotasikan dengan karakter H pada sistem.

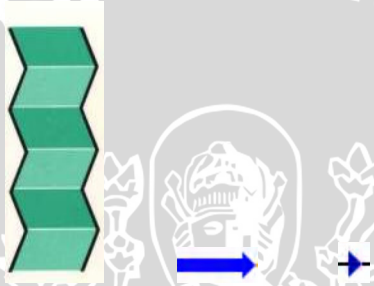


(a) (b)

Gambar 2.2 (a) Bentuk Alfa Heliks (b) Bentuk Alfa Heliks pada RSCB Protein Data Bank

b. *Beta pleated sheet*

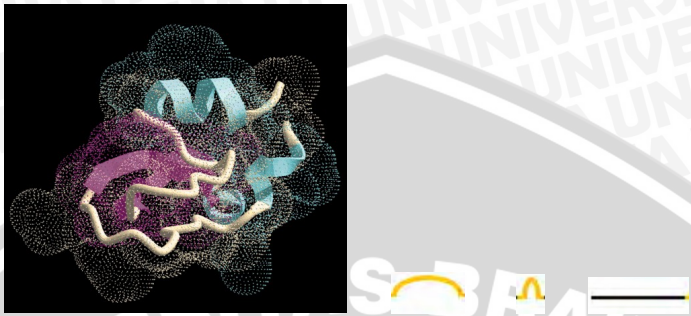
Struktur ini merupakan salah satu bentuk struktur sekunder protein. Istilah ini diambil karena bentuknya yang menyerupai *pleated sheet* atau genteng logam yang berkerut. Protein dalam bentuk *pleated sheet* dapat bertahan oleh adanya ikatan hidrogen antara rantai yang bersatu (Fessenden, 1997). *Beta pleated sheet* akan dinotasikan dengan karakter E pada sistem.



Gambar 2.3 (a) Bentuk *Beta pleated sheet*, (b) dan (c) Bentuk *Beta pleated sheet* pada RSCB Protein Data Bank

c. *Coil (Random coil)*

Bentuk *random coil* pada dasarnya adalah bentuk struktur sekunder selain alfa heliks dan *beta pleated sheet*. Alfa heliks dan *beta pleated sheet* adalah contoh yang paling lazim dari struktur sekunder, tetapi susunan teratur lain dari rantai polipeptida juga telah ditemukan. Jika pada satu bagian dari rantai polipeptida tidak dapat dilihat suatu pola tertentu, maka bagian tersebut disebut mempunyai struktur sekunder putaran acak *(random coil)* (Kimball, 1983). *Random coil* akan dinotasikan dengan karakter C pada sistem.



(a) (b) (c) (d)

Gambar 2.4 (a) Bentuk *Random coil*, (b) (c) dan (d) Bentuk *Random coil* pada RSCB Protein Data Bank

2.2 Prediksi Struktur Sekunder Protein

2.2.1 Skema Reduksi 8 menjadi 3

Dalam memprediksi struktur sekunder, terdapat beberapa pengelompokkan bentuk struktur sekunder. Dari 8 bentuk struktur sekunder dari RSCB Protein Data Bank akan direduksi menjadi 3. Jumlah reduksi menjadi 3 karena pada dasarnya bentuk struktur sekunder protein terdiri dari 3 macam. Yaitu alfa heliks, *beta pleated sheet* dan *random coil*. Reduksi didasarkan pada kemiripan dari 8 bentuk struktur sekunder kode DSSP. Skema reduksi ditunjukkan pada tabel 2.2 adalah skema reduksi yang digunakan pada metode PHD. PHD adalah salah satu metode prediksi struktur sekunder dengan menggunakan jaringan syaraf tiruan *feed-forward* dengan 2 layer. Reduksi yang ditunjukkan pada tabel 2.2 akan diterapkan pada penelitian prediksi struktur sekunder protein menggunakan jaringan syaraf tiruan *LVQ*. Penentuan bentuk struktur sekunder yang diterapkan pada metode PHD didasarkan pada algoritma pendefinisian struktur sekunder DSSP.

Tabel 2.2. Skema Reduksi 8-to-3 State
(Singh dkk, 2008)

Reduksi	Kode DSSP
H	H
H	G
H	I
C	B
E	E
C	T
C	S
C	-

2.3 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (*artificial neural network*) atau disingkat JST adalah sistem komputasi di mana arsitektur dan operasi diilhami dari pengetahuan tentang sel syaraf biologi didalam otak. JST dapat digambarkan sebagai model matematis dan komputasi untuk fungsi aproksimasi non linier, klasifikasi data, *cluster* dan regresi non parametrik atau sebagai sebuah simulasi dari koleksi model syaraf biologi.(Kristanto, 2004).

2.3.1 Normalisasi Data

Normalisasi data digunakan untuk menyamakan skala data ke dalam jangkauan nilai tertentu, misal -1 sampai dengan 1. Misalkan suatu data akan diubah kedalam suatu jangkauan nilai antara 0.05 dan 0.9, maka dapat dihitung menggunakan persamaan 2.1.

$$x' = \frac{0.9\,(x - \min values)}{(\max values - \min values)} + 0.05 \qquad (2.1)$$

di mana :
x' : data hasil normalisasi
x : data awal
min values : nilai terkecil dari seluruh data
max values : nilai terbesar dari seluruh data

Penggunaan persamaan 2.1 akan digunakan untuk mengkodekan residu asam amino dalam suatu rangkaian asam amino yang menjadi inputan dari jaringan syaraf tiruan dalam prediksi struktur sekunder protein.

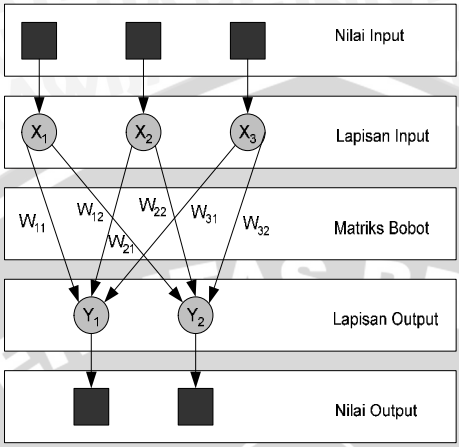
2.3.2 Arsitektur Jaringan Syaraf Tiruan

Jaringan syaraf tiruan terdiri dari neuron-neuron yang dikelompokkan dalam berbagai lapisan. Pada umumnya, lapisan-lapisan yang berada pada lapisan yang sama akan memiliki keadaan yang sama. Faktor yang menentukan keadaan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Untuk neuron pada lapisan yang sama akan memiliki fungsi aktivasi yang sama pula. Apabila neuron-neuron yang terletak pada lapisan yang sama akan dihubungkan dengan neuron-neuron pada lapisan yang lain, maka setiap neuron yang terletak pada lapisan tersebut juga harus dihubungkan dengan setiap lapisan pada lapisan lainnya. Misalkan akan dihubungkan neuron-neuron pada lapisan tersembunyi dengan neuron-neuron pada lapisan output, maka setiap neuron pada lapisan tersembunyi harus dihubungkan dengan setiap neuron pada lapisan output.

Ada beberapa arsitektur jaringan syaraf tiruan, diantaranya (Kusumadewi, 2003):

a. Jaringan dengan lapisan tunggal (*single layer net*)

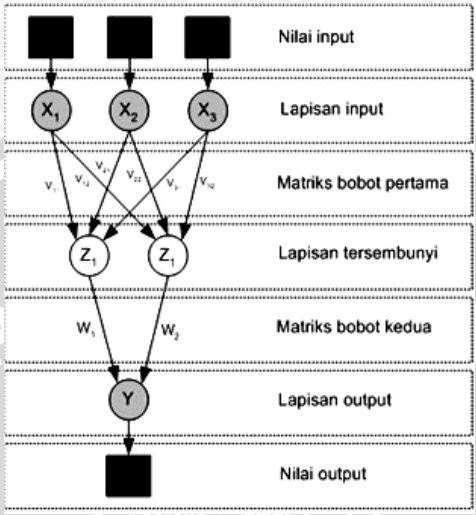
Sesuai dengan namanya, jaringan ini hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi output tanpa melalui lapisan tersembunyi. Gambar 2.5 menunjukkan arsitektur jaringan syaraf tiruan dengan lapisan tunggal.



Gambar 2.5 Jaringan Syaraf dengan Lapisan Tunggal

b. Jaringan dengan banyak lapisan (*multilayer net*)

Jaringan dengan banyak lapisan mempunyai satu atau lebih lapisan yang terletak diantara lapisan input dengan lapisan output. Lapisan ini disebut sebagai lapisan tersembunyi (*hidden layer*). Jaringan yang mempunyai banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan tunggal, tentu dengan pembelajaran yang lebih rumit. Akan tetapi, pada beberapa kasus pembelajaran jaringan yang mempunyai banyak lapisan ini lebih sukses dalam menyelesaikan masalah. Gambar 2.6 menunjukkan arsitektur jaringan syaraf tiruan dengan banyak lapisan.



Gambar 2.6 Jaringan Syaraf dengan Banyak Lapisan

2.3.3 Proses Pembelajaran

Jaringan syaraf akan mencoba untuk mensimulasikan kemampuan otak manusia untuk belajar. Jaringan syaraf tiruan juga tersusun atas neuron-neuron dan dendrit. Tidak seperti model biologis, jaringan syaraf memiliki struktur yang tidak dapat diubah, dibangun oleh sejumlah neuron dan memiliki nilai tertentu yang menunjukkan seberapa besar koneksi antar neuron (yang dikenal dengan nama bobot). Pengubahan yang terjadi selama proses pembelajaran adalah pengubahan nilai bobot. Nilai bobot akan bertambah jika informasi yang diberikan oleh neuron yang bersangkutan tersampaikan, sebaliknya jika informasi tidak disampaikan oleh neuron ke neuron yang lain, maka nilai bobot yang menghubungkan keduanya akan dikurangi. Pada saat pembelajaran dilakukan pada input yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang. Apabila nilai ini telah tercapai, mengindikasikan bahwa tiap-tiap input telah berhubungan dengan output yang diharapkan.

Ada dua jenis proses pembelajaran yaitu (Kusumadewi, 2003):

a. Pembelajaran terawasi (*supervised learning*)
   Metode pembelajaran pada jaringan syaraf disebut terawasi jika target output yang diharapkan telah diketahui sebelumnya.
b. Pembelajaran tak terawasi (*unsupervised learning*)
   Pada metode pembelajaran yang tak terawasi ini tidak memerlukan target output. Pada metode ini tidak dapat ditentukan hasil seperti yang diharapkan selama proses pembelajaran.

Di dalam proses pembelajaran, *learning rate* adalah sebuah variabel yang nilainya diantara 0 sampai 1 yang berguna untuk menentukan laju pembelajaran. Sedangkan *max epoch* adalah batas maksimum perulangan untuk melakukan proses pembelajaran.

2.4 *Learning Vector Quantization*

Learning Vector Quantization (*LVQ*) merupakan salah satu metode dalam jaringan syaraf tiruan yang telah digunakan secara luas dalam proses pengenalan pola. *LVQ* pertama kali diperkenalkan oleh Teuvo Kohonen sebagai algoritma *clustering* yang didasarkan pada sekumpulan prototipe. Pemikiran tersebut dikombinasikan dengan *self-organizing learning* namun menggunakan pelatihan yang terawasi. Kegunaan metode *LVQ* sering dijumpai pada aplikasi-aplikasi *data-mining*, robotik, maupun pengenalan linguistik (Kohonen, dkk., 1996).

LVQ merupakan metode yang mudah dipahami dan diterapkan. Penentuan klasifikasi suatu dari sekumpulan vektor dilakukan dengan membandingkan jarak *Euclidean* vektor-vektor tersebut dengan sejumlah vektor-vektor prototipe yang merepresentasikan kelas dari sekumpulan data dalam klasifikasi yang sama.

LVQ adalah suatu metode untuk melakukan pembelajaran pada lapisan kompetitif yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor input. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor input. Jika 2 vektor input mendekati sama maka lapisan kompetitif akan meletakkan kedua vektor input tersebut ke dalam kelas yang sama (Kusumadewi, 2003).

Pada proses pembelajaran, suatu vektor prototipe akan mengalami pengubahan yang akan semakin memperjelas klasifikasi. Adapun vektor prototipe yang mengalami pengubahan tersebut adalah vektor

pemenang, yaitu vektor prototipe yang terdekat (jarak *Euclidean*-nya) terhadap vektor input. Dalam hal ini berlaku skema *Winner Takes All (WTA)*, karena pengubahan signifikan hanya dialami oleh vektor prototipe pemenang dengan jarak *Euclidean* terkecil (Biehl, 2005).

2.4.1 *LVQ* 1

LVQ 1 ini merupakan algoritma dasar *LVQ*. Inti dari algoritma ini adalah pelatihan yang bertujuan untuk mengklasifikasikan vektor input ke dalam kelas prototipe yang tersedia.

Proses pembelajaran pada algoritma *LVQ* 1 diawali dengan inisialisasi secara acak untuk memilih vektor-vektor yang berlaku sebagai prototipe. Kemudian dilakukan perulangan hingga ditemukan kondisi berhenti, di mana dalam setiap perulangan dilakukan penentuan prototipe yang terdekat dari vektor input. Kemudian vektor prototipe tersebut akan mengalami pengubahan yang bersifat mendekati atau menjauh dari vektor input, tergantung pada kecocokan terhadap target pelatihan. Apabila target pelatihan sesuai dengan vektor prototipe terdekat dari vektor input, maka vektor prototipe akan berubah mendekati vektor input. Sebaliknya apabila target pelatihan tidak sesuai dengan vektor prototipe terdekat dari vektor input, maka vektor prototipe akan berubah menjauhi vektor input. Penentuan kedekatan antara vektor input ke vektor prototipe ditentukan dengan menggunakan jarak *Euclidean.*

$$c = \arg\min \{|| x - m_i ||\} \tag{2.2}$$

$$d = \sum_{i=1}^{n} | x_i - w_i | \tag{2.3}$$

$m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$ (2.4)
Jika x dan m_c termasuk ke dalam kelas yang sama

$m_c(t+1) = m_c(t) - \alpha(t)[x(t) - m_c(t)]$ (2.5)
Jika x dan m_c tidak termasuk ke dalam kelas yang sama

$m_i(t+1) = m_i(t)$ (2.6)
Untuk vektor-vektor lain di mana $i \neq c$.

Dimisalkan terdapat sejumlah i vektor prototipe, m_i (disebut juga *codebook vector*). Vektor m_c merupakan vektor prototipe dengan jarak *Euclidean d* terkecil, di mana *c* didapatkan dari persamaan 2.2, dan jarak *Euclidean d* didapat dengan persamaan 2.3. Pada persamaan 2.4 dan 2.5, m_c merupakan vektor prototype yang terdekat dengan vektor input *x*. Sehingga m_c diperbarui (*update*) sebelum digunakan pada perulangan berikutnya. Besarnya *update* yang dialami vektor m_c tergantung pada nilai laju pelatihan α. Nilai laju pelatihan α dapat berupa konstanta ataupun variabel yang berubah secara monoton terhadap iterasi (Kohonen, 1995).

2.4.2 Algoritma *LVQ*

Langkah 1 : Inisialisasi vektor referensi dan *learning rate*, $\alpha(0)$

Langkah 2 : Selama kondisi berhenti bernilai salah,

a. Untuk masing-masing pelatihan vektor input x
b. Temukan j sehingga $\| x - w_j \|$ bernilai minimum
c. Perbaiki w_j dengan:
    1. Jika $T = c_j$ maka
    $$w_j(baru) = w_j(lama) + \alpha[x - w_j(lama)] \quad (2.7)$$
    2. Jika $T \neq c_j$ maka
    $$w_j(baru) = w_j(lama) - \alpha[x - w_j(lama)] \quad (2.8)$$
d. Kurangi *learning rate*
e. Tes kondisi berhenti.

Keterangan:

x : vektor pelatihan $(x_1, \ldots, x_i, \ldots, x_n)$

T : kategori atau kelas yang benar untuk vektor pelatihan

w_j : vektor bobot untuk unit output j $(w_{1j}, \ldots, w_{ij}, \ldots, x_{nj})$

c_j : kategori atau kelas yang direpresentasikan oleh unit output j

$\| x - w_j \|$: jarak *eucledian* antara vektor input (vektor bobot) dan unit output j.

(Kristanto, 2004)

2.4.2.1 Contoh Penghitungan Manual Algoritma *LVQ*

Ini adalah contoh yang paling sederhana dari penggunaan *LVQ* dimana ada 2 vektor referensi akan digunakan. Vektor input merepresentasikan 2 kelas, yaitu 1 dan 2.

5 vektor dan kelasnya adalah sebagai berikut :

Vektor	Kelas
(1,1,0,0)	1
(0,0,0,1)	2
(0,0,1,1)	2
(1,0,0,0)	1
(0,1,1,0)	2

Dari vektor input diatas, 2 vektor pertama akan digunakan untuk menginisialisasi 2 vektor referensi. Unit output pertama merepresentasikan sebagai kelas 1 dan yang kedua sebagai kelas 2. $C_1 = 1$ dan $C_2 = 2$.

3 vektor input terakhir yaitu (0,0,1,1), (1,0,0,0), (0,1,1,0) dinyatakan sebagai vektor pelatihan.

Hanya iterasi pertama yang akan ditunjukkan pada contoh ini. Untuk iterasi selanjutnya dapat dicoba dengan cara yang sama.

Langkah 1 : Inisialisasikan bobot
$W_1 = (1,1,0,0)$
$W_2 = (0,0,0,1)$
Inisialisasikan *learning rate* $\alpha = 0,1$.

Langkah 2 : Mulai perhitungan

a. Untuk input x = (0,0,1,1) dengan T = 2
b. J = 2, x lebih dekat ke W_2 daripada W_1
c. Karena T = 2 dan $C_2 = 2$, perbaiki W_2
$W_2 = (0,0,0,1) + 0,1[(0,0,1,1) – (0,0,0,1)]$
$W_2 = (0,0,0.1,1)$

Langkah 2 : Mulai perhitungan

a. Untuk input x = (1,0,0,0) dengan T = 1
b. J = 1, x lebih dekat ke W_1 daripada W_2
c. Karena T = 1 dan C_2 = 1, perbaiki W_1
   W_1 = (1,1,0,0) + 0,1[(1,0,0,0) – (1,1,0,0)]
   W_1 = (1,0.9,0,0)

Langkah 2 : Mulai perhitungan

a. Untuk input x = (0,1,1,0) dengan T = 2
b. J = 1, x lebih dekat ke W_1 daripada W_2
c. Karena T = 2 tetapi C_1 = 1, perbaiki W_1
   W_1 = (1,0.9,0,0) - 0,1[(0,1,1,0) – (1,0.9,0,0)]
   W_2 = (1.1,0.89,-0.1,0)

Langkah 3 : Untuk iterasi pertama sudah selesai dan bisa dilanjutkan iterasi berikutnya.
Kurangi *learning rate*nya

Langkah 4 : Tes kondisi berhenti.

(Kristanto, 2004)

2.4.3 Kondisi Berhenti

Seringkali dalam praktek terjadi terlalu banyak pembelajaran yang dilakukan. Hal ini tidak selamanya berpengaruh baik terhadap akurasi pengenalan. Ketika pengulangan yang terjadi pada pembelajaran mencapai suatu titik tertentu, maka akurasi yang didapatkan mencapai tingkat optimal. Apabila perulangan pembelajaran tetap dilanjutkan melewati titik optimal tersebut, maka akurasi pengenalan akan mengalami penurunan akurasi secara lambat. Untuk mendapatkan titik di mana akurasi mencapai tingkat optimal, sangat tergantung pada kasus dan arsitektur jaringan syaraf. Oleh karena itu titik optimal tersebut hanya dapat ditentukan dari hasil percobaan (Kohonen, 1995).

