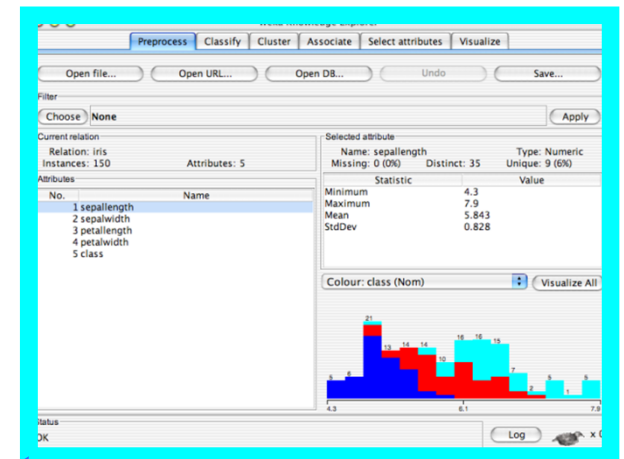


Neural Networks and SVM



Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file...

Open URL...

Open DB...

Undo

Edit...

Save...

Filter

Choose **None**

Apply

Current relation

Relation: None

Instances: None

Attributes: None

Selected attribute

Name: None

Missing: None

Distinct: None

Type: None

Unique: None

Attributes

All

None

Invert

Remove

Visualize All

Status

Welcome to the Weka Explorer

Log

 x 0

Open file...

Open URL...

Open DB...

Undo

Edit...

Save...

Filter

Choose

None

Apply

Current relation

Relation: None

Instances: None

Attributes: None

Selected attribute

Name: None

Missing: None

Distinct: None

Type: None

Unique: None

Attributes

All

None

Invert

Remove



Visualize All

Status

Welcome to the Weka Explorer

Log

 x 0

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file...

Open URL...

Open DB...

Undo

Edit...

Save...

Filter

Choose

None

Apply

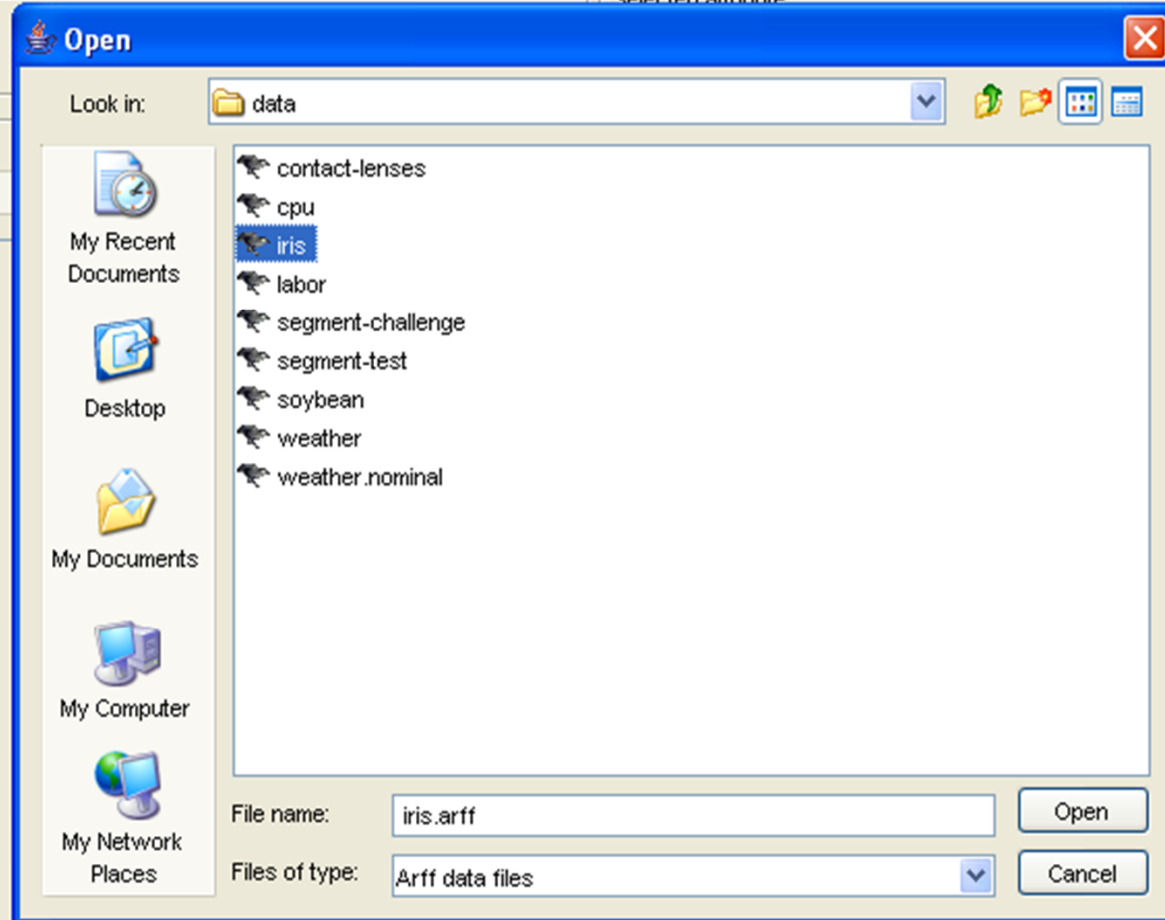
Current relation

Relation: None

Instances: None

Attributes

All



Type: None

Unique: None

Visualize All

Remove

Status

Welcome to the Weka Explorer

Log

x 0

Preprocess Classify Cluster Associate Select attributes Visualize

Open file...