2.5 Pengukuran Keakurasian

Metode Q3 digunakan untuk mengukur presentasi dari residu yang diidentifikasikan dengan benar untuk semua kelas sruktur sekunder (H, E dan C). Q3 adalah metode yang paling sederhana dalam mengukur keakurasian prediksi. Terdapat metode untuk mengukur keakurasian dari setiap kelas struktur sekunder (H, E dan

C). Metode ini dituliskan pada persamaan (2.9), (2.10) dan (2.11). Sedangkan pengukuran Q3 didefinisikan pada persamaan (2.12).

$$Q\,h = \frac{P_h}{n_h} \tag{2.9}$$

$$Q\,e = \frac{P_e}{n_e} \tag{2.10}$$

$$Q\,c = \frac{P_c}{n_c} \tag{2.11}$$

$$Q\,3 = \frac{P_h + P_e + P_c}{N} \tag{2.12}$$

(Clayton,2006)

Keterangan:

Q {h,e,c}	: Tingkat keakurasian perkelas struktur sekunder
p {h,e,c}	: Jumlah residu asam amino yang benar terprediksikan pada setiap jenis struktur sekunder (alfa heliks, *beta sheet* dan *random coil*)
n {h,e,c}	: Total jumlah residu (untuk masing-masing alfa heliks, *beta pleated sheet* dan *random coil*)
N	: Jumlah seluruh residu
Q3	: Tingkat keakurasian pada suatu rangkaian.

Untuk menghitung rata-rata tingkat keakurasian dari beberapa nilai Q3 digunakan metode Q3*.

$$Q\,3^{*} = \frac{\sum_{i=1}^{N} Q3_i}{N} \tag{2.13}$$

(Clayton,2006)

Keterangan:

Q3*	: Nilai rata-rata dari seluruh Q3
Q3	: Tingkat keakurasian pada suatu rangkaian asam amino
i	: Indeks Q3
N	: Jumlah data rangkaian asam amino

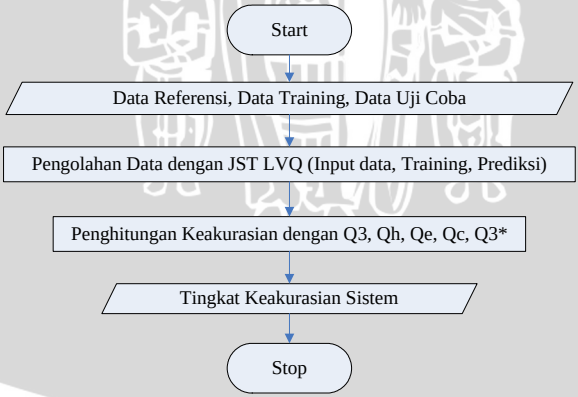
BAB III
METODOLOGI DAN PERANCANGAN SISTEM

3.1 Sistem Prediksi Struktur Sekunder Protein

Sistem prediksi struktur sekunder protein menggunakan *LVQ* adalah sistem yang memproses suatu rangkaian asam amino untuk menentukan bentuk struktur sekunder protein dengan menggunakan algoritma *Learning Vector Quantization.* Input dari sistem ini adalah rangkaian asam amino yang akan diprediksi bentuk struktur sekundernya, sedangkan outputnya adalah prediksi bagaimana bentuk struktur sekundernya. Prediksi struktur sekunder protein dengan algoritma LVQ terdiri dari beberapa proses, yaitu penginisialisasian data, pembelajaran, uji coba prediksi dan penghitungan keakurasian.

Proses inisialisasi data bertujuan untuk memberikan nilai awal pada sistem, yaitu penginputan *codebook vector* pada *LVQ* sehingga menjadi nilai acuan untuk memprediksi rangkaian asam amino. Sedangkan proses pembelajaran ini bertujuan untuk pembelajaran sistem yang diharapkan dapat meningkatkan keakurasian hasil dari prediksi yang dikeluarkan oleh sistem.

Sedangkan proses uji coba prediksi akan menghasilkan prediksi bentuk struktur sekunder dari rangkaian asam amino yang diinputkan. Hasilnya akan dianalisa dengan menggunakan metode Qh, Qe, Qc, Q3 dan Q3* untuk mengukur tingkat keakurasian sistem.



Gambar 3.1 Flowchart Prediksi Struktur Sekunder Protein dengan *LVQ*

3.1.1 Data

Data yang digunakan dalam penelitian ini berasal dari RSCB Protein Data Bank. Data terbagi menjadi 3 bagian, yaitu data referensi, data pembelajaran dan data ujicoba. Identitas data rangkaian asam amino penyusun suatu protein dirujukkan pada kode PDB dari RSCB Protein Data Bank.

3.1.1.1 Data Referensi

Data referensi berfungsi untuk memberikan *codebook vector* pada *LVQ* sebagai nilai acuan pada sistem untuk mengenali suatu pola inputan. Nilai dari data referensi ini akan berubah selama proses pembelajaran berlangsung. Rangkaian asam amino yang digunakan sebagai data referensi dilampirkan pada lampiran 1.

3.1.1.2 Data Pembelajaran

Data pembelajaran berfungsi untuk mengembangkan kemampuan prediksi sistem, sehingga diharapkan dapat memberikan prediksi yang lebih akurat. Rangkaian asam amino yang digunakan sebagai data pembelajaran dilampirkan pada lampiran 2.

3.1.1.3 Data Uji Coba

Data uji coba digunakan untuk menguji bagaimana sistem bekerja. Hasil dari prediksi data ujicoba akan dihitung keakuratannya dengan menggunakan metode Qh, Qe, Qc, Q3 dan Q3*. Rangkaian asam amino yang digunakan sebagai data ujicoba dilampirkan pada lampiran 3.

3.1.2 Proses Dalam Sistem

Di dalam sistem prediksi protein, terdapat beberapa proses yaitu penginisialisasian data, proses pembelajaran serta proses prediksi struktur sekunder. Didalam penginisialisasian data terdapat 2 proses, yaitu proses pemartisian rangkaian asam amino dan proses pengkodean asam amino.

3.1.2.1 Penginisialisasian Data

Penginisialisasian data merupakan penggabungan proses pemartisian rangkaian asam amino dan proses pengkodean residu asam amino. Hasil yang didapat akan dimasukkan dalam file sebagai *codebook vector*. File data akan menggunakan tipe .txt.

a. Proses Partisi Rangkaian Asam Amino

Proses pemartisian rangkaian asam amino pada dasarnya adalah membagi rangkaian asam amino yang menjadi inputan ke dalam beberapa *codebook vector*. Di dalam *codebook vector* tersebut, terdapat 5 residu asam amino yang terdapat pada rangkaian asam amino tersebut. Misalkan dalam sebuah rangkaian asam amino terdapat 11 residu asam amino, maka pemartisian rangkaian akan bergerak mulai dari residu pertama. Dari residu pertama, akan dimasukkan ke dalam *codebook vector* residu 1,2,3,4 dan 5. Kemudian, dilakukan pemartisian lagi, dimulai dari residu kedua,sehingga, residu asam amino yang dimasukkan ke dalam *codebook vector* adalah residu ke 2,3,4,5 dan 6. Proses ini dilakukan secara berulang sampai residu ke n-4. Jika terdapat 11 residu asam amino dalam sebuah rangkaian, maka jumlah pemartisian rangkaian asam amino adalah sebanyak 11 - 4 = 7 *codebook vector*. Yaitu *codebook vector* yang berisi residu ke 1,2,3,4,5; 2,3,4,5,6; 3,4,5,6,7; 4,5,6,7,8; 5,6,7,8,9; 6,7,8,9,10; 7,8,9,10,11.

b. Pengkodean Residu Asam Amino

Untuk setiap data penelitian, yaitu data referensi, data pembelajaran dan data ujicoba akan dilakukan proses pengkodean, yaitu pengubahan data dari bentuk karakter menjadi bentuk bilangan desimal dengan rentang antara 0,05 sampai dengan 0,95. Dari hasil pengkodean data tersebut, akan diletakkan pada suatu file untuk dijadikan suatu kelas di *LVQ*.

Persamaan yang digunakan untuk mengkodekan residu asam amino berasal dari persamaan (2.1). Maka variabel pada persamaan 2.1 berubah dalam penggunaannya, yaitu :

x’ = hasil pengkodean residu asam amino
x = nilai desimal bilangan ASCII karakter residu asam amino

min values = nilai desimal bilangan ASCII karakter A
max values = nilai desimal bilangan ASCII karakter Z

Pada tabel 3.1 akan dicantumkan bilangan-bilangan ASCII untuk karakter A sampai dengan Z.

Tabel 3.1 Daftar Bilangan ASCII 7-bit (Hartono, 1999)

Binari	Desimal	Karakter	Tampak Di layar	Keterangan
1000001	65	A	A	Upper case A
1000010	66	B	B	Upper case B
1000011	67	C	C	Upper case C
1000100	68	D	D	Upper case D
1000101	69	E	E	Upper case E
1000110	70	F	F	Upper case F
1000111	71	G	G	Upper case G
1001000	72	H	H	Upper case H
1001001	73	I	I	Upper case I
1001010	74	J	J	Upper case J
1001011	75	K	K	Upper case K
1001100	76	L	L	Upper case L
1001101	77	M	M	Upper case M
1001110	78	N	N	Upper case N
1001111	79	O	O	Upper case O
1010000	80	P	P	Upper case P
1010001	81	Q	Q	Upper case Q
1010010	82	R	R	Upper case R
1010011	83	S	S	Upper case S
1010100	84	T	T	Upper case T
1010101	85	U	U	Upper case U
1010110	86	V	V	Upper case V
1010111	87	W	W	Upper case W
1011000	88	X	X	Upper case X
1011001	89	Y	Y	Upper case Y
1011010	90	Z	Z	Upper case Z

3.1.2.2 Pembelajaran Sistem

Dalam pembelajaran (*training*), sistem akan diinputkan beberapa *codebook vector* dari rangkaian asam amino yang berbeda dari rangkaian asam amino data referensi. Rangkaian asam amino data pembelajaran juga telah mengalami proses pemartisian rangkaian asam amino, peng-*encoding*-an asam amino serta pereduksian. Sistem akan di-*training* sebanyak 100 kali dengan *learning rate* sebesar 0,01.

3.1.2.3 Prediksi Struktur Sekunder Protein

Proses ini akan memprediksikan bagaimana bentuk struktur sekunder dari rangkaian asam amino yang menyusun suatu protein. Inputan dari proses ini adalah suatu *string* rangkaian asam amino yang menghasilkan output berupa *string* prediksi bentuk struktur sekunder yang merupakan kombinasi dari huruf H, E, dan C. Kombinasi H, E dan C merupakan bentuk dari pereduksian bentuk struktur sekunder sesuai dengan skema reduksi 8-to-3 state di tabel 2.2.

3.2 Rancangan Program

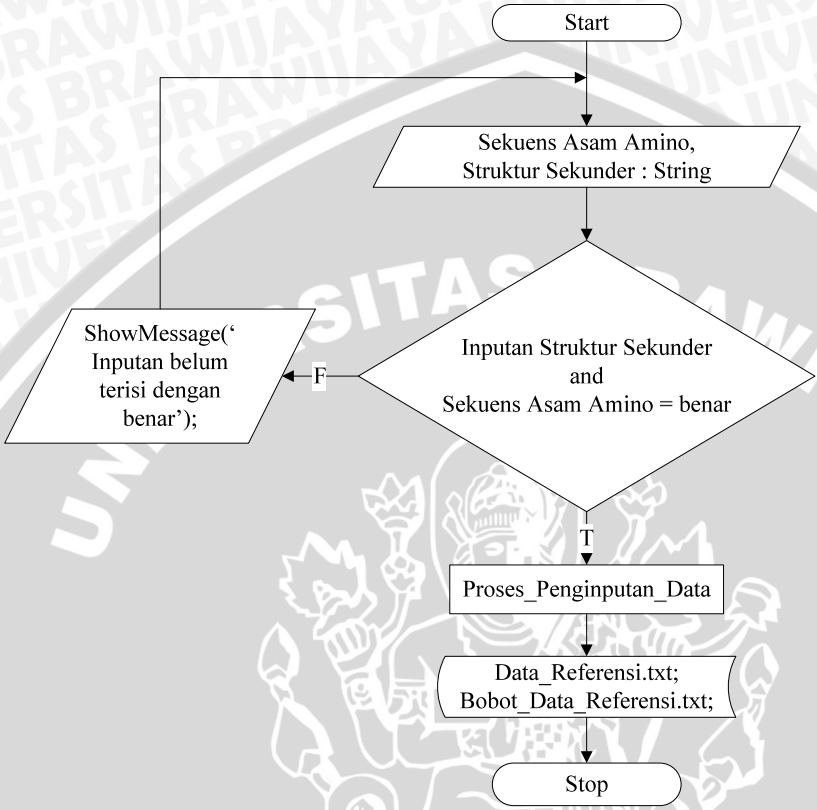
3.2.1 *Flowchart* Sistem

3.2.1.1 Penginisialisasian Data

Pada prosedur penginisialisasian data, terdapat 4 macam data yang akan diinputkan. Yaitu, data referensi, data pembelajaran, *learning rate* dan *max epoch*.

a. Data Referensi

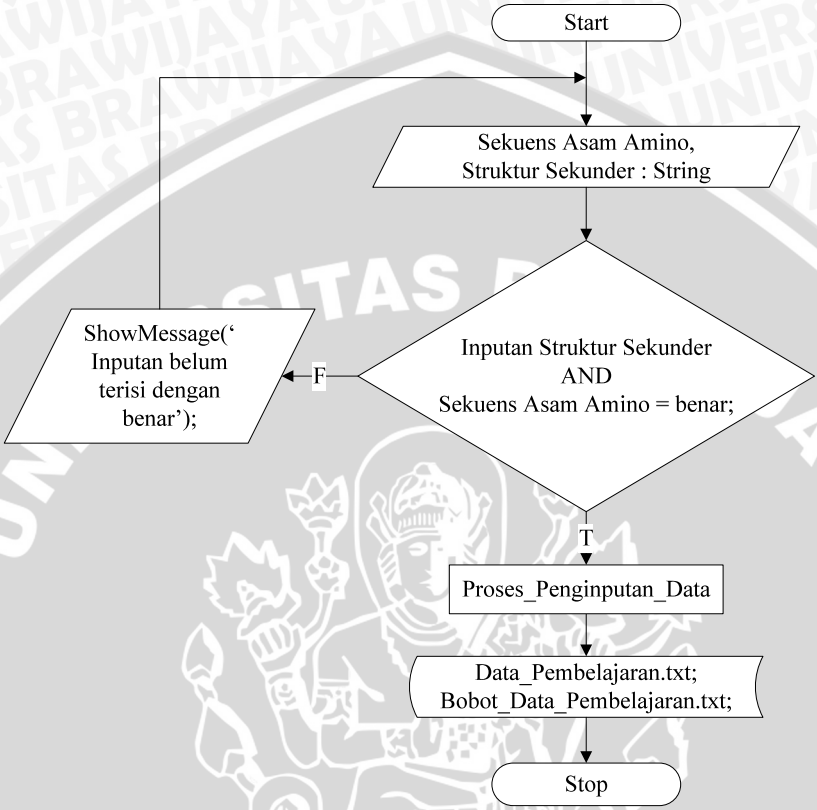
Pada penginisialisasian data referensi mempunyai inputan rangkaian asam amino dan bentuk struktur sekundernya yang masing-masing berbentuk *string*. Ouput dari prosedur ini adalah file yang bertipe .txt, yang berisi residu asam amino serta bobot yang merupakan hasil pengkodeannya. *Flowchart* penginisialisasian data referensi ditunjukkan pada gambar 3.2.



Gambar 3.2 *Flowchart* Penginisialisasian Data Referensi

b. Data Pembelajaran

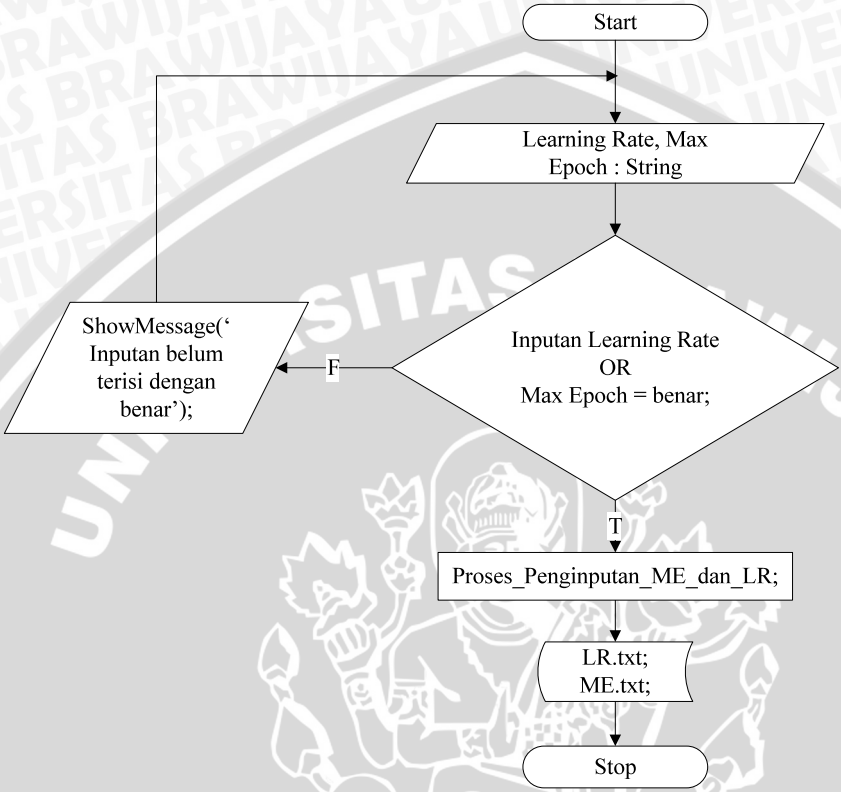
Pada penginisialisasian data pembelajaran, mempunyai inputan berupa rangkaian asam amino dan bentuk struktur sekunder dengan tipe data *string*. Ouput dari prosedur ini adalah file yang bertipe .txt berisi residu asam amino serta bobot yang merupakan hasil pengkodeannya. *Flowchart* penginisialisasian data pembelajaran ditunjukkan pada gambar 3.3.



Gambar 3.3 *Flowchart* Penginisialisasian Data Pembelajaran

c. *Learning Rate* dan *Max Epoch*

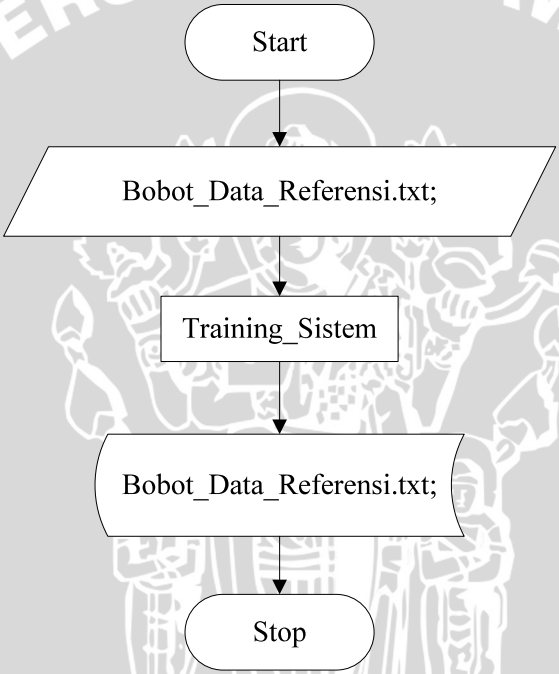
Pada penginisialisasian *learning rate* dan *max epoch* mempunyai inputan *learning rate* dan *max epoch* yang masing-masing berbentuk *string*. Ouput dari prosedur ini adalah file yang bertipe .txt, yang berisi nilai *learning rate* dan *max epoch* yang diinputkan. *Flowchart* penginisialisasian *learning rate* dan *max epoch* ditunjukkan pada gambar 3.4.



Gambar 3.4 *Flowchart* Penginisialisasian *Learning Rate* dan *Max Epoch*

3.2.1.2 Pembelajaran Sistem

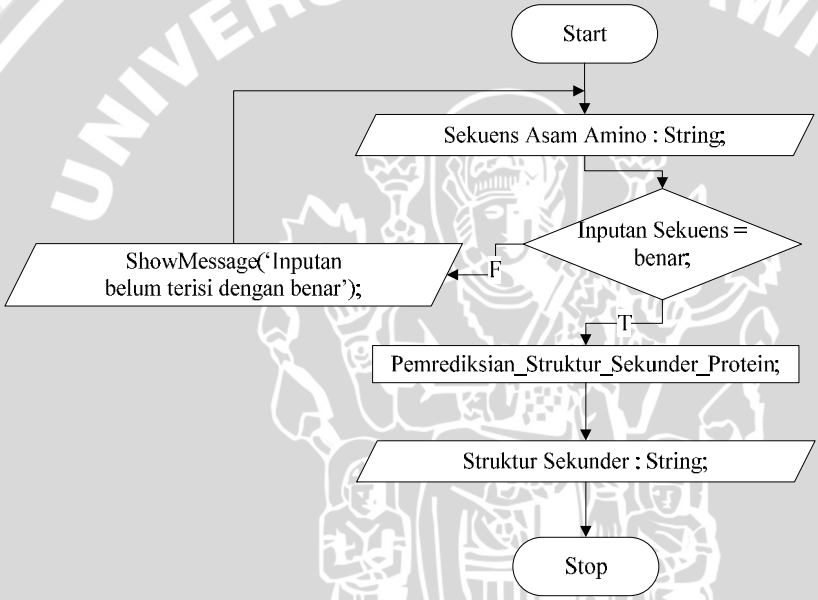
Prosedur pembelajaran sistem digunakan untuk mengubah nilai pada bobot data referensi. Pengubahan bobot data referensi, mengacu pada jarak *Euclidean* terhadap bobot data pembelajaran. Dari pengubahan bobot data referensi, berakibat terhadap keakurasian dari prediksi struktur sekunder. *Flowchart* pembelajaran sistem ditunjukkan pada gambar 3.5.



Gambar 3.5 *Flowchart* Pembelajaran Sistem

3.2.1.3 Prediksi Struktur Sekunder Protein

Prosedur prediksi struktur sekunder digunakan untuk memprediksikan bagaimana bentuk struktur sekunder dari rangkaian asam amino yang diinputkan. Hasil prediksi sistem akan dianalisa dengan dihitung tingkat keakurasiaannya menggunakan metode Q3. Qh, Qe, dan Qc. Untuk menghitung keakurasian sistem, menggunakan Q3*. *Flowchart* prediksi struktur sekunder protein ditunjukkan pada gambar 3.6.



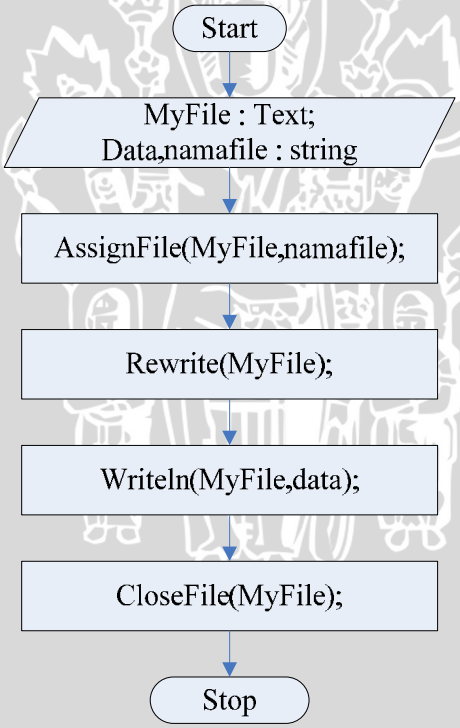
Gambar 3.6 *Flowchart* Proses Prediksi Struktur Sekunder Protein

3.2.1.4 Prosedur Lain

Disamping prosedur utama, terdapat prosedur penunjang yang digunakan dalam sistem ini. Antara lain, prosedur untuk membuat file baru, prosedur untuk meng-load nilai dalam suatu file yang bertipe integer, prosedur untuk meng-load nilai dalam suatu file yang bertipe real serta prosedur pengkodean asam amino.

a. Prosedur Buat_File_Baru

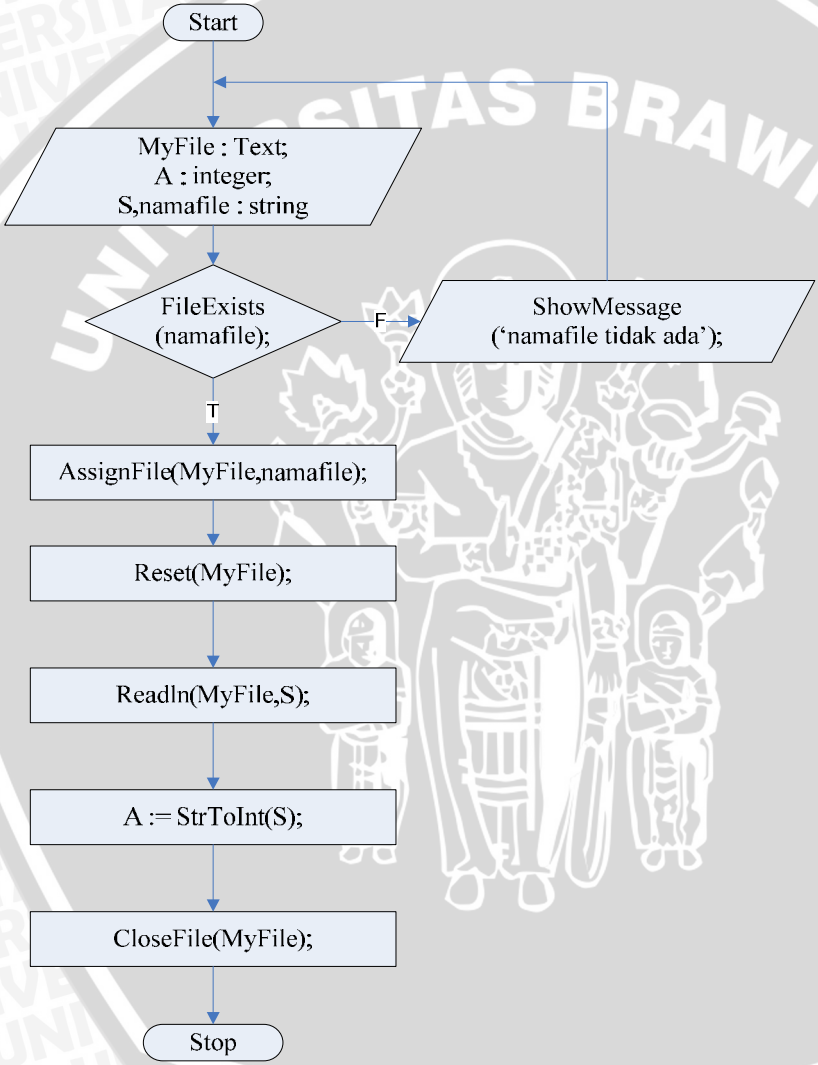
Prosedur buat_file_baru digunakan untuk membuat file. File yang dibuat menggunakan prosedur ini. File yang telah dibuat digunakan untuk proses lain dalam sistem prediksi struktur sekunder. *Flowchart* prosedur Buat_File_Baru ditunjukkan pada gambar 3.7.



Gambar 3.7 *Flowchart* Prosedur Buat_File_Baru

b. Prosedur Loadfile_integer

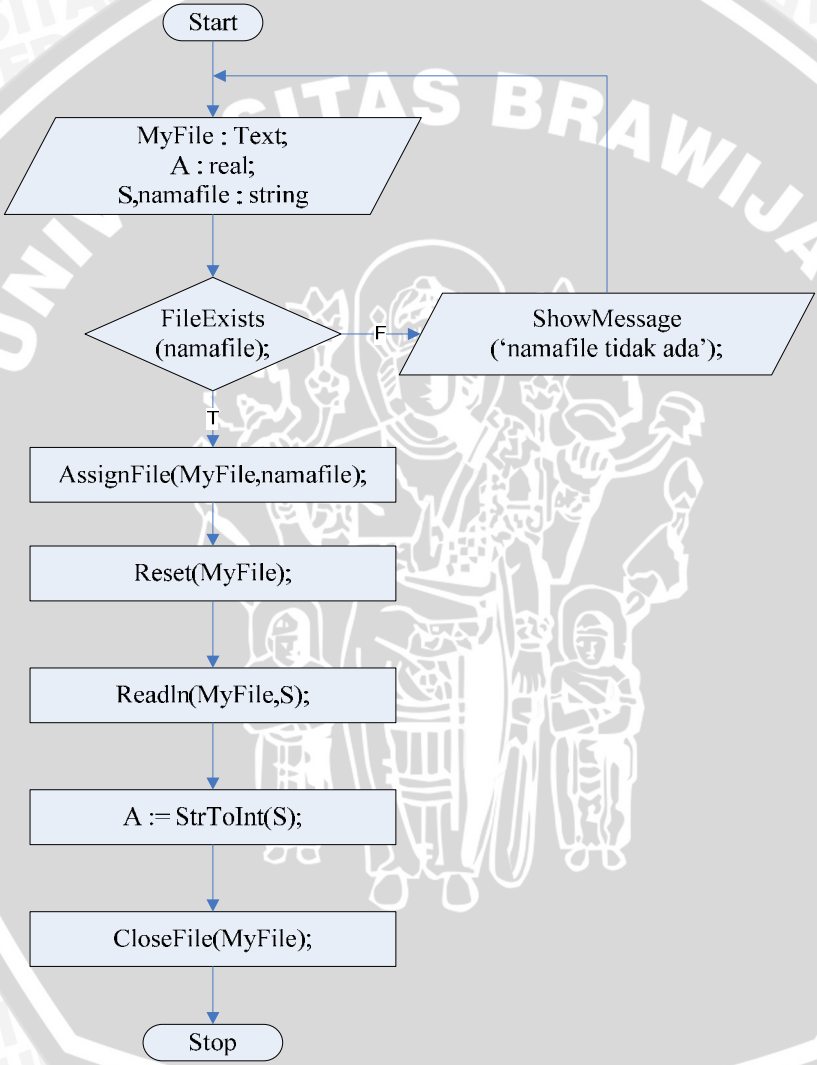
Prosedur loadfile_integer digunakan untuk me-*load* nilai pada suatu file yang kemudian dijadikan nilai yang bertipe integer. *Flowchart* prosedur loadfile_integer ditunjukkan pada gambar 3.8.



Gambar 3.8 *Flowchart* Prosedur Loadfile_Integer

c. Prosedur Loadfile_real

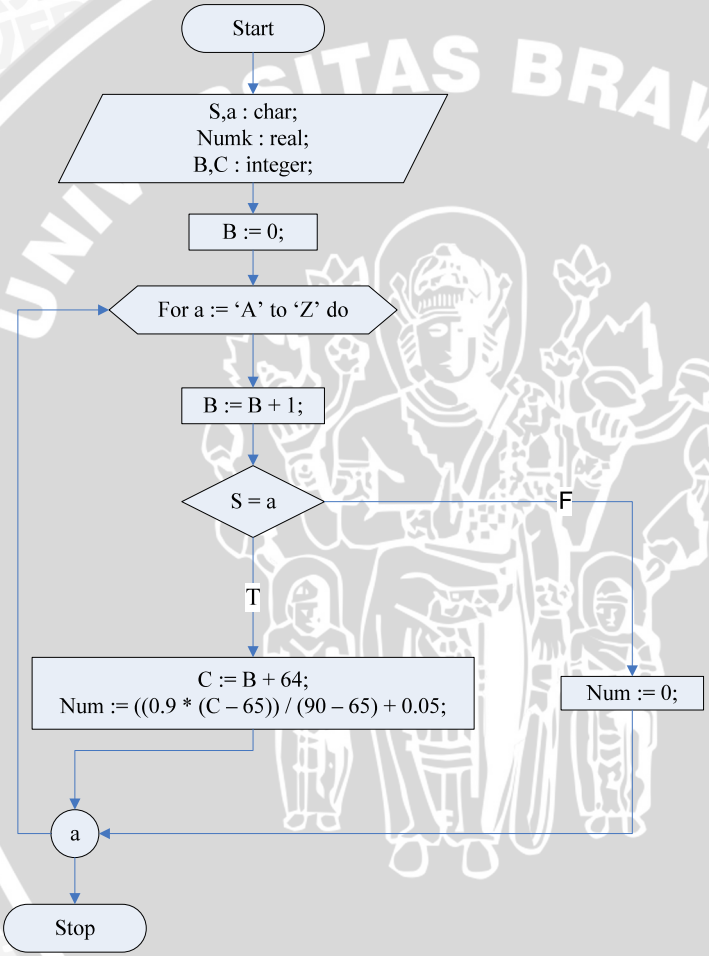
Prosedur loadfile_real digunakan untuk me-*load* nilai pada suatu file yang kemudian dijadikan nilai yang bertipe real. *Flowchart* prosedur loadfile_real ditunjukkan pada gambar 3.9



Gambar 3.9 *Flowchart* Prosedur Loadfile_Real

d. Prosedur Encoding_Asam_Amino

Prosedur Encoding_Asam_Amino digunakan untuk mengkodekan residu asam amino yang bertipe data karakter menjadi data yang bertipe numerik. *Flowchart* prosedur encoding_asam_amino ditunjukkan pada gambar 3.10.



Gambar 3.10 *Flowchart* Prosedur Encoding_Asam_Amino

3.2.2 Struktur Data

3.2.2.1 *Array* Rangkaian Asam Amino

Rangkaian asam amino akan diletakkan pada sebuah *array* yang bertipekan karakter. Karakter yang diinputkan hanyalah karakter yang sesuai dengan singkatan dari 20 jenis asam amino. Sehingga untuk karakter B, J, U, X, Z, b, j, u, x, dan z serta karakter tanda baca tidak dapat di masukkan ke dalam *array* ini. *Array* ini akan diolah untuk proses selanjutnya, seperti diinputkan ke dalam *record*. *Array* rangkaian asam amino ditunjukkan pada gambar 3.11.

Residu ke 1 : char	Residu ke 2 : char	.....	.....	Residu ke n : char

Gambar 3.11 *Array* Rangkaian Asam Amino

3.2.2.2 *Record* Residu Asam Amino

Record residu asam amino, digunakan untuk meletakkan residu asam amino yang berasal dari *array* rangkaian asam amino. *Record* residu asam amino berisi 5 residu asam amino, beserta nomor indeks dan bentuk struktur sekundernya. *Record* Residu Asam Amino, ditunjukkan pada gambar 3.12

Nomor Indeks Vektor : Integer;
Struktur Sekunder : *String*;
Residu ke 1 : *Char*;
Residu ke 2 : *Char*;
Residu ke 3 : *Char*;
Residu ke 4 : *Char*;
Residu ke 5 : *Char*;

Gambar 3.12 *Record* Residu Asam Amino

3.2.2.3 *Record* Bobot Residu Asam Amino

Record bobot residu asam amino, digunakan untuk meletakkan hasil pengkodean residu asam amino dari *array* rangkaian asam amino. *Record* bobot residu asam amino, mempunyai struktur nomor indeks, bentuk struktur sekunder serta hasil pengkodean residu asam

amino ke 1 sampai dengan ke 5. *Record* bobot residu asam amino, ditunjukkan pada gambar 3.13

Nomor Indeks Vektor : Integer;
Struktur Sekunder : *String*;
Bobot Residu ke 1 : *Real*;
Bobot Residu ke 2 : *Real*;
Bobot Residu ke 3 : *Real*;
Bobot Residu ke 4 : *Real*;
Bobot Residu ke 5 : *Real*;

Gambar 3.13 *Record* Bobot Residu Asam Amino

3.2.2.4 *Array* Jarak *Euclidean*

Array jarak *Euclidean*, digunakan untuk meletakkan hasil penghitungan jarak *Euclidean* antara *record* bobot residu asam amino data pembelajaran dengan *record* bobot residu asam amino data refererensi. *Array* Jarak *Euclidean* disini, selain digunakan pada proses pembelajaran juga digunakan pada proses prediksi.

Pada proses pembelajaran, akan diubah nilai bobot pada *record* bobot residu asam amino data referensi yang memiliki nilai jarak *Euclidean* terkecil terhadap *record* bobot residu asam amino data pembelajaran. Jarak *Euclidean* terkecil dipilih dari *array* jarak *Euclidean* dengan melakukan pengurutan data.

Sedangkan pada proses prediksi, akan dipilih struktur sekunder pada *record* bobot residu asam amino data referensi yang memiliki nilai jarak *Euclidean* terkecil terhadap *record* bobot residu asam amino rangkaian *user*. Jarak *Euclidean* terkecil juga dipilih dari *array* jarak *Euclidean* dengan melakukan pengurutan data. *Array* jarak *Euclidan* ditunjukkan pada gambar 3.14.

Euclidean ke 1 : Real;
Euclidean ke 2 : Real;
....
....
Euclidean ke n : Real;

Gambar 3.14 *Array* Jarak *Euclidean*

3.2.3 Arsitektur Sistem

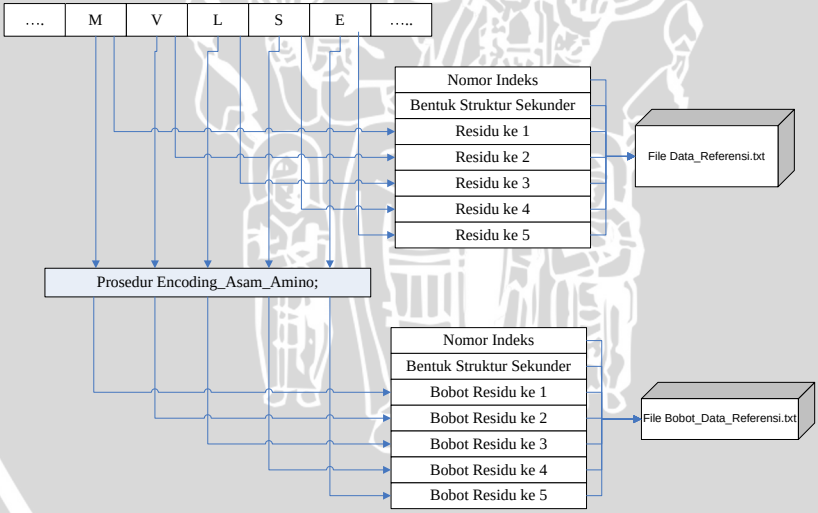
Arsitektur sistem yang akan ditunjukkan adalah arsitektur dari beberapa proses inti dari sistem prediksi struktur sekunder. Yaitu, proses inisialisasi data, pembelajaran sistem dan proses prediksi struktur sekunder

3.2.3.1 Penginisialisasian Data

a. Data Referensi

Arsitektur sistem pada penginisialisasian data untuk data referensi, ditunjukkan pada gambar 3.15. Pada gambar, terdapat *array* rangkaian asam amino, *record* residu asam amino data referensi dan *record* bobot residu asam amino data referensi.

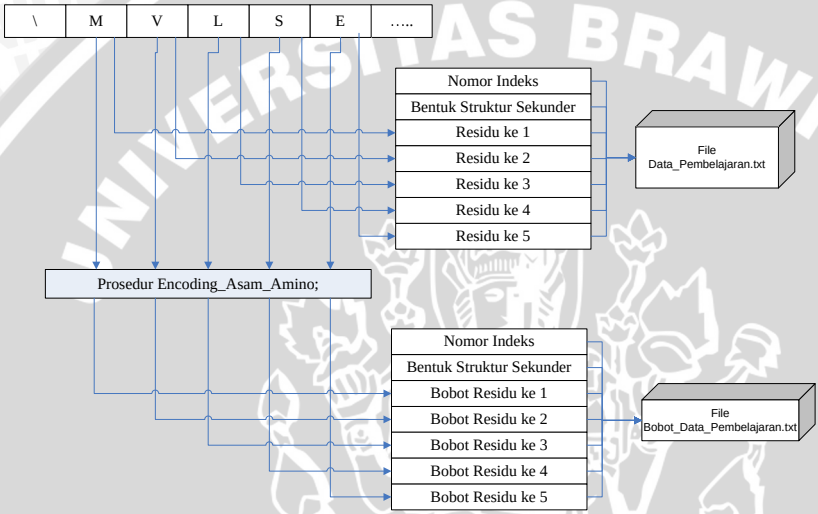
Prosedur Encoding_Asam_Amino, juga digambarkan pada arsitektur sistem ini yang ditempatkan pada pertengahan aliran data dari *array* rangkaian asam amino menuju *record* bobot residu asam amino data referensi.



Gambar 3.15 Arsitektur Sistem pada Proses Inisialisasi Data Referensi

b. Data Pembelajaran

Arsitektur sistem pada proses inisialisasi data pembelajaran juga sama dengan data referensi. Hanya saja, akhir dari proses ini berbeda, yaitu file yang disimpan berupa file data_pembelajaran.txt dan bobot_data_pembelajaran.txt. Gambar arsitektur sistem pada penginisialisasian data pembelajaran ditunjukkan pada gambar 3.16.