Open URL...

Open DB...

Undo

Edit...

Save...

Filter

Choose **None**

Apply

Current relation

Relation: iris
Instances: 150

Attributes: 5

Attributes

All

None

Invert

No.	Name
1	<input checked="" type="checkbox"/> sepallength
2	<input type="checkbox"/> sepalwidth
3	<input type="checkbox"/> petallength
4	<input type="checkbox"/> petalwidth
5	<input type="checkbox"/> class

Remove

Selected attribute

Name: sepallength

Missing: 0 (0%)

Distinct: 35

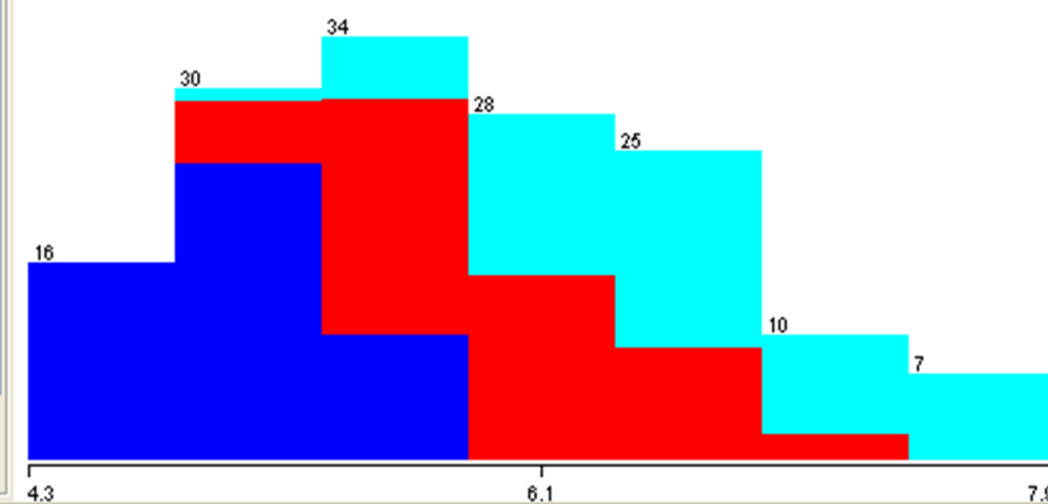
Type: Numeric

Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Class: class (Nom)

Visualize All



Status

OK

Log

x 0

Preprocess **Classify** Cluster Associate Select attributes Visualize

Open file...

Open URL...

Open DB...

Undo

Edit...

Save...

Filter

Choose

None

Apply

Current relation

Relation: iris

Instances: 150

Attributes: 5

Attributes

All

None

Invert

No.	Name
1	<input checked="" type="checkbox"/> sepallength
2	<input type="checkbox"/> sepalwidth
3	<input type="checkbox"/> petallength
4	<input type="checkbox"/> petalwidth
5	<input type="checkbox"/> class

Remove

Selected attribute

Name: sepallength

Missing: 0 (0%)

Distinct: 35

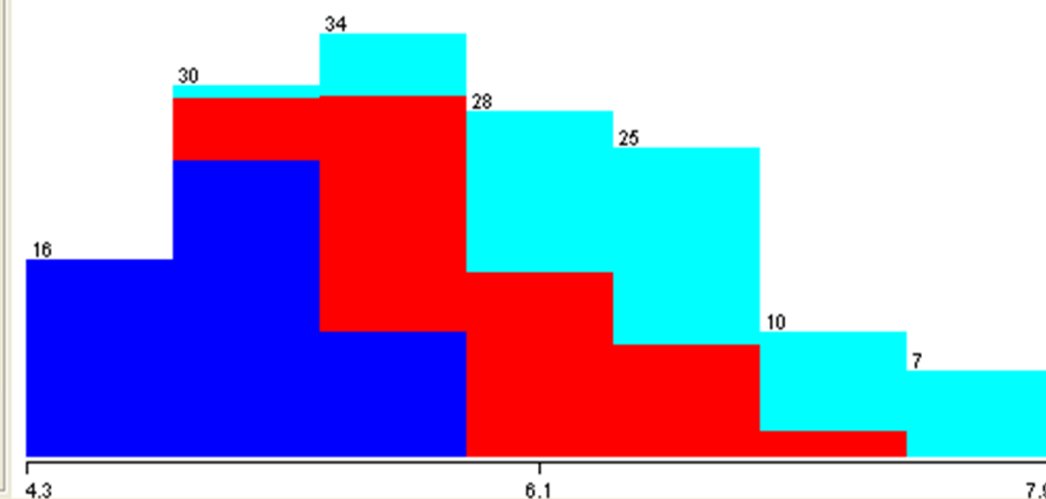
Type: Numeric

Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Class: class (Nom)

Visualize All



Status

OK

Log

x 0

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **ZeroR**

Test options

 Use training set Supplied test set

Set...