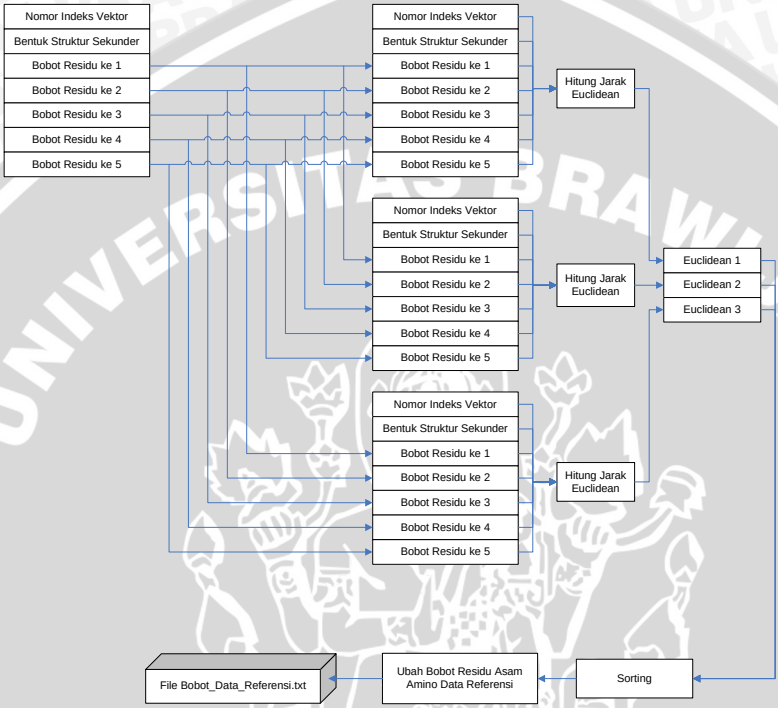
Gambar 3.16 Arsitektur Sistem pada Penginisialisasian Data Pembelajaran

3.2.3.2 Pembelajaran Sistem

Untuk pembelajaran sistem, ditunjukkan proses pencarian jarak *Euclidean* dari sebuah *record* bobot residu asam amino data pembelajaran terhadap semua *record* bobot residu asam amino data referensi. Kemudian diurutkan dan dipilih mana jarak *Euclidean* yang terkecil. Hasilnya, akan disimpan pada *array* jarak *Euclidean*.

Proses selanjutnya adalah pengubahan nilai bobot pada *record* bobot residu asam amino data referensi yang memiliki jarak *Euclidean* terkecil terhadap *record* bobot residu asam amino data pembelajaran. Hasil pengubahan bobot, akan disimpan kembali ke

dalam file bobot_data_referensi.txt. Gambar 3.17 menunjukkan arsitektur pembelajaran sistem.



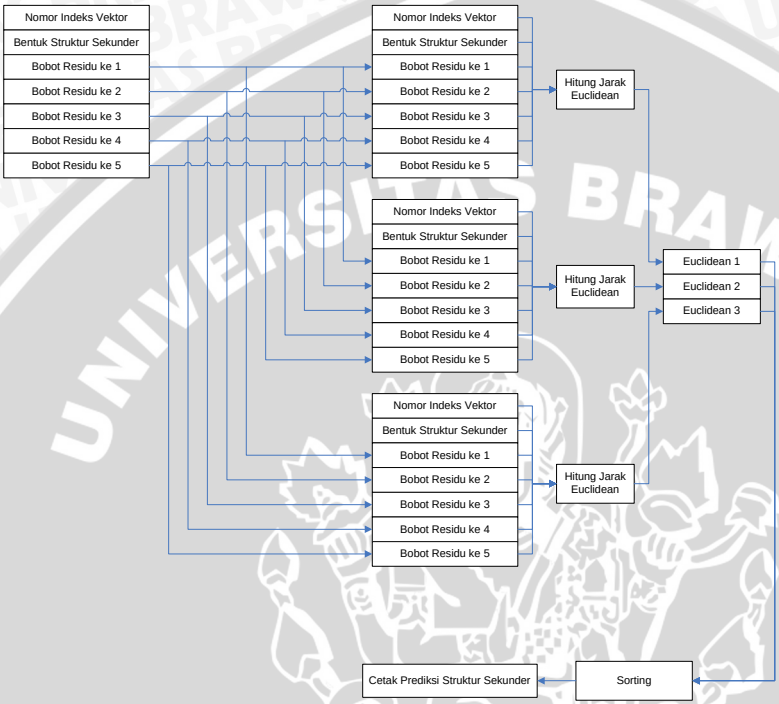
Gambar 3.17 Arsitektur Sistem pada Pembelajaran Sistem

3.2.3.3 Prediksi Struktur Sekunder Protein

Proses prediksi struktur sekunder protein hampir sama dengan pembelajaran sistem. Akan tetapi, jarak *Euclidean* didapatkan dengan membandingkan *record* bobot residu asam amino dari rangkaian asam amino *user* dengan *record* bobot residu asam amino data referensi.

Selain itu tidak dilakukan pengubahan bobot. Tetapi mengeluarkan output berupa prediksi struktur sekunder dari *record* bobot residu asam amino data referensi dengan jarak *Euclidean* terkecil terhadap *record* bobot residu asam amino rangkaian asam amino *user*.

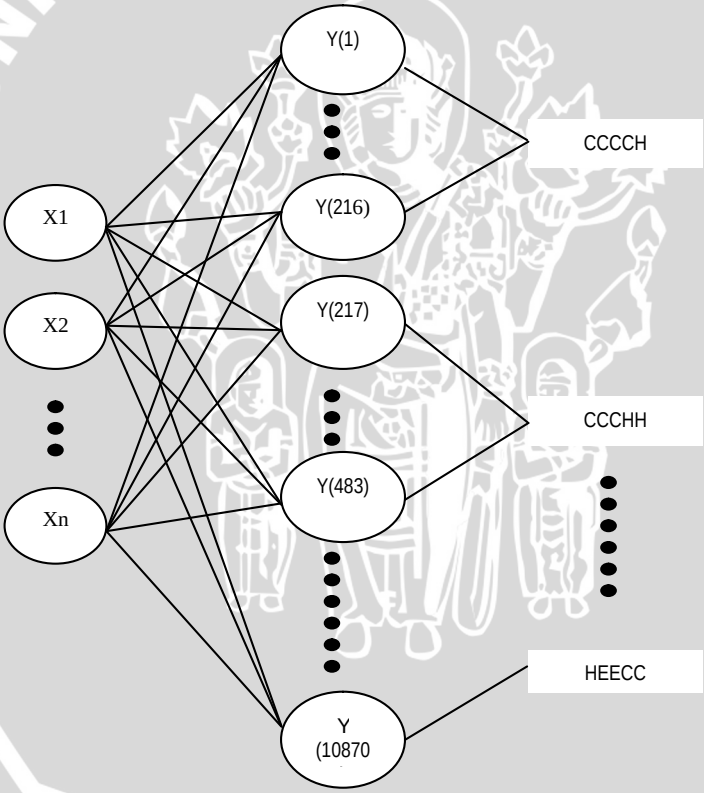
Gambar 3.18 menunjukkan bagaimana bentuk arsitektur sistem proses prediksi struktur sekunder.



Gambar 3.18 Arsitektur Sistem pada Proses Prediksi Struktur Sekunder Protein

3.2.4 Arsitektur Jaringan Syaraf Tiruan *Learning Vector Quantization*

Arsitektur dari *Learning Vector Quantization* terdiri dari lapisan masukkan dan lapisan keluaran. Pada neuron lapisan masukkan (X) akan diinputkan *codebook vector* dari rangkaian asam amino user. Sedangkan lapisan keluaran terdiri dari 10870 neuron. Nilai 10870 jumlah data yang didapat dari pemartisian data referensi. Terdapat beberapa neuron yang mewakili 1 kelas. Total kelas yang ada berjumlah 66. Perincian jumlah neuron yang mewakili pada satu kelas dapat dilihat pada tabel 3.2. Gambar arsitektur *LVQ* dapat dilihat pada gambar 3.19.



Gambar 3.19 Arsitektur Jaringan Syaraf Tiruan *LVQ*

Tabel 3.2 Kelas Pengenalan

No	Kelas	Jumlah neuron yang terhubung
1	CCCCH	216
2	CCCHH	267
3	CCHHH	329
4	CHHHH	308
5	HHHHH	2337
6	HHHHC	332
7	HHHCH	37
8	HHCHH	37
9	HCHHH	37
10	HHHCC	358
11	HHCCC	305
12	HCCCC	239
13	CCCCC	2029
14	CHHHC	68
15	HHCCH	34
16	HCCHH	34
17	CCEEE	340
18	CEEEE	312
19	EEEEE	736
20	EEEEC	293
21	EEECC	308
22	EECCC	305
23	ECCCC	253
24	CCCCE	276
25	CCCEE	338
26	EECCH	20
27	ECCHH	20
28	EECCE	52
29	ECCEE	52

No	Kelas	Jumlah neuron yang terhubung
30	EEECH	13
31	EECHH	17
32	ECHHH	17
33	ECCCH	19
34	HCCCE	29
35	EEEEH	25
36	EEEHH	31
37	EEHHH	35
38	EHHHH	24
39	EHHHC	11
40	CCEEC	67
41	CEECC	68
42	CEEEC	39
43	HCCCH	30
44	CEEEH	5
45	HHHCE	14
46	HHCEE	14
47	HCEEE	10
48	EEECE	6
49	EECEE	6
50	ECEEC	3
51	ECEEE	3
52	CEECH	4
53	HCEEC	3
54	ECCCE	31
55	CHHHE	8
56	HHHEE	8
57	HHEEE	7
58	HEEEE	6
59	HHCCE	16
60	HCCEE	16

No	Kelas	Jumlah neuron yang terhubung
61	CCEEH	3
62	CEEHH	4
63	HEEEH	1
64	HCEEH	1
65	HHEEC	1
66	HEECC	1

3.2.5 Algoritma *LVQ* Untuk Prediksi Struktur Sekunder Protein

3.2.5.1 Inisialisasi Awal

Untuk inisialisasi awal, dilakukan langkah – langkah sebagai berikut:

1. Inisialisasi *Max epoch*
2. Inisialisasi *Learning Rate*
3. Inisialisasi Data Pembelajaran (Vektor Residu Asam Amino dan Vektor Bobot Data Pembelajaran)
4. Inisialisasi Data Referensi (Vektor Residu Asam Amino dan Vektor Bobot Data Referensi)

3.2.5.2 Pembelajaran Sistem

Langkah-langkah dalam pembelajaran sistem adalah sebagai berikut:

a. Lakukan iterasi sampai kondisi berhenti terpenuhi. Pada setiap iterasi dilakukan :

Untuk setiap vektor bobot data pembelajaran x dilakukan :

1. Penghitungan jarak *Euclidean* terhadap semua vektor bobot data referensi $\| x - w_j \|$ sehingga bernilai minimum
2. Tandai/Simpan indeks sebagai *minEuc* vektor bobot data referensi yang mempunyai jarak *Euclidean* minimum terhadap vektor bobot data pembelajaran x
3. Perbaiki bobot pada vektor bobot data referensi yang mempunyai indeks *minEuc* dengan :
   Jika bentuk struktur sekunder vektor data pembelajaran x = bentuk struktur sekunder vektor data referensi, maka ubah nilai bobot dengan persamaan 2.7
   Jika tidak sama, maka ubah nilai bobot dengan persamaan 2.8
4. Kurangi *learning rate* dengan cara :
   Learning rate = Learning Rate – (0,1**Learning Rate*)

b. Tes Kondisi berhenti, apakah sudah terpenuhi atau tidak.

3.2.5.3 Prediksi Struktur Sekunder Protein

Langkah-langkah dalam memprediksi struktur sekunder protein adalah sebagai berikut :

Untuk setiap vektor bobot rangkaian asam amino *user* (x) dilakukan :

1. Penghitungan jarak *Euclidean* terhadap semua vektor bobot data referensi $\| x - w_j \|$ sehingga bernilai minimum
2. Tandai/simpan indeks sebagai *minEuc* vektor bobot data referensi yang mempunyai jarak *Euclidean* minimum terhadap vektor bobot data pembelajaran x
3. Maka bentuk struktur sekunder rangkaian asam amino *user* adalah bentuk struktur sekunder dari vektor data referensi yang mempunyai jarak *Euclidean* minimum

3.2.6 Contoh Penghitungan Manual

3.2.6.1 Pengkodean Asam Amino

Berikut ini adalah contoh bagaimana penghitungan dalam mengkodekan asam amino, dari data bertipe karakter menjadi bertipe numerik. Dimisalkan terdapat sebuah vektor A yang mempunyai data M,V,L,S,E. Langkah-langkah untuk pengkodean adalah sebagai berikut :

Misalnya untuk Huruf M :

1. Kenali bilangan ASCII dari huruf M , yaitu = 77
2. Kemudian dihitung dengan persamaan 2.1

$$x' = \frac{0.9(77-65)}{(90-65)} + 0.05$$

$$x = 0.482$$

Langkah-langkah ini juga diberlakukan terhadap huruf-huruf yang lain, yaitu V,L,S dan E.

3.2.6.2 Pembelajaran Sistem Menggunakan *LVQ*

Berikut ini adalah contoh penerapan algoritma *Learning Vector Quantization* pada proses pembelajaran. Dimisalkan terdapat dua vektor prototipe A dan B masing-masing berukuran 5, dan suatu vektor masukan *x* yang berukuran 5 pula. Langkah-langkah penghitungan dalam dua iterasi adalah sebagai berikut:

1. Ditetapkan :
   - Bobot awal (*w*) untuk unit prototipe w_1 (L, S, E, G, E) = (0.446 ; 0.698 ; 0.194 ; 0.266 ; 0.194) dan unit prototipe w_2 (S, E, G, E, W) = (0.698 ; 0.194 ; 0.266 ; 0.194 ; 0.842). Untuk unit prototipe w_1 mempunyai bentuk struktur sekunder berupa (CCHHH), sedangkan unit prototipe w_2 mempunyai bentuk struktur sekunder berupa (CHHHH).
   - *MaxEpoch*, misalkan 10000
   - *Learning Rate* (LR) : 0,1

2. Dipilih :
   - Unit input (L, S, E, G, A) = (0.446 ; 0.698 ; 0.194 ; 0.266 ; 0.05)
   - Struktur sekunder berupa (CCHHH).
   - Tetapkan kondisi awal : *Epoh* ← 0, *Eps* ← 1

3. Iterasi ke-1
   a. *Epoch* 1

   b. Untuk tiap-tiap unit output ditentukan jarak *Euclidean*-nya.

$$d_1 = \sum_{i=1}^{5} | x_i - A_i |$$

$$= |0.446 - 0.446| + |0.698 - 0.698| + |0.194 - 0.194| + |0.226 - 0.226| + |0.05 - 0.194|$$

$$= 0 + 0 + 0 + 0 + 0.144$$

$$= 0.144$$

$$d_2 = \sum_{i=1}^{5} | x_i - B_i |$$

$= |0.446 – 0.698| + |0.698 – 0.194| + |0.194 – 0.226| +$
$|0.226 – 0.194| + |0.05 – 0.842|$
$= 0.252 + 0.504 + 0.032 + 0.032 + 0.792$
$= 1.58$

d. Jarak minimum pada prototipe A, sedangkan bentuk struktur sekunder sama, sehingga
   - $A_1 = A_1 + LR\ (x_1 - A_1) = 0.446 + 0{,}1(0) = 0.446$
   - $A_2 = A_2 + LR\ (x_2 - A_2) = 0.698 + 0{,}1(0) = 0.698$
   - $A_3 = A_3 + LR\ (x_3 - A_3) = 0.194 + 0{,}1(0) = 0.194$
   - $A_4 = A_4 + LR\ (x_4 - A_4) = 0.266 + 0{,}1(0) = 0.266$
   - $A_5 = A_5 + LR\ (x_5 - A_5) = 0.194 + 0{,}1(-0.144) = 0.1826$

   Sehingga bobot A setelah iterasi ke-1 (0.446; 0.698; 0.194; 0.266; 0.1826)

e. LR = LR – 0,1(LR)
   = 0,1 – 0,1(0,1)
   = 0,09

4. Iterasi ke-2

   a. *Epoh* = 2

   b. Ditentukan :
      - *minDist* ← 10000
      - *minIdx* ← 0

   c. Untuk tiap-tiap unit output ditentukan jarak *Euclidean*-nya.

$$d_1 = \sum_{i=1}^{5} | x_i - A_i |$$

$= |0.446 – 0.446| + |0.698 – 0.698| + |0.194 – 0.194| +$
$|0.226 – 0.226| + |0.05 – 0.1826|$
$= 0 + 0 + 0 + 0 + 0.1326$
$= 0.1326$

$$d_2 = \sum_{i=1}^{5} | x_i - B_i |$$

= |0.446 – 0.698| + |0.698 – 0.194| + |0.194 – 0.226| + |0.226 – 0.194| + |0.05 – 0.842|

= 0.252 + 0.504 + 0.032 + 0.032 + 0.792

= 1.58

d. Jarak minimum pada prototipe A, sedangkan T=A sehingga :

- $A_1 = A_1 + LR\ (x_1 - A_1) = 0.446 + 0{,}1(0) = 0.446$
- $A_2 = A_2 + LR\ (x_2 - A_2) = 0.698 + 0{,}1(0) = 0.698$
- $A_3 = A_3 + LR\ (x_3 - A_3) = 0.194 + 0{,}1(0) = 0.194$
- $A_4 = A_4 + LR\ (x_4 - A_4) = 0.266 + 0{,}1(0) = 0.266$
- $A_5 = A_5 + LR\ (x_5 - A_5) = 0.1826 + 0{,}1(-0.1326) = 0.16934$

Sehingga bobot A setelah iterasi ke-1 (0.446; 0.698; 0.194; 0.266; 0.16934)

e. LR = LR – 0,1(LR)

= 0,09 – 0,1(0,09)

= 0,081

5. Demikian seterusnya selama Epoch < MaxEpoch.

3.2.6.3 Prediksi Struktur Sekunder Protein Menggunakan *LVQ*

Berikut ini adalah contoh penerapan algoritma *Learning Vector Quantization* pada proses pengenalan. Dimisalkan terdapat dua vektor prototipe A dan B dan satu vektor masukan, dengan ukuran dan bobot sebagaimana hasil proses pembelajaran pada di atas. Langkah-langkah penghitungan dalam proses pengenalan adalah sebagai berikut:

1. Untuk tiap-tiap unit output ditentukan jarak *Euclidean*-nya.

$$d_1 = \sum_{i=1}^{5} | x_i - A_i |$$

= |0,05 - 0.446| + |0.122 - 0.698| + |0.194 - 0.194| + |0.41 – 0.266| + | 0.518 - 0.16934|

= 0.396 + 0.576 + 0 + 0.144 + 0.34866

= 1.46466

$$d_2 = \sum_{i=1}^{5} | x_i - B_i |$$

= |0,05 - 0.698| + |0.122 - 0.194| + |0.194 - 0.266| + |0.41 – 0.194| + | 0.518 – 0.842|

= 0.648 + 0.072 + 0.072 + 0.216 + 0.324

= 1.332

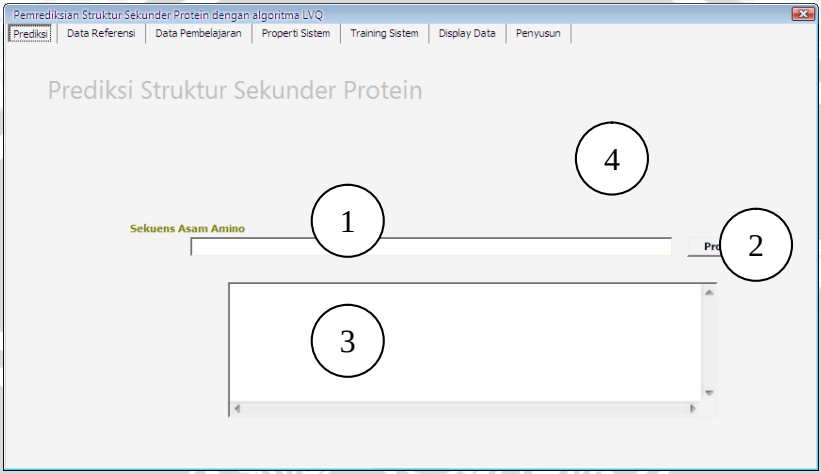
2. Jarak minimum adalah jarak vektor input terhadap prototipe B sehingga output dari proses prediksi struktur sekunder dari vektor masukan adalah bentuk struktur sekunder dari vektor B.

3.2.7 Rancangan Tampilan Antarmuka

Antarmuka sistem terbagi menjadi 6 *form*, yaitu *form* prediksi, *form* input data pembelajaran, *form* input data pembelajaran, *form* properti sistem, *form* data sistem dan *form* pembelajaran sistem.

3.2.7.1 *Form* Prediksi Struktur Sekunder Protein

Untuk *form* prediksi, perancangan antarmukanya terdapat pada gambar 3.20.



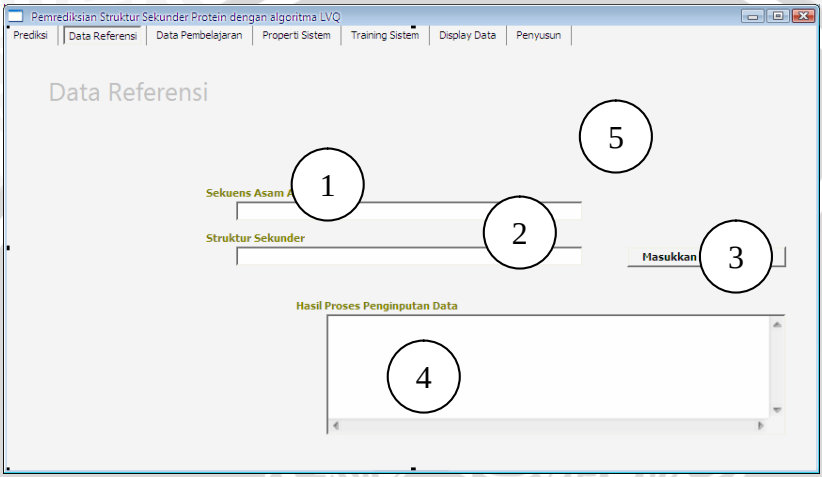
Gambar 3.20 Tampilan Antarmuka *Form* Prediksi Struktur Sekunder Protein

Keterangan gambar 3.20:

1: Edit Box untuk menuliskan rangkaian asam amino yang akan diprediksi struktur sekundernya.
2: Tombol untuk mengeksekusi prosedur prediksi struktur sekunder protein.
3: Memo Box untuk menampilkan hasil prediksi.
4: Gambar *background*

3.2.7.2 *Form* Penginputan Data Referensi

Untuk *form* input data referensi, perancangan antarmukanya terdapat pada gambar 3.21.



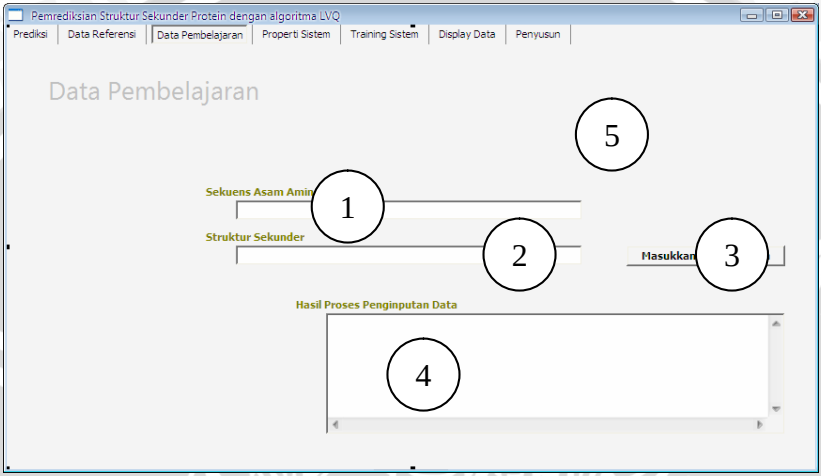
Gambar 3.21 Tampilan Antarmuka *Form* Penginputan Data Referensi

Keterangan gambar 3.21:

1: Edit Box untuk menuliskan rangkaian asam amino sebagai data referensi.
2: Edit Box untuk menuliskan bentuk struktur sekunder dari rangkaian asam amino.
3: Tombol mengeksekusi prosedur memasukkan data pada file.
4: Memo Box untuk menampilkan bahwa telah berhasil disimpan dalam file.
5: Gambar *background*

3.2.7.3 *Form* Penginputan Data Pembelajaran

Untuk *form* penginputan data pembelajaran, perancangan antarmukanya terdapat pada gambar 3.22.



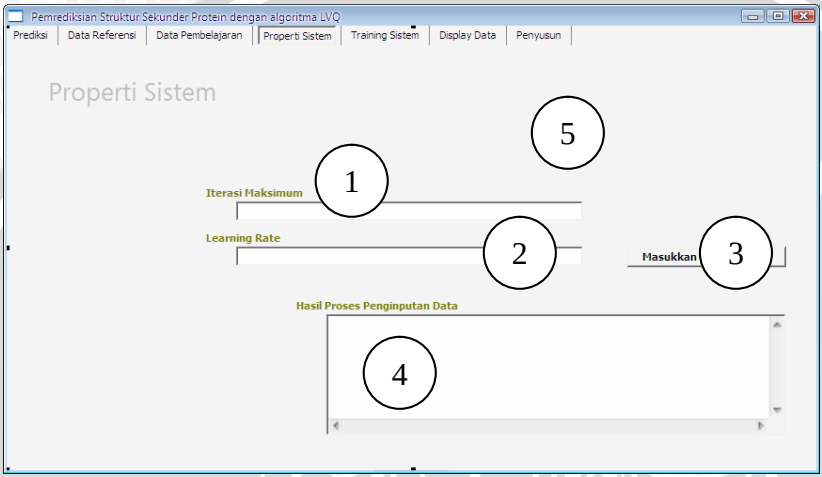
Gambar 3.22 Tampilan Antarmuka *Form* Penginputan Data Pembelajaran

Keterangan gambar 3.22:

1: Edit Box untuk menuliskan rangkaian asam amino sebagai data pembelajaran.
2: Edit Box untuk menuliskan bentuk struktur sekunder dari rangkaian asam amino yang diinputkan
3: Tombol mengeksekusi prosedur memasukkan data pada file.
4: Memo Box untuk menampilkan bahwa telah berhasil disimpan dalam file.
5: Gambar *background*

3.2.7.4 *Form* Penginputan Iterasi Maksimum Dan *Learning Rate*

Form ini bertujuan untuk memasukkan *max epoch* serta besarnya *learning rate*. Perancangan antarmuka ditunjukan pada gambar 3.23



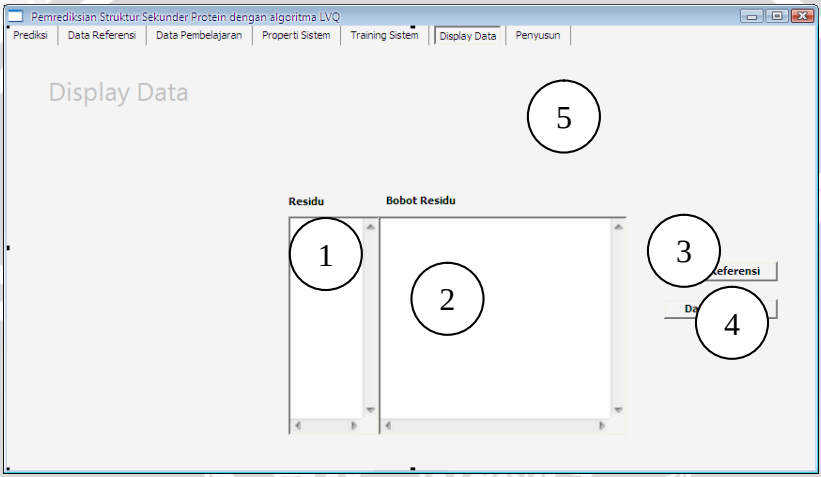
Gambar 3.23 Tampilan Antarmuka *Form* Penginputan *Max epoch* dan *Learning Rate*

Keterangan gambar 3.23:

1: Edit Box untuk menuliskan *max epoch*.
2: Edit Box untuk menuliskan besarnya *learning rate*.
3: Tombol untuk mengeksekusi prosedur memasukkan besarnya iterasi maksimum (*max epoch*) serta besarnya *learning rate*.
4: Memo untuk menampilkan nilai yang telah dimasukkan
5: Gambar *background*

3.2.7.5 *Form* Penampilan Data

Form ini bertujuan untuk menunjukkan pada *user* data-data yang terdapat dalam sistem yaitu data pembelajaran dan data referensi. Untuk perancangan antarmuka *form* data sistem terdapat pada gambar 3.24.



Gambar 3.24 Tampilan Antarmuka *Form* Penampilan Data

Keterangan gambar 3.24:

1: Memo Box untuk menampilkan residu asam amino yang menjadi data pembelajaran dan data referensi.

2: Memo Box untuk menampilkan hasil pengkodean residu asam amino pada data referensi dan data pembelajaran.

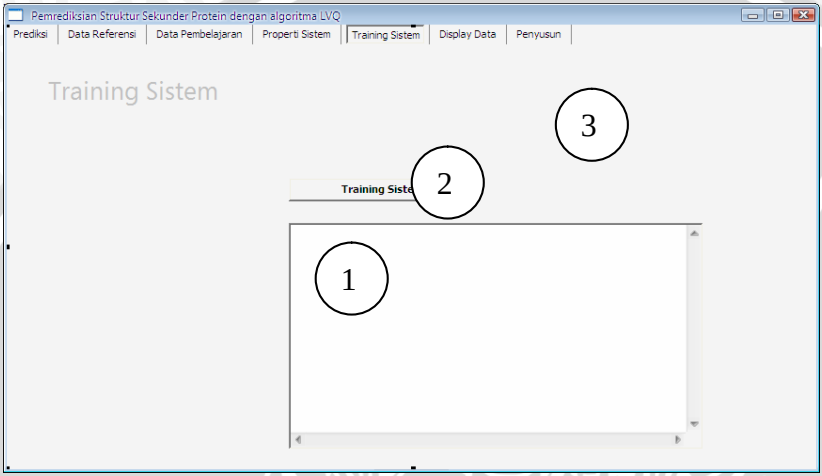
3: Tombol menampilkan data pembelajaran yang berupa residu asam amino beserta hasil pengkodean pada memo box 1 dan memo box 2.

4:Tombol menampilkan data referensi yang berupa residu asam amino beserta hasil pengkodean pada memo box 1 dan memo box 2.

5: Gambar *background*

3.2.7.6 *Form* Pembelajaran Sistem

Form ini digunakan untuk mengeksekusi prosedur pembelajaran sistem. Perancangan antarmukanya terdapat pada gambar 3.25.



Gambar 3.25 Tampilan Antarmuka *Form* Pembelajaran Sistem

Keterangan gambar 3.25 :
1: Memo Box untuk menampilkan pembelajaran pada sistem
2: Tombol untuk mengeksekusi prosedur pembelajaran sistem
3: Gambar *background*

3.3 Rancangan Uji Coba

3.3.1 Tujuan Uji Coba

Tujuan uji coba yang dilakukan pada penelitian ini adalah :

1. Mengetahui iterasi maksimum untuk proses *training* yang menghasilkan tingkat keakurasian paling optimal
2. Mengetahui tingkat keakurasian dari sistem prediksi struktur sekunder protein yang mengimplementasikan algoritma *LVQ* di dalamnya

3.3.2 Proses Uji Coba

Uji coba dilakukan dengan melatih sistem prediksi struktur sekunder dengan batas iterasi maksimum proses *training* berbeda-beda. Iterasi maksimum yang diiterapkan bernilai 50, 100, 150, 200 dan sampai tidak ada pengubahan pada bobot data referensi (konvergen).

Setelah itu, dimasukkan rangkaian asam amino penyusun suatu protein yang terdapat pada data uji coba ke dalam sistem. Kemudian, akan didapat hasil prediksi struktur sekunder protein dari rangkaian asam amino tersebut. Setelah itu, hasil dari proses prediksi tersebut akan dihitung tingkat keakurasiannya dengan menggunakan metode Q3, Qh, Qe, Qc dan Q3*. Dari hasil penghitungan akan dianalisa tingkat keakurasian sistem dalam memprediksikan struktur sekunder protein dengan membandingkan antara hasil training sebanyak 50, 100, 150, 200 dan sampai konvergen.

3.3.3 Penghitungan Keakurasian Hasil

3.3.3.1 Qh

Metode Qh digunakan untuk mengukur tingkat keakurasian prediksi struktur sekunder jenis alfa heliks. Metode Qh dinyatakan pada persamaan 2.9.

3.3.3.2 Qe

Metode Qe digunakan untuk mengukur tingkat keakurasian prediksi struktur sekunder jenis *beta pleated sheet*. Metode Qe dinyatakan pada persamaan 2.10.

3.3.3.3 Qc

Metode Qc digunakan untuk mengukur tingkat keakurasian prediksi struktur sekunder jenis *random coil*. Metode Qc dinyatakan pada persamaan 2.11.

3.3.3.4 Q3

Metode Q3 digunakan untuk mengukur tingkat keakurasian prediksi struktur sekunder pada suatu rangkaian asam amino penyusun satu protein. Metode Q3 dinyatakan pada persamaan 2.12.

3.3.3.5 Q3*

Metode Q3* digunakan untuk mengukur tingkat keakurasian prediksi struktur sekunder pada suatu sistem. Metode Q3* dinyatakan pada persamaan 2.13.

3.3.4 Hasil Penghitungan Prediksi Struktur Sekunder Protein

Hasil prediksi struktur sekunder pada tiap iterasi maksimum, akan dihitung keakuratannya menggunakan metode Q3, Qh, Qe dan Qc. Untuk menghitung rata-rata keakuratan sistem pada tiap iterasi maksimum, akan digunakan metode Q3*. Hasil penghitungannya akan diletakkan pada tabel 3.3. Sedangkan rata-rata Qh,Qe,Qc dan Q3* untuk setiap nilai iterasi maksimum pada proses training, masing-masing akan diletakkan pada tabel 3.4, tabel 3.5, tabel 3.6 dan tabel 3.7.

Tabel 3.3 Tabel Hasil Pengukuran Keakurasian Prediksi Struktur Sekunder

No	Kode PDB	Q3	Qh	Qs	Qc

Tabel 3.4 Tabel Rata-Rata Qh untuk Setiap Iterasi Maksimum

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen

Tabel 3.5 Tabel Rata-Rata Qe untuk Setiap Iterasi Maksimum

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen

Tabel 3.6 Tabel Rata-Rata Qc untuk Setiap Iterasi Maksimum

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen

Tabel 3.7 Tabel Nilai Q3* untuk Setiap Iterasi Maksimum

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi Sistem

Secara umum, sistem prediksi struktur sekunder protein dibuat dengan menggunakan sistem komputer dengan spesifikasi sebagai berikut:

1. Intel Celeron Processor 540 (1.86 GHz, 533 MHz FSB, 1 MB L2 Cache).
2. Memori 512 MB DDR2.
3. Berbagai jenis monitor dan jenis VGA card.
4. Hard disk berukuran 80 GB.

Sedangkan sistem operasi yang digunakan adalah Sistem operasi Microsoft Windows XP Professional Service Pack 2. Untuk pengembangan sistem menggunakan perangkat lunak Borland Delphi 7.0.

4.2 Implementasi Algoritma *LVQ* pada Sistem Prediksi Struktur Sekunder Protein

4.2.1 Struktur Data

Untuk struktur data, dibuat beberapa *record*. Yaitu *record* Bobot, *record* Residu. *Record* Bobot merupakan *record* bobot residu asam amino, *record* yang digunakan untuk meletakkan hasil pengkodean residu asam amino. Sedangkan *record* Residu adalah *record* residu asam amino, yang digunakan untuk meletakkan karakter residu asam amino dari suatu rangkaian asam amino. *Record* Bobot dan Residu digunakan untuk data referensi dan data pembelajaran.

Selain *record* Bobot dan Residu, dibuat beberapa *record* tambahan. Digunakan untuk proses pengurutan data, agar bisa memilih *record* Bobot data referensi mana yang mempunyai jarak *Euclidean* terkecil terhadap *record* Bobot data pembelajaran. *Record-record* tersebut adalah *record* Ranking dan Urutan_Struktur.

Array untuk menyimpan jarak *Euclidean* diwujudkan dalam matrix_*Euclidean*, yang bertipe *array* of real. Beberapa *array* juga dibuat untuk mendukung prosedur dalam sistem. Untuk melihat pengimplementasiannya, dapat dilihat pada *Source Code* 4.1

```
unit StrukturData;

interface

type
  Residux = record
    Indeks : integer;
    Struktur_Sekunder : string;
    Data : array [1..5] of char;
    end;

type
  Bobotx = record
    Indeks : integer;
    Struktur_Sekunder : string;
    Data : array [1..5] of real;
    end;

type
  Ranking = record
    Nomor : integer;
    Indeks : integer;
    Minimum_Euclidean : real;
    Struktur_Sekunder : string;
  end;

type
  Urutan_Struktur = record
    Nomor : integer;
    Indeks : integer;
  end;

type
  matrix_bobot_referensi = array of Bobotx;
  matrix_Euclidean = array of real;
  matrix_residu = array of residux;
  matrix_hasil_encoding = array of real;
  matrix_urutan = array of integer;
  SS = array of string;
  Rank = array of Ranking;
  Rank2nd = array of Urutan_Struktur;

implementation

end.
```

Source Code 4.1 Struktur Data pada Sistem

4.2.2 Penginisialisasian Data

Penginisialisasian data digunakan untuk memasukkan rangkaian asam amino yang diinputkan oleh *user* untuk dijadikan data referensi atau data pembelajaran. Selain itu, prosedur ini juga digunakan untuk memasukkan besarnya *learning rate* dan *max epoch*.

a. Data Referensi

1. Penginputan Rangkaian dan Struktur Sekunder

Penginputan rangkaian dan struktur sekunder, dimasukkan ke dalam edit box. Di akhir rangkaian dan struktur sekunder diberi karakter '.' untuk memberikan batas akhir rangkaian dan struktur sekunder. Pengimplementasiannya ditunjukkan pada *Source Code* 4.2

```
sekuens := Sekuens_Asam_Amino.Text+'.';
struktur := Struktur_Sekunder.Text+'.';
```

Source Code 4.2 Penginputan Rangkaian dan Struktur Sekunder

2. Memeriksa Kondisi Inputan Rangkaian dan Struktur Sekunder

Proses ini digunakan untuk memeriksa apakah *user* telah memasukkan rangkaian dan struktur dengan benar. Untuk pengimplementasiannya ditunjukkan pada *Source Code* 4.3

```
if ((jumlah_karakter_sekuens < 5) or (jumlah_karakter_struktur < 5))
then
  begin
    lanjut := false;
    ShowMessage('Jumlah karakter yang dimasukkan kurang dari 5');
  end;

dilarang_tampil1 := '1234567890BJOUXZbjouxz~!@#$%^&*()_-
+={}|[]\:";<>?,./';
dilarang_tampil2 :=
'1234567890ABDFGIJKLMNOPQRSTUVWXYZabdfgijklmnopqrstuvwxyz~!@#$%^&*()_-
+={}|[]\:";<>?,./';

for a := 1 to jumlah_karakter_sekuens do
  begin
      for b := 1 to 52 do
      begin
        if (sekuens[a] = dilarang_tampil1[b]) then
          begin
            ShowMessage('Terdapat residu asam amino yang tidak
diketahui');
            lanjut := false;
            Break;
          end;
```

```
      end;
      if (lanjut = false) then
        Break;
  end;

for a := 1 to jumlah_karakter_struktur do
  begin
      for b := 1 to 86 do
      begin
        if (struktur[a] = dilarang_tampil2[b]) then
          begin
            ShowMessage('Terdapat struktur sekunder yang tidak
diketahui. '+struktur[a] );
            lanjut := false;
            Break;
          end;
      end;
      if (lanjut = false) then
        Break;
  end;

if ((sekuens = '.') or (struktur = '.')) then
  begin
    ShowMessage(' Inputan belum terisi dengan benar ');
    lanjut := false;;
  end;

if (lanjut = true) then
  begin
    if (jumlah_karakter_sekuens = jumlah_karakter_struktur) then
      begin
         //Penginisialisasian data
      end
    else
       begin
         ShowMessage(' Kesalahan dalam penginputan. Jumlah karakter
sekuens dan struktur tidak sama ');
       end;
   end;
```

Source Code 4.3 Pengecekan Penginputan Rangkaian dan Struktur Sekunder

3. Penginisialisasian Data

Penginsialisasian data dilakukan dengan melakukan beberapa iterasi. Iterasi dilmulai dari 1 sampai jumlah grup rangkaian. Jumlah grup rangkaian didapatkan dari hasil pengurangan jumlah karakter pada rangkaian dikurangi 4. Di setiap iterasi, dilakukan pengisian terhadap *record* yang kemudian disimpan ke dalam file melalui komponen memo. Pengimplementasian ditunjukkan pada *Source Code 4.4*.

```
  begin
    if (jumlah_karakter_sekuens = jumlah_karakter_struktur) then
      begin

        jumlah_grup_sekuens := jumlah_karakter_struktur - 4;
        loadfile_integer(jumlah_CV_referensi,filename3);
```

```
        for a := 1 to jumlah_grup_sekuens do
          begin

            CV_Residu.Indeks := jumlah_CV_referensi + a;
            CV_Bobot.Indeks := jumlah_CV_referensi + a;

            Memo_File_Residu.Lines.Add(IntToStr(CV_Residu.Indeks));
            Memo_File_Bobot.Lines.Add(IntToStr(CV_Bobot.Indeks));

Memo_Hasil_Proses_Penginputan_Data.Lines.Add(IntToStr(CV_Residu.Indeks))
;

            CV_Residu.Struktur_Sekunder :=
struktur[a]+struktur[a+1]+struktur[a+2]+struktur[a+3]+struktur[a+4];
            CV_Bobot.Struktur_Sekunder :=
struktur[a]+struktur[a+1]+struktur[a+2]+struktur[a+3]+struktur[a+4];


            Memo_File_Residu.Lines.Add(CV_Residu.Struktur_Sekunder);
            Memo_File_Bobot.Lines.Add(CV_Bobot.Struktur_Sekunder);

Memo_Hasil_Proses_Penginputan_Data.Lines.Add(CV_Residu.Struktur_Sekunder
);

          for b := 1 to 5 do
            begin

             CV_Residu.Data[b] := sekuens[a+(b-1)];
             encoding_asam_amino(sekuens[a+(b-1)],num);

             CV_Bobot.Data[b] := num;
             Memo_File_Residu.Lines.Add(CV_Residu.Data[b]);
             Memo_File_Bobot.Lines.Add(FloatToStr(CV_Bobot.Data[b]));

Memo_Hasil_Proses_Penginputan_Data.Lines.Add(CV_Residu.Data[b]);

            end;

            Memo_File_Residu.Lines.SaveToFile(filename2);
            Memo_File_Bobot.Lines.SaveToFile(filename1);

          end;

        temp_LD := jumlah_CV_referensi + jumlah_grup_sekuens;
        temp_LD_str := IntToStr(temp_LD);
        buat_file_baru(temp_LD_str,filename3);

      end
```

Source Code 4.4 Penginisialisasian Data Referensi

b. Data Pembelajaran

Isi dari prosedur penginisialisasian data pembelajaran sama seperti penginsialisasian data referensi. Hanya saja, *file* yang disimpan berbeda namanya.

1. Penginputan Rangkaian dan Struktur Sekunder

Proses penginputan rangkaian dan struktur sekunder ditunjukkan pada *Source Code* 4.5

```
sekuens := Sekuens_Asam_Amino_Training.Text+'.';
```

```
struktur := Struktur_Sekunder_Training.Text+'.';
```

Source Code 4.5 Penginputan Rangkaian dan Struktur Sekunder

2. Memeriksa Kondisi Inputan Rangkaian dan Struktur Sekunder

Proses pemeriksaan kondisi inputan rangkaian dan struktur sekunder ditunjukkan pada *Source Code* 4.6

```
if ((jumlah_karakter_sekuens < 5) or (jumlah_karakter_struktur < 5))
then
  begin
    ShowMessage('Jumlah karakter yang dimasukkan kurang dari 5');
    lanjut := false;
  end;

dilarang_tampil1 := '1234567890BJOUXZbjouxz~!@#$%^&*()_-
+={}|[]\:";<>?,./';
dilarang_tampil2 :=
'1234567890ABDFGIJKLMNOPQRSTUVWXYZabdfgijklmnopqrstuvwxyz~!@#$%^&*()_-
+={}|[]\:";<>?,./';

for a := 1 to jumlah_karakter_sekuens do
  begin
      for b := 1 to 52 do
      begin
        if (sekuens[a] = dilarang_tampil1[b]) then
          begin
            ShowMessage('Terdapat residu asam amino yang tidak
diketahui');
            lanjut := false;
            Break;
          end;
      end;
      if (lanjut = false) then
        Break;
  end;

for a := 1 to jumlah_karakter_struktur do
  begin
      for b := 1 to 86 do
      begin
        if (struktur[a] = dilarang_tampil2[b]) then
          begin
            ShowMessage('Terdapat struktur sekunder yang tidak
diketahui');
            lanjut := false;
            Break;
          end;
      end;
      if (lanjut = false) then
        Break;
  end;

if ((sekuens = '.') or (struktur = '.')) then
  begin
    ShowMessage(' Inputan belum terisi dengan benar ');
    lanjut := false;
  end;

if (lanjut = true) then
  begin
    if (jumlah_karakter_sekuens = jumlah_karakter_struktur) then
      begin
```

```
        //Proses penginisialisasian data
      End
    else
      begin
        ShowMessage(' Kesalahan dalam penginputan. Jumlah karakter
sekuens dan struktur tidak sama ');
      end;
  end;
```

Source Code 4.6 Pemeriksaan Penginputan Rangkaian dan Struktur Sekunder

3. Penginisialisasian Data

Proses penginisialisasian data ditunjukkan pada *Source Code* 4.7.

```
      begin
        jumlah_grup_sekuens := jumlah_karakter_struktur - 4;
        loadfile_integer(jumlah_CV_training,filename3);

        for a := 1 to jumlah_grup_sekuens do
          begin

            CV_Residu.Indeks := jumlah_CV_training + a;
            CV_Bobot.Indeks := jumlah_CV_training + a;


Memo_File_Residu_Training.Lines.Add(IntToStr(CV_Residu.Indeks));

Memo_File_Bobot_Training.Lines.Add(IntToStr(CV_Bobot.Indeks));

Memo_Hasil_Proses_Penginputan_Data_Training.Lines.Add(IntToStr(CV_Residu
.Indeks));

            CV_Residu.Struktur_Sekunder :=
struktur[a]+struktur[a+1]+struktur[a+2]+struktur[a+3]+struktur[a+4];
            CV_Bobot.Struktur_Sekunder :=
struktur[a]+struktur[a+1]+struktur[a+2]+struktur[a+3]+struktur[a+4];


Memo_File_Residu_Training.Lines.Add(CV_Residu.Struktur_Sekunder);

Memo_File_Bobot_Training.Lines.Add(CV_Bobot.Struktur_Sekunder);

Memo_Hasil_Proses_Penginputan_Data_Training.Lines.Add(CV_Residu.Struktur
_Sekunder);


          for b := 1 to 5 do
            begin

            CV_Residu.Data[b] := sekuens[a+(b-1)];
            encoding_asam_amino(sekuens[a+(b-1)],num);
            CV_Bobot.Data[b] := num;

            Memo_File_Residu_Training.Lines.Add(CV_Residu.Data[b]);

Memo_File_Bobot_Training.Lines.Add(FloatToStr(CV_Bobot.Data[b]));

Memo_Hasil_Proses_Penginputan_Data_Training.Lines.Add(CV_Residu.Data[b])
;


            end;
```

```
            Memo_File_Residu_Training.Lines.SaveToFile(filename2);
            Memo_File_Bobot_Training.Lines.SaveToFile(filename1);
          end;

        temp_TD := jumlah_CV_training + jumlah_grup_sekuens;
        temp_TD_str := IntToStr(temp_TD);
        buat_file_baru(temp_TD_str,filename3);
      end
```

Source Code 4.7 Penginisialisasian Data Training

c. *Learning Rate* dan *Max epoch*

Proses penginputan *learning rate* dan *max epoch*, menggunakan beberapa prosedur tambahan. Yaitu prosedur buat_file_baru. *Source Code* 4.8 menunjukkan bagaimana pengimplementasiannya.

```
filename1 := 'F:\Skripsi\Data\Program\Skripsi3\ME.txt';
filename2 := 'F:\Skripsi\Data\Program\Skripsi3\LR.txt';


  begin
    buat_file_baru(IM,filename1);
    buat_file_baru(LR,filename2);
  end;
```

Source Code 4.8 Penginputan *Learning Rate* dan *Max epoch*

4.2.3 Pembelajaran Sistem

Proses pembelajaran dilakukan dalam suatu proses iterasi. Proses pembelajaran dihentikan apabila sudah mencapai *max epoch*.

a. Kondisi yang Menghentikan Iterasi

Kondisi yang menghentikan iterasi adalah jika *Max epoch* sudah terlampaui atau learning rate sudah tidak berpengaruh lagi. Iterasi diproses dengan menggunakan *while – do*. *Source Code* 4.9 menunjukkan proses w*hile – do* untuk batas iterasi.

```
while (stop_iterasi = false) do //untuk sampai konvergen

begin
      //Proses pencarian nilai Euclidean terkecil
      //Proses pengubahan bobot
        if (count = 5) then
            begin
             stop_iterasi = true;
            end;
       b := b + 1;
      end


while (b < ME) do//untuk ME ditetapkan

      begin
      //Proses pencarian nilai Euclidean terkecil
```

```
//Proses pengubahan bobot
   if (count = 5) then
       begin
          b := ME;
       end;
 b := b + 1;
end
```

Source Code 4.9 Iterasi dengan *While – Do*

b. Peng-*load*-an Data dari File ke *Record*

Pada *Source Code* ke 4.10, ditunjukkan bagaimana *file* diload ke *record*. Tahapannya adalah, *file* diload ke komponen memo, kemudian di letakkan ke dalam *record* berdasarkan baris pada memo.

```
Memo_File_Bobot_Data.Lines.Clear;
Memo_File_Bobot_Data.Lines.LoadFromFile(filename3);

for a := 1 to jumlah_data_referensi do
  begin
    bobot_referensi[a].Indeks :=
StrToInt(Memo_File_Bobot_Data.Lines[(a*7)-7]);
    bobot_referensi[a].Struktur_Sekunder :=
Memo_File_Bobot_Data.Lines[(a*7)-6];
    for b := 1 to 5 do
      begin
        bobot_referensi[a].Data[b] :=
StrToFloat(Memo_File_Bobot_Data.Lines[(a*7)-(6-b)]);
      end;
  end;

Memo_File_Bobot_Data.Lines.Clear;
Memo_File_Bobot_Data.Lines.LoadFromFile(filename4);

for a := 1 to jumlah_data_training do
  begin
    temp_record.Struktur_Sekunder := Memo_File_Bobot_Data.Lines[(a*7)-
6];

    for b := 1 to 5 do
      begin
        temp_record.Data[b] :=
StrToFloat(Memo_File_Bobot_Data.Lines[(a*7)-(6-b)]);
      end;
```

Source Code 4.10 *Load* Data ke Dalam *Record*

c. Penentuan Jarak *Euclidean*

Dalam menentukan jarak *Euclidean*, dilakukan dengan membandingkan antara *record* bobot residu asam amino data referensi dengan *record* bobot residu asam amino data pembelajaran. Pengimplementasiannya ditunjukkan pada *Source Code* 4. 11. Nilai min *Euclidean* diisikan dengan nilai yang besar, seperti 1000. Kemudian, untuk setiap data referensi di hitung jarak *Euclidean*nya terhadap data pembelajaran. *Array* jarak *Euclidean* digunakan untuk

meletakkan hasil penghitungan. Kemudian jarak *Euclidean* terkecil, diletakkan pada variabel min_euc dan indeksnya diletakkan pada min_Id.

```
for c := 1 to jumlah_data_referensi do
        begin
          Euclidean[c] := 0;


          for d := 1 to 5 do
            begin
              temp := temp_record.Data[d]-bobot_referensi[c].Data[d];

              if (temp < 0) then
                begin
                  temp := temp * (-1);
                end;

              Euclidean[c] := Euclidean[c] + temp;

            end;

          if(min_euc > Euclidean[c]) then
            begin
              min_euc := Euclidean[c];
              min_Id := c;
            end;
        end;
```

Source Code 4.11 Penentuan Jarak *Euclidean*

d. Perbaikan Nilai Pada Bobot

Setelah ditemukan mana *record* yang memiliki jarak *Euclidean* terkecil, dilakukan perbaikan bobot.. Jika bentuk struktur sekundernya sama, maka pengimplementasiannya ditunjukkan pada *Source Code* 4.12. Jika tidak sama ditunjukkan pada *Source Code* 4.13

```
        if (temp_record.Struktur_Sekunder =
bobot_referensi[min_Id].Struktur_Sekunder) then
          begin
            count := 0;
            for d := 1 to 5 do
              begin
                temp := bobot_referensi[min_Id].Data[d];
                bobot_referensi[min_Id].Data[d] :=
bobot_referensi[min_Id].Data[d] + (LR*(temp_record.Data[d]-
bobot_referensi[min_Id].Data[d]));
                if (temp = bobot_referensi[min_Id].Data[d]) then
                  begin
                    count := count + 1;
                  end;
            end;
               //Cek kondisi berhenti
          end
```

Source Code 4.12 Pengubahan Bobot Jika Struktur Sekundernya Sama

```
else if (temp_record.Struktur_Sekunder <>
bobot_referensi[min_Id].Struktur_Sekunder) then
          begin

            count := 0;
            for d := 1 to 5 do
              begin

              temp := bobot_referensi[min_Id].Data[d];

              bobot_referensi[min_Id].Data[d] :=
bobot_referensi[min_Id].Data[d] - (LR*(temp_record.Data[d] -
bobot_referensi[min_Id].Data[d]));

              if (temp = bobot_referensi[min_Id].Data[d]) then
                begin
                  count := count + 1;
                end;

            end;
              // Cek kondisi berhenti

          end;
```

Source Code 4.13 Pengubahan Bobot Jika Struktur Sekundernya Tidak Sama

e. Pengurangan *Learning Rate*

Pengurangan *Learning Rate* dilakukan pada setiap satu kali iterasi. *Source Code* 4.14 menunjukkan proses pengurangan *Learning Rate*

```
LR := LR - (0.1*LR);
b:= b + 1;
```

Source Code 4.14 Pengurangan *Learning Rate*

4.2.4 Prediksi Struktur Sekunder Protein

Proses prediksi hampir sama dengan proses pembelajaran. Pada proses prediksi ditetapkan prosedur pengurutan untuk menentukan string hasil. Proses pengurutan dilakukan dengan cara mengurutkan *record* dengan jarak *Euclidean* terkecil dan memutuskan bentuk struktur sekunder mana dulu yang harus diletakkan pada *array* bentuk struktur sekunder.

a. Memeriksa Kondisi Inputan Rangkaian Asam Amino

Pemeriksaan kondisi inputan rangkaian dan struktur sekunder ditunjukkan pada *Source Code* 4.15

```
if (jumlah_karakter < 5)then
  begin
    lanjut := false;
```

```
    ShowMessage('Jumlah karakter yang dimasukkan kurang dari 5');
  end;

if (sekuens = '.') then
  begin
    ShowMessage (' Sekuens asam amino belum dimasukkan ');
    lanjut := false;
  end;

dilarang_tampil := '1234567890BJOUXZbjouxz~!@#$%^&*()_-
+={}|[]\:";<>?,./';

for a := 1 to jumlah_karakter do
  begin
      for b := 1 to 52 do
      begin
        if (sekuens[a] = dilarang_tampil[b]) then
          begin
         ShowMessage('Terdapat residu asam amino yang tidak diketahui');
            lanjut := false;
            Break;
          end;
      end;
      if (lanjut = false) then
        Break;
  end;
```

Source Code 4.15 Pemeriksaan Penginputan Rangkaian Asam Amino

b. Peng-*load*-an Data dari *File* ke *Record*

Peng-*load*-an data dari *file* ke *record* ditunjukkan pada *Source Code* 4.16.

```
    for a := 1 to jumlah_CV_referensi do
      begin

        bobot_referensi[a].Indeks :=
StrToInt(Memo_Temp_Pemrediksian.Lines[0+(7*(a-1))]);
        bobot_referensi[a].Struktur_Sekunder :=
Memo_Temp_Pemrediksian.Lines[1+(7*(a-1))];

       for b := 1 to 5 do
          begin
            bobot_referensi[a].Data[b] :=
StrToFloat(Memo_Temp_Pemrediksian.Lines[(b+1)+(7*(a-1))]);
          end;
      end;

    SetLength(bobot_asam_amino_baru,jumlah_grup_sekuens+1);
    SetLength(num_encoding,jumlah_karakter+1);

    for a := 1 to jumlah_karakter do
      begin

        encoding_asam_amino(sekuens[a],num);
        num_encoding[a] := num;
      end;
```

```
for a := 1 to jumlah_grup_sekuens do
  begin
    bobot_asam_amino_baru[a].Indeks := a;
    for b := 1 to 5 do
      begin
        bobot_asam_amino_baru[a].Data[b] := num_encoding[a+(b-1)];
      end;
  end;
```

Source Code 4.16 Peng-load-an file dan rangkaian ke *record*

c. Penentuan Jarak *Euclidean*

Penentuan jarak *Euclidean* ditunjukkan pada *Source Code* 4.17

```
for a := 1 to jumlah_grup_sekuens do
    begin
      for b := 1 to jumlah_CV_referensi do
       begin
          Euclidean[b] := 0;
       end;
      for b := 1 to jumlah_CV_referensi do
       begin
           begin
            temp := bobot_asam_amino_baru[a].Data[c] -
bobot_referensi[b].Data[c];

            if (temp < 0) then
              begin
                temp := temp * (-1);
              end;

            Euclidean[b] := Euclidean[b] + temp;

           end;

         if (Euclidean[b] < minimum) then
           begin
             minimum := Euclidean[b];
             indeks := b;
           end;
       end;
```

Source Code 4.17 Penentuan Jarak *Euclidean*

d. Penempatan Bentuk Struktur Sekunder

Struktur sekunder diletakkan pada suatu *array*, tetapi, penempatannya didasarkan pada jarak *Euclidean*. Jarak *Euclidean* yang terbesar, diletakkan pada awal, sedangkan jarak *Euclidean* terkecil diletakkan belakangan. *Source Code* 4.18 menunjukkan penempatan bentuk struktur sekunder.

```
SetLength(Struktur_Sekunder,jumlah_grup_sekuens+1);
maksimum := -1;


for a := 1 to jumlah_grup_sekuens do
  begin

    for b := 1 to jumlah_grup_sekuens do
      begin
```

```
            if (Cool[b].Minimum_Euclidean >= maksimum) then
              begin

                maksimum := Cool[b].Minimum_Euclidean;
                indeks := b;
                Struktur_Sekunder[a].Indeks := Cool[b].Indeks;
                Struktur_Sekunder[a].Nomor := Cool[b].Nomor;
              end;
          end;


        Cool[indeks].Minimum_Euclidean := -1;
        maksimum := -1;
      end;

    SetLength(SSB,((jumlah_grup_sekuens*5)+1));
     for a := 1 to jumlah_grup_sekuens do
      begin
        temp1 := Struktur_Sekunder[a].Indeks;
        temp_string := bobot_referensi[temp1].Struktur_Sekunder;
        temp1 := Struktur_Sekunder[a].Nomor;
        SSB[temp1] := temp_string[1];
        SSB[temp1+1] := temp_string[2];
        SSB[temp1+2] := temp_string[3];
        SSB[temp1+3] := temp_string[4];
        SSB[temp1+4] := temp_string[5];
      end;
```

Source Code 4.18 Penempatan Bentuk Struktur Sekunder

e. Mengeluarkan Output berupa Hasil Prediksi

Setelah diletakkan pada *array*, bentuk struktur sekunder ditunjukkan kepada *user* dengan menggunakan komponen memo. *Source Code* 4.19 menunjukkan proses ini.

```
    temp_string := '';
    for a := 1 to jumlah_grup_sekuens*5 do
      begin
        temp_string := temp_string + SSB[a];
      end
```

Source Code 4.19 Pengeluaran Output

4.3 Prosedur Lain

4.3.1 Prosedur buat_file_baru

Prosedur buat_file_baru ditunjukkan oleh *Source Code* 4.20

```
procedure buat_file_baru(var data,namafile : string);
var
MyFile : text;
begin
  AssignFile(MyFile,namafile);
  rewrite(MyFile);
  write(MyFile, data);
  CloseFile(MyFile);
end;
```

Source Code 4.20 Prosedur Buat_File_Baru

4.3.2 Prosedur loadfile_integer

Prosedur loadfile_integer ditunjukkan oleh *Source Code* 4.21

```
procedure loadfile_integer(var A : integer; namafile : string);
var
MyFile : Text;
S : String;
begin
if FileExists(namafile)then
  begin
    AssignFile(MyFile,namafile);
    Reset(MyFile);
    Readln(MyFile,S);
    A := StrToInt(S);
    CloseFile(MyFile);
  end
else
  ShowMessage('File '+namafile+' tidak ada');
end;
```

Source Code 4.21 Prosedur Loadfile_Integer

4.3.3 Prosedur loadfile_real

Prosedur loadfile_real ditunjukkan oleh *Source Code* 4.22

```
procedure loadfile_real(var A : real; namafile : string);
var
MyFile : Text;
S : String;
begin
if FileExists(namafile)then
  begin
    AssignFile(MyFile,namafile);
    Reset(MyFile);
    Readln(MyFile,S);
    A := StrToFloat(S);
    CloseFile(MyFile);
  end
else
  ShowMessage('File '+namafile+' tidak ada');
end;
```

Source Code 4.22 Prosedur Loadfile_real

4.3.4 Prosedur Encoding_asam_amino

Prosedur Encoding_asam_amino ditunjukkan oleh *Source Code* 4.23

```
procedure encoding_asam_amino(var S : char; var num : real);
var
a : char;
B,C : integer;
begin
B := 0;
```

```
for a := 'A' to 'Z' do
  begin
  B := B + 1;
  if (S = a) then
    begin
      C := B + 64;
      num := ((0.95 * (C - 65))/(90 - 65))+0.05;
    end
  else if ((S = '-') or (S = '')) then
    begin
      num := 0;
    end;
  end;
end;
```

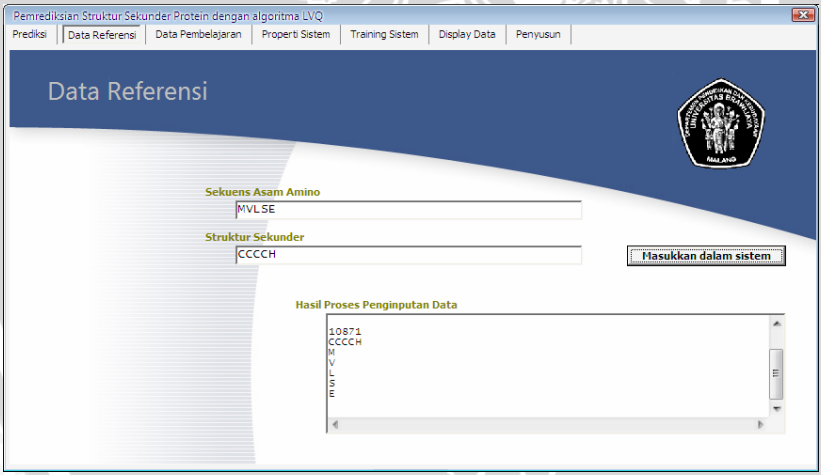
Source Code 4.23 Prosedur Encoding_Asam_Amino

4.4 Output Sistem

4.4.1 Proses Inisialisasi Data

a. Data Referensi

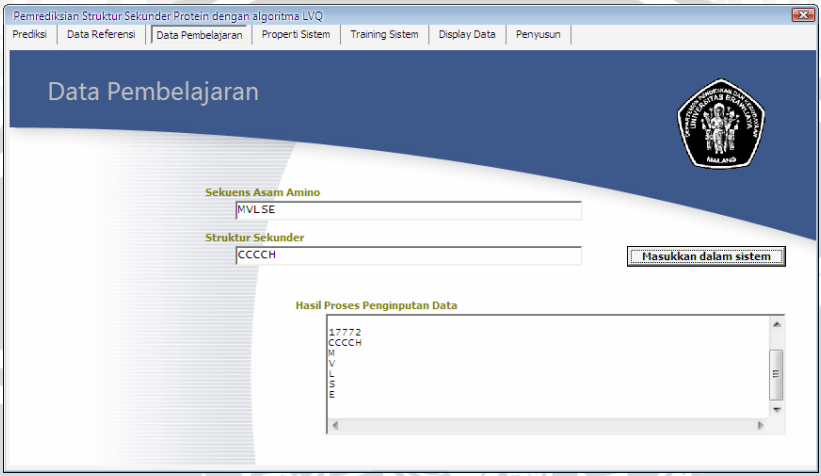
Output sistem untuk proses inisialisasi data referensi, ditunjukkan pada gambar 4.1



Gambar 4.1 Output Sistem Penginisialisasian Data Referensi

b. Data Pembelajaran

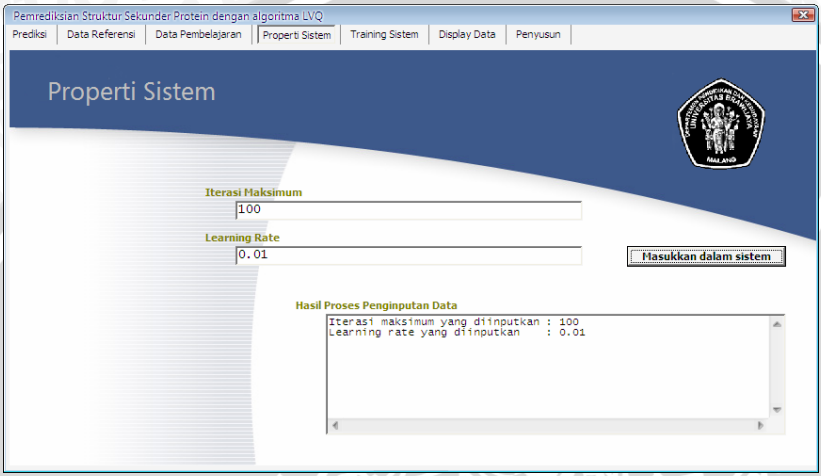
Output sistem untuk proses inisialisasi data pembelajaran, ditunjukkan pada gambar 4.2



Gambar 4.2 Output Sistem Penginisialisasian Data Pembelajaran

c. *Learning Rate* dan *Max epoch*

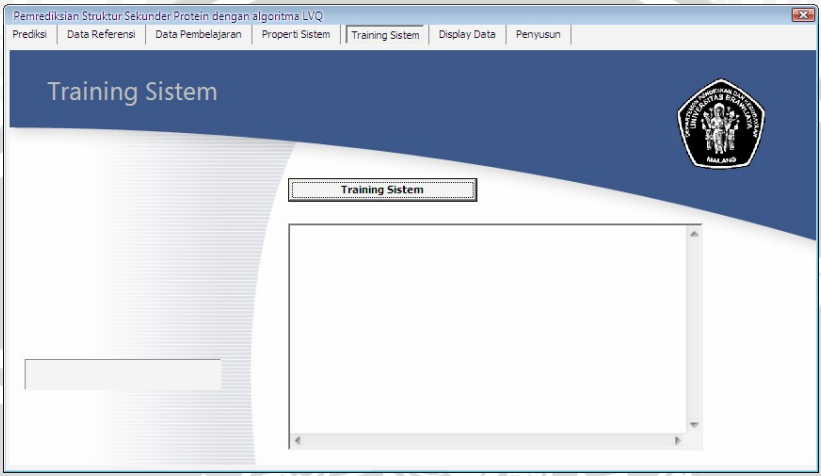
Output sistem untuk proses inisialisasi *learning rate* dan *max epoch* ditunjukkan pada gambar 4.3



Gambar 4.3 Output Sistem Penginisialisasian Data *Learning Rate* dan *Max epoch*

4.4.2 Pembelajaran Sistem

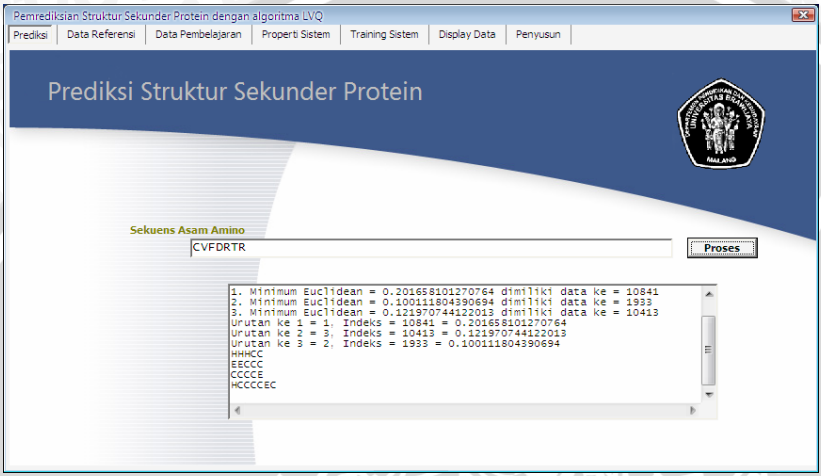
Output sistem untuk pembelajaran sistem ditunjukkan pada gambar 4.4



Gambar 4.4 Output Sistem Pembelajaran Sistem

4.4.3 Prediksi Struktur Sekunder Protein

Output sistem untuk proses prediksi struktur sekunder, ditunjukkan pada gambar 4.5



Gambar 4.5 Output Sistem Prediksi Struktur Sekunder Protein

4.5 Hasil Uji Coba

4.5.1 Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein

Tabel hasil penghitungan keakurasian prediksi struktur sekunder protein pada tiap iterasi maksimum (50, 100, 150, 200 dan sampai konvergen) dapat dilihat pada lampiran. Untuk iterasi maksimum 50 pada tabel 7.4, untuk iterasi maksimum 100 pada tabel 7.5, untuk iterasi maksimum 150 pada tabel 7.6, untuk iterasi maksimum 200 pada tabel 7.7, untuk iterasi maksimum sampai konvergen pada tabel 7.8.

4.5.2 Rata-rata Qh pada Tiap Iterasi Maksimum

Hasil rata-rata Qh pada tiap iterasi maksimum untuk 10 data uji coba, diletakkan pada tabel 4.1.

Tabel 4.1 Tabel Rata-Rata Qh untuk Setiap Iterasi Maksimum Dalam Persen (%)

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen
48.71401073	48.71401073	48.71401073	48.71401073	48.71401073

4.5.3 Rata-rata Qe pada Tiap Iterasi Maksimum

Hasil rata-rata Qe pada tiap iterasi maksimum untuk 10 data uji coba, diletakkan pada tabel 4.2.

Tabel 4.2 Tabel Rata-Rata Qe untuk Setiap Iterasi Maksimum Dalam Persen (%)

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen
40.86259869	40.86259869	40.86259869	40.86259869	40.86259869

4.5.4 Rata-rata Qc pada Tiap Iterasi Maksimum

Hasil rata-rata Qc pada tiap iterasi maksimum untuk 10 data uji coba, diletakkan pada tabel 4.3.

Tabel 4.3 Tabel Rata-Rata Qc untuk Setiap Iterasi Maksimum Dalam Persen (%)

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen
51.38657942	50.94119335	51.11263306	51.33391366	51.21897113

4.5.5 Q3* untuk Tiap Iterasi Maksimum

Hasil Q3* pada tiap iterasi maksimum untuk 10 data uji coba, diletakkan pada tabel 4.4.

Tabel 4.4 Tabel Nilai Q3* untuk Setiap Iterasi Maksimum Dalam Persen (%)

ME = 50	ME = 100	ME = 150	ME = 200	ME sampai Konvergen
46.92724668	46.66970117	46.76799058	46.89925087	46.83099149

4.6 Analisa Hasil

Dari tabel hasil uii coba, didapatkan tingkat keakurasian sistem dalam memprediksikan struktur sekunder jenis alfa heliks, *beta pleated sheet* dan *random coil*. Selain itu, juga didapat tingkat keakurasian sistem prediksi struktur sekunder dengan menggunakan metode Q3*.

Prediksi struktur sekunder mengacu pada data yang dikeluarkan oleh RSCB Protein Data Bank Sehingga diasumsikan bentuk struktur sekunder yang dikeluarkan oleh RSCB Protein Data Bank merupakan data yang akurat 100 %.

Nilai 0 % pada tabel 7.4, 7.5, 7.6, 7.7, 7.8 pada lampiran, merepresentasikan bentuk struktur sekunder tersebut tidak terdapat pada data struktur sekunder RSCB Protein Data Bank.

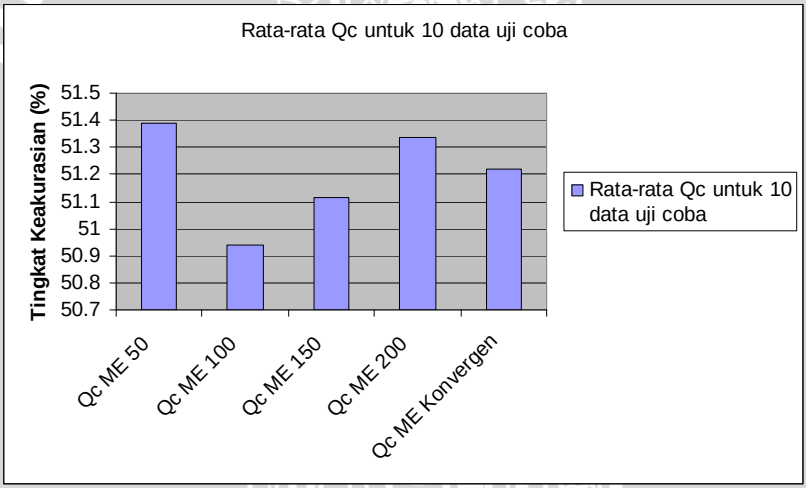
Tabel 4.4 menunjukkan bahwa pada iterasi maksimum 50, nilai Q3* memperoleh hasil tertinggi yaitu 46,92 %. Disusul nilai Q3* pada iterasi maksimum 200 yaitu 46,90 %. Kemudian nilai Q3* pada iterasi maksimum sampai konvergen yaitu 46,83 %. Selanjutnya nilai Q3* pada iterasi maksimum 150, yaitu 46,77 %. Terakhir pada iterasi maksimum 100 sebesar 46,67 %.

Tabel 4.1 dan tabel 4.2 menunjukkan bahwa meskipun iterasi maksimum berubah-ubah, hasil rata-rata Qh dan Qe tetap. Untuk Qh adalah 48,71 % sedangkan Qe yaitu 40,86 %. Sedangkan Tabel 4.3 menunjukkan bahwa, iterasi maksimum yang berubah-ubah, juga mengubah rata-rata Qc.

Bedasarkan nilai Q3* yang didapat, maka iterasi maksimum yang mempunyai hasil yang optimal berada pada iterasi maksimum 50.

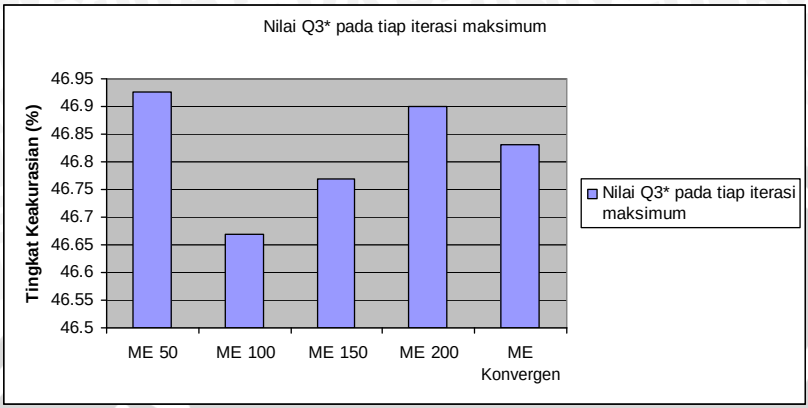
4.7 Analisa Sistem

Dari tabel 4.1, 4.2, 4.3 dan 4.4, pengubahan iterasi maksimum mengubah rata-rata keakurasian prediksi struktur sekunder *random coil*. Gambar 4.6 menunjukkan grafik pengubahan rata-rata Qc pada tiap iterasi maksimum.



Gambar 4.6 Grafik rata-rata Qc untuk 10 data uji coba

Hasil pengubahan iterasi maksimum juga berpengaruh pada nilai Q3*. Dari gambar 4.7, menggambarkan grafik nilai Q3* untuk tiap iterasi maksimum.



Gambar 4.7 Grafik Nilai Q3* untuk 10 data uji coba

Pengubahan nilai iterasi maksimum memberikan pengubahan nilai pada bobot data referensi. Pengubahan nilai pada bobot data referensi berbeda antar tiap iterasi maksimum yang diberikan.

Perbedaan pengubahan nilai pada bobot data referensi untuk tiap iterasi maksimum, menyebabkan berubahnya rata-rata keakurasian dari metode Qc. Sedangkan rata-rata keakurasian dari metode Qh dan Qe tetap. Pengubahan nilai rata-rata keakurasian pada tiap iterasi maksimum dari metode Qc berbanding lurus dengan pengubahan nilai Q3* untuk tiap iterasi maksimum.

Penambahan nilai iterasi maksimum tidak selalu berbanding lurus dengan penambahan nilai keakurasian Q3*. Hal ini ditunjukkan pada gambar 4.7 bahwa *training* dengan iterasi maksimum 50 mempunyai nilai Q3* paling tinggi dibandingkan dengan yang lain. Dari gambar 4.7 menunjukkan bahwa iterasi maksimum 50 sudah mendapatkan hasil yang paling optimal.

Pengubahan iterasi maksimum memberikan perubahan pada nilai bobot data referensi. Berubahnya nilai Q3* untuk tiap iterasi maksimum, disebabkan karena nilai bobot data referensi berubah. Dengan jumlah iterasi maksimum yang berbeda-beda, maka nilai bobot data referensi yang berubah berbeda-beda untuk tiap iterasi maksimum. Hal ini mendorong terjadinya beberapa pergeseran indeks data yang memiliki jarak *euclidean* terkecil terhadap data uji coba. Misalnya pada iterasi maksimum 150, data uji coba 1NXB mempunyai jarak *euclidean* terkecil dengan bobot data referensi

indeks 6784 dengan bentuk CCEEE. Tetapi, setelah diiterasi dengan iterasi maksimum 200, bergeser ke bobot data referensi indeks 2178 dengan bentuk struktur sekunder CEEEC. Setelah digabungkan dengan prediksi struktur sekunder dari karakter asam amino yang lain, terjadi peningkatan keakurasian. Dari 61,111 % menjadi 63,8889 %. Hal sama terjadi pada data uji coba lain yang mengalami kenaikan atau penurunan tingkat keakurasian.

4.7.1 Analisa Algoritma *LVQ*

Dalam penelitian ini, algoritma *LVQ* telah berhasil menyimpan seluruh bentuk struktur sekunder rangkaian asam amino dari data referensi yang kemudian digunakan untuk dasar prediksi struktur sekunder rangkaian asam amino.

Pada proses *training* algoritma *LVQ*, nilai dari bobot data referensi selalu berubah-ubah. Hal ini dikarenakan proses *training LVQ* mengubah nilai dari bobot data referensi menjadi mendekati nilai pada bobot data *training*. Tetapi karena batas iterasi maksimum yang belum tercapai, kondisi yang telah optimal memungkinkan menjadi tidak optimal karena *over learning*. Hal ini ditunjukkan pada ujicoba dengan data 2SOD. Pada data uji coba 2SOD, keakurasian dalam memprediksi struktur sekunder *random coil* menurun. Setelah dilakukan proses *training* sebanyak 50 kali, terjadi penurunan tingkat keakurasian pada iterasi maksimum 100 kali. Karena ketidakstabilan tersebut, algoritma *LVQ* tidak mempunyai patokan yang jelas untuk penentuan kondisi berhenti pada tiap kasus. Oleh karena itu, perlu dilakukan percobaan untuk menentukan kondisi berhenti dengan hasil yang optimal. Setelah dilakukan percobaan pada kasus prediksi struktur sekunder protein ini, dicapai titik optimal (nilai Q3* terbesar) dengan iterasi maksimum proses *training* sebanyak 50 kali. Selain itu, proses training dari algoritma *LVQ* yang bersifat *Brute Force* menyebabkan perubahan bobot data referensi mengacu pada bobot data training yang terakhir. Pemilihan bentuk struktur sekunder, didasarkan dari jarak *Euclidean* yang paling kecil.

Dari sekuens asam amino yang diinputkan sebagai data referensi, terdapat beberapa *codebook vector* yang memiliki 5 residu asam amino yang sama. Seperti, *codebook vector* ke 1 dengan *codebook vector* ke 5032. Semuanya memiliki residu M, V, L, S, E. Tetapi, mempunyai bentuk struktur sekunder yang berbeda. Pada *codebook vector* dengan indeks ke 1, memiliki bentuk struktur sekunder

CCCCH. Sedangkan *codebook vector* dengan indeks ke 5032, memiliki bentuk struktur sekunder CCCCC.

Pada proses training dilakukan pembandingan untuk mencari jarak *euclidean* terkecil antara *codebook vector* dari data pembelajaran dan data referensi. Proses yang terjadi pada proses *training* berlangsung secara *Brute Force* atau satu *codebook vector* dibandingkan dengan semua *codebook vector* yang ada pada data referensi. Hal ini diterapkan pada semua *codebook vector* data pembelajaran. Proses pembandingan ini tidak memungkinkan memilih 2 *codebook vector* yang mempunyai jarak *euclidean* terkecil yang sama. Sistem secara otomatis, memilih *codebook vector* yang berindeks terkecil. Hal ini membuat tidak terjadi pengubahan bobot pada *codebook vector* pada indeks ke 5032. Karena pengubahan bobot hanya terjadi pada *codebook vector* indeks ke 1. Jika sistem dicoba untuk memprediksikan struktur sekunder dari sekuens M, V, L, S, E maka yang hasil yang dikeluarkan adalah struktur sekunder dari *codebook vector* indeks ke 5032 yaitu CCCCC. Karena nilai jarak *euclidean* bernilai 0. Jika diinputkan M, V, L, S, E, G maka diprediksikan oleh sistem berstruktur sekunder CCCCCH. Sedangkan hasil yang seharusnya adalah CCCCHH.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penelitian yang telah dilaksanakan, terdapat beberapa kesimpulan yang dapat diambil, yaitu :

1. Tingkat keakurasian metode *LVQ* jika dihitung dengan metode Q3* pada tiap iterasi maksimum mempunyai nilai yang tidak sama. Pada iterasi maksimum 50, nilai Q3* memperoleh hasil tertinggi yaitu 46,92 %. Disusul nilai Q3* pada iterasi maksimum 200 yaitu 46,90 %. Kemudian nilai Q3* pada iterasi maksimum sampai konvergen yaitu 46,83 %. Selanjutnya nilai Q3* pada iterasi maksimum 150, yaitu 46,77 %. Terakhir pada iterasi maksimum 100 sebesar 46,67 %.
2. Kondisi berhenti optimal untuk prediksi struktur sekunder protein berada pada iterasi maksimum 50.
3. Penambahan iterasi maksimum tidak selalu berbanding lurus dengan penambahan nilai keakurasian Q3*.

5.2 Saran

Codebook vector yang memiliki residu asam amino yang sama pada data referensi akan mempengaruhi tingkat keakurasian sistem. Maka perlu dilakukan penambahan data pada *codebook vector*, sehingga membuat *codebook vector* bersifat unik antara 1 dengan yang lain.

DAFTAR PUSTAKA

Abdalla, S.O., Deris, S. 2005. ***Predicting Protein Secondary Structure Using Artificial Neural Network: Current Status and Future Directions.*** http://eprints.utm.my/4309/1/SaadOsmanAbdallaPCD2005TTT.pdf. Tanggal akses : 9 Januari 2009

Arjunan, S. 2007. ***Literature Survey of Protein Secondary Prediction 1.0.*** http://eprints.utm.my/929/1/JT34C8.pdf. Tanggal Akses : 30 Juni 2008

Arnshea, C. 2006. ***The Relative Importance Of Input Encoding And Learning Methodology On Protein Secondary Structure Prediction.*** http://etd.gsu.edu/theses/available/etd-02072006-103620/unrestricted/clayton_arnshea_200605_ms.pdf. Tanggal Akses : 23 Mei 2009

Campbell, N. 1955. ***A Text-Book of Organic Chemistry***. Oliver and Boyd. London

Fessenden, R.J. dan J.S. Fessenden. 1997. ***Dasar-Dasar Kimia Organik***. Binarupa Aksara. Jakarta.

Harper, H. A. 1963. ***Review of Physiological Chemistry***. Lange Medical. Los Altos.

Hartono, J. 1999. ***Pengenalan Komputer***. ANDI. Yogyakarta.

Kimball, J. W. 1983. ***Biologi***. Erlangga. Jakarta

Kohonen, T., J. Hynninen, J.Kangas, J. Laaksonen dan Torkkola. 1996. ***LVQ PAK : The Learning Vector Quantization Program Enckage.*** http://reference.kfupm.edu.sa/content/l/v/lvq_pak__the_learning_vector_quantizatio_442191.pdf. Tanggal akses : 19 Agustus 2008

Kristanto, Andri. 2004. ***Jaringan Syaraf Tiruan***. Gaya Media. Yogyakarta.

Kusumadewi, Sri. 2003. ***Artificial Intelligence (Teknik dan Aplikasinya)***. Graha Ilmu. Yogyakarta.

Lauritzen,G.C. 1992. ***What is Protein.*** http://extension.usu.edu/files/publications/publication/FN_191.pdf. Tanggal Akses : 20 Agustus 2008

Mazur, A. dan B. Harrow. 1971. ***Textbook of Biochemistry***. W. B. Saunders. Philadelphia.

RSCB Protein Data Bank. 2009. ***SS.txt.*** http://www.pdb.org/pdb/files/ss.txt. Tanggal akses : 8 September 2008

Sawhney, S.K. dan R. Singh. 2005. ***Introductory practical biochemistry***. Alpha Science International. Harrow

Sheperd, A. 1998. ***Protein Secondary Structure Prediction with Neural Networks.*** http://www.biochem.ucl.ac.uk/~shepherd/sspred_tutorial/ss-index.html. Tanggal akses : 18 Januari 2009

Singh, M., P.S. Shandu, dan R.K. Kaur. 2008. ***Protein Secondary Structure Prediction.*** PWASET VOLUME 32 AUGUST 2008 ISSN 2070-3740. http://www.waset.org. Tanggal Akses : 19 Januari 2009

Lampiran 1. Data Referensi Sistem

Protein-protein yang dijadikan sebagai data referensi, ditunjukkan pada tabel 7.1.

Tabel 7.1 Data Referensi Sistem

No	Nama	Kode PDB
1	MYOGLOBIN	101M:A
2	GLUTATHIONE S-TRANSFERASE P1-1	10GS:B
3	PROTEIN (CYTOSINE-SPECIFIC METHYLTRANSFERASE HHAI)	10MH:A
4	PROTEIN (INORGANIC PYROPHOSPHATASE)	117E:A
5	ASPARAGINE SYNTHETASE	11AS:B
6	PROTEIN (RIBONUCLEASE, SEMINAL) I	11BA:B
7	GAL4 (CD)	125D:A
8	CARBONIC ANHYDRASE II	12CA:A
9	IGG1-KAPPA 2E8 FAB (HEAVY CHAIN)	12E8:P
10	IGG1-KAPPA 2E8 FAB (LIGHT CHAIN)	12E8:M
11	HEN EGG WHITE LYSOZYME	132L:A
12	HUMAN LYSOZYME	133L:A
13	3-PHOSPHOGLYCERATE KINASE	13PK:A
14	T4 LYSOZYME	149L:A
15	PHOSPHOPEPTIDE	14PS:R
16	14-3-3 PROTEIN ZETA/DELTA	14PS:B
17	T4 LYSOZYME	150L:D
18	CYTOCHROME C551	151C:A
19	T4 LYSOZYME	152L:A
20	GOOSE LYSOZYME	153L:A
21	CYTOCHROME C550	155C:A
22	PROTEIN (VP16, VMW65, ATIF)	16VP:A
23	PROTEIN (NINE-HAEM CYTOCHROME C)	19HC:B
24	HEMOGLOBIN (BETA CHAIN)	1A00:D
25	HEMOGLOBIN (ALPHA CHAIN)	1A00:C
26	CALCYCLIN (RABBIT, CA2+)	1A03:B
27	NITRATE/NITRITE RESPONSE REGULATOR PROTEIN NARL	1A04:B
28	3-ISOPROPYLMALATE DEHYDROGENASE	1A05:B

No	Nama	Kode PDB
29	PROTEIN (PHOSPHATE SYSTEM POSITIVE REGULATORY PROTEIN PHO4)	1A0A:A
30	XYLOSE ISOMERASE	1A0D:A
31	XYLOSE ISOMERASE	1A0E:D
32	D-AMINO ACID AMINOTRANSFERASE	1A0G:B
33	MEIZOTHROMBIN	1A0H:E
34	MEIZOTHROMBIN	1A0H:D
35	DNA LIGASE	1A0I:A
36	CHEA	1A0O:H
37	SITE-SPECIFIC RECOMBINASE XERD	1A0P:A
38	29G11 FAB (LIGHT CHAIN)	1A0Q:L
39	TRANSDUCIN (BETA SUBUNIT)	1A0R:B
40	SUCROSE-SPECIFIC PORIN	1A0S:R
41	ALCALASE (SUBTILISIN CARLSBERG VARIANT) IN COMPLEX WITH 2 CHYMOTRYPSIN INHIBITOR CI-2A(M59P)	1A10:I
42	ALCALASE (SUBTILISIN CARLSBERG VARIANT) IN COMPLEX WITH 2 CHYMOTRYPSIN INHIBITOR CI-2A(M59P)	1A10:E
43	REGULATOR OF CHROMOSOME CONDENSATION 1	1A12:C
44	AMINOPEPTIDASE P	1A16:A
45	ADIPOCYTE LIPID BINDING PROTEIN	1A18:A
46	QGSR ZINC FINGER PEPTIDE	1A1H:A
47	HLA CLASS I HISTOCOMPATIBILITY ANTIGEN, BW-53 B*5301 ALPHA CHAIN	1A1M:A
48	BETA-2-MICROGLOBULIN	1A1N:B
49	ORNITHINE CARBAMOYLTRANSFERASE	1A1S:A

Lampiran 2. Data Pembelajaran Sistem

Protein-protein yang dijadikan sebagai data pembelajaran, ditunjukkan pada tabel 7.2.

Tabel 7.2 Data Pembelajaran Sistem

No	Nama	Kode PDB
1	DCOH (DIMERIZATION COFACTOR OF HNF-1)	1DCH:A
2	ETR1 PROTEIN	1DCF:A
3	PROTEIN (RAB GERANYLGERANYLTRANSFERASE ALPHA SUBUNIT)	1DCE:A
4	PROTEIN (RAB GERANYLGERANYLTRANSFERASE BETA SUBUNIT)	1DCE:D
5	CYTOCHROME C PEROXIDASE	1DCC:A
6	GLYCERALDEHYDE-3-PHOSPHATE DEHYDROGENASE	1DC6:A
7	CYCLIN-DEPENDENT KINASE 4 INHIBITOR A (P16INK4A)	1DC2:A
8	FRUCTOSE-1,6-BISPHOSPHATASE	1DBZ:A
9	DETHIOBIOTIN SYNTHETASE	1DBS:A
10	HYPOXANTHINE GUANINE XANTHINE PHOSPHORIBOSYLTRANSFERASE	1DBR:A
11	D-RIBOSE-BINDING PROTEIN	1DBP:A
12	PURINE REPRESSOR	1DBQ:B
13	IGG1-KAPPA DB3 FAB (LIGHT CHAIN)	1DBM:L
14	IGG1-KAPPA DB3 FAB (HEAVY CHAIN)	1DBM:H
15	AK.1 SERINE PROTEASE	1DBI:A
16	PROTEIN (HUMAN SOS 1)	1DBH:A
17	PROTEIN (PHOSPHOLIPASE A2)	1DB5:A
18	PROTEIN (CHORISMATE MUTASE)	1DBF:C
19	REGULATORY PROTEIN E2	1DBD:A
20	CATABOLITE GENE ACTIVATOR PROTEIN	1DBC:A
21	PLASMINOGEN ACTIVATOR INHIBITOR-1	1DB2:A
22	HIV-1 PROTEASE (RETROPEPSIN)	1DAZ:D
23	SOLUBLE TISSUE FACTOR	1DAN:U
24	EXCINUCLEASE UVRABC COMPONENT UVRB	1D9X:A

No	Nama	Kode PDB
25	GLUCOSAMINE-6-PHOSPHATE ISOMERASE	1D9T:A
26	MHC I-AK A CHAIN (ALPHA CHAIN)	1D9K:G
27	L-RHAMNOSE ISOMERASE	1D8W:A
28	STROMELYSIN-1 PRECURSOR	1D8M:B
29	mRNA TRIPHOSPHATASE CET1	1D8H:A
30	FARNESYLTRANSERASE (ALPHA SUBUNIT)	1D8E:A
31	HUMAN ORNITHINE DECARBOXYLASE	1D7K:A
32	CYCLODEXTRIN GLUCANOTRANSFERASE	1D7F:A
33	PROTEIN (TETANUS TOXIN HC)	1D6Z:B
34	DNA TOPOISOMERASE III	1D6M:A
35	CHALCONE SYNTHASE	1D6I:A
36	PROTEIN (TETANUS TOXIN HC)	1D5A
37	C. ELEGANS ACTIN 1/3	1D4X:A
38	FLAVOCYTOCHROME C FUMARATE REDUCTASE	1D4E:A
39	COLLAGEN ADHESIN	1D2P:A
40	ELONGATION FACTOR TU (EF-TU)	1D2E:B
41	BOVINE ENDOTHELIAL NITRIC OXIDE SYNTHASE HEME DOMAIN	1D1Y:A
42	MYOSIN	1D1C:A
43	PROTEIN (TETANUS TOXIN HC)	1D0H:A
44	TUMOR NECROSIS FACTOR RECEPTOR ASSOCIATED PROTEIN 2	1D0A:F
45	ASPARTATE CARBAMOYLTRANSFERASE CATALYTIC CHAIN	1D09:A
46	TUMOR NECROSIS FACTOR RECEPTOR ASSOCIATED PROTEIN 2	1CZZ:C
47	CHYMOSIN	1CZI:E
48	CYCLODEXTRIN GLUCANOTRANSFERASE	1CYG:A
49	DNA TOPOISOMERASE I	1CY6:A
50	DIMETHYL SULFOXIDE REDUCTASE	1CXT:A

Lampiran 3. Data Uji Coba Sistem

Protein-protein yang dijadikan sebagai data uji coba, ditunjukkan pada tabel 7.3.

Tabel 7.3 Data Uji Coba Sistem

No	Nama	Kode PDB
1	CRAMBIN	1CRN
2	D-GLYCERALDEHYDE-3-PHOSPHATE DEHYDROGENASE	1GPD
3	HUMAN LYSOZYME	1LHI
4	BETA-PUROTHIONIN	1BHP
5	COPPER,ZINC SUPEROXIDE DISMUTASE	2SOD
6	CALCIUM-DEPENDENT PROTEASE, SMALL SUBUNIT	1NX2:A
7	ADENYLATE KINASE	3ADK
8	RHODANESE	1RHD
9	NEUROTOXIN B	1NXB
10	OVOMUCOID THIRD DOMAIN	1OVO

Lampiran 4. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 50

Tabel 7.4. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 50

No	Kode PDB	Qh	Qe	Qc	Q3
1	1CRN	45.45454545	50	70	56.52173913
2	1GPD	48.27586207	40.57971014	53.67231638	49.54954955
3	1LH1	46.21848739	0	41.17647059	45.09803922
4	1BHP	64.70588235	0	40.90909091	44.44444444
5	2SOD	66.66666667	15.51724138	65.55555556	46.35761589
6	1NX2_A	35.45454545	44.44444444	62.96296296	44.50867052
7	3ADK	57.62711864	0	26.31578947	45.36082474
8	1RHD	54.02298851	40.625	51.14942529	50.85324232
9	1NXB	0	11.53846154	63.88888889	41.93548387
10	1OVO	20	83.33333333	38.23529412	44.64285714
Rata-rata		48.71401073	40.86259869	51.38657942	46.92724668

Lampiran 5. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 100

Tabel 7.5. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 100

No	Kode PDB	Qh	Qe	Qc	Q3
1	1CRN	45.45454545	50	70	56.52173913
2	1GPD	48.27586207	40.57971014	53.10734463	49.24924925
3	1LH1	46.21848739	0	41.17647059	45.09803922
4	1BHP	64.70588235	0	40.90909091	44.44444444
5	2SOD	66.66666667	15.51724138	64.44444444	45.69536424
6	1NX2_A	35.45454545	44.44444444	62.96296296	44.50867052
7	3ADK	57.62711864	0	26.31578947	45.36082474
8	1RHD	54.02298851	40.625	51.14942529	50.85324232
9	1NXB	0	11.53846154	61.11111111	40.32258065
10	1OVO	20	83.33333333	38.23529412	44.64285714
Rata-rata		48.71401073	40.86259869	50.94119335	46.66970117

Lampiran 6. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 150

Tabel 7.6. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 150

No	Kode PDB	Qh	Qe	Qc	Q3
1	1CRN	45.45454545	50	70	56.52173913
2	1GPD	48.27586207	40.57971014	53.67231638	49.54954955
3	1LH1	46.21848739	0	41.17647059	45.09803922
4	1BHP	64.70588235	0	40.90909091	44.44444444
5	2SOD	66.66666667	15.51724138	64.44444444	45.69536424
6	1NX2_A	35.45454545	44.44444444	62.96296296	44.50867052
7	3ADK	57.62711864	0	26.31578947	45.36082474
8	1RHD	54.02298851	40.625	52.29885057	51.53583618
9	1NXB	0	11.53846154	61.11111111	40.32258065
10	1OVO	20	83.33333333	38.23529412	44.64285714
Rata-rata		48.71401073	40.86259869	51.11263306	46.76799058

Lampiran 7. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 200

Tabel 7.7. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* 200

No	Kode PDB	Qh	Qe	Qc	Q3
1	1CRN	45.45454545	50	70	56.52173913
2	1GPD	48.27586207	40.57971014	53.10734463	49.24924925
3	1LH1	46.21848739	0	41.17647059	45.09803922
4	1BHP	64.70588235	0	40.90909091	44.44444444
5	2SOD	66.66666667	15.51724138	64.44444444	45.69536424
6	1NX2_A	35.45454545	44.44444444	62.96296296	44.50867052
7	3ADK	57.62711864	0	26.31578947	45.36082474
8	1RHD	54.02298851	40.625	52.29885057	51.53583618
9	1NXB	0	11.53846154	63.88888889	41.93548387
10	1OVO	20	83.33333333	38.23529412	44.64285714
Rata-rata		48.71401073	40.86259869	51.33391366	46.89925087

Lampiran 8. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* Sampai Konvergen

Tabel 7.8. Hasil Penghitungan Keakurasian Prediksi Struktur Sekunder Protein Dengan Iterasi Maksimum *Training* Sampai Konvergen

No	Kode PDB	Qh	Qe	Qc	Q3
1	1CRN	45.45454545	50	70	56.52173913
2	1GPD	48.27586207	40.57971014	53.10734463	49.24924925
3	1LH1	46.21848739	0	41.17647059	45.09803922
4	1BHP	64.70588235	0	40.90909091	44.44444444
5	2SOD	66.66666667	15.51724138	64.44444444	45.69536424
6	1NX2_A	35.45454545	44.44444444	62.96296296	44.50867052
7	3ADK	57.62711864	0	26.31578947	45.36082474
8	1RHD	54.02298851	40.625	51.14942529	50.85324232
9	1NXB	0	11.53846154	63.88888889	41.93548387
10	1OVO	20	83.33333333	38.23529412	44.64285714
Rata-rata		48.71401073	40.86259869	51.21897113	46.83099149