 Cross-validation

Folds

10

 Percentage split

%

66

More options...

Classifier output

(Nom) class



Start

Stop

Result list (right-click for options)

Status

OK

Log



x 0

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **ZeroR**

Test options

- Use training set
- Supplied test set
- Cross-validation Folds
- Percentage split %
-

Classifier output

(Nom) class

Result list (right-click for options)

Status

OK

x 0

Classifier

- weka
 - classifiers
 - bayes
 - functions
 - LeastMedSq
 - LinearRegression
 - Logistic
 - MultilayerPerceptron**
 - PaceRegression
 - RBFNetwork
 - SimpleLinearRegression
 - SimpleLogistic
 - SMO
 - SMOreg
 - VotedPerceptron
 - Winnow
 - lazy
 - meta
 - trees
 - rules

0 -S 0 -E 20 -H a

er output

Status

OK

Log

x 0

Preprocess Classify Cluster Associate Select attributes Visualize

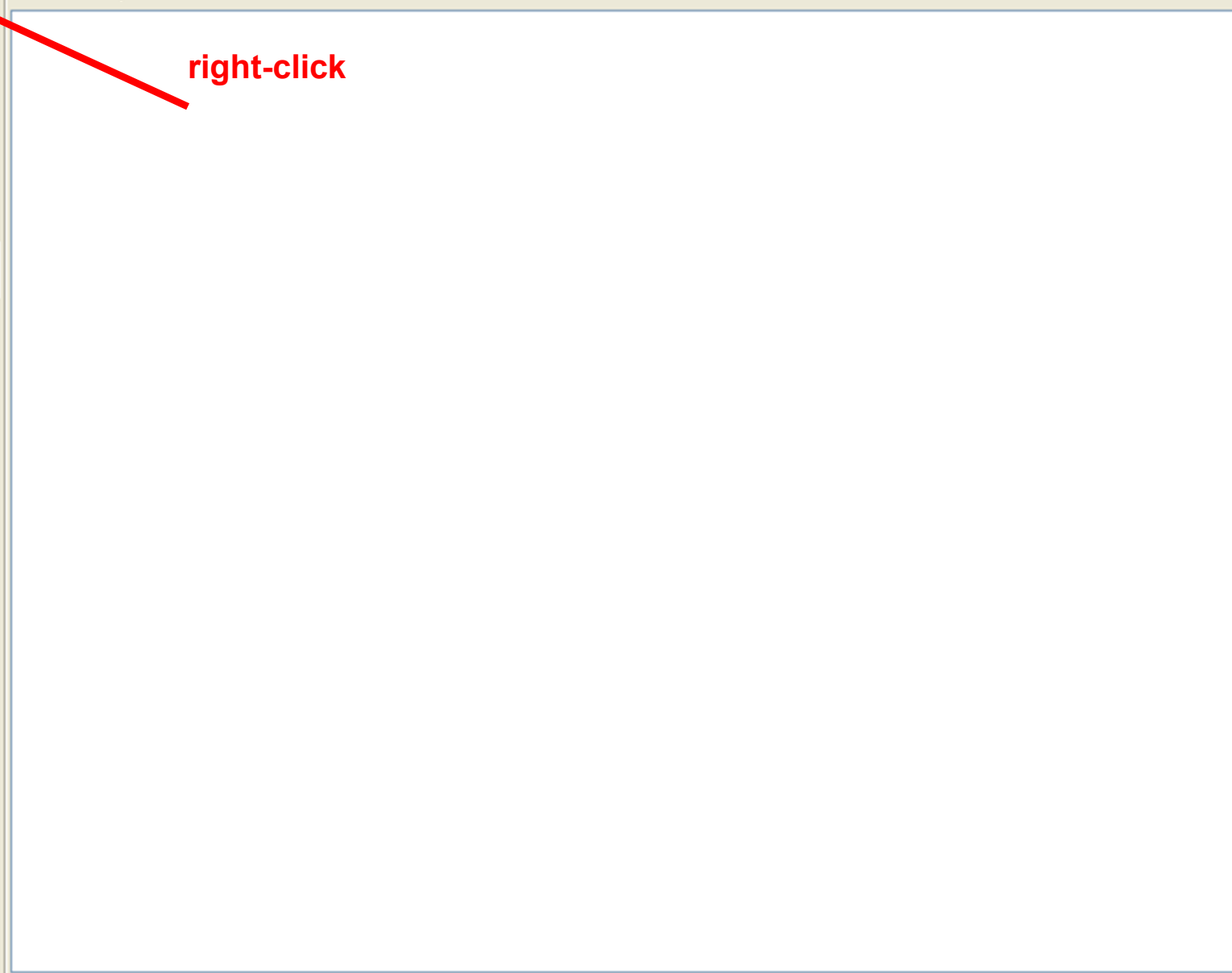
Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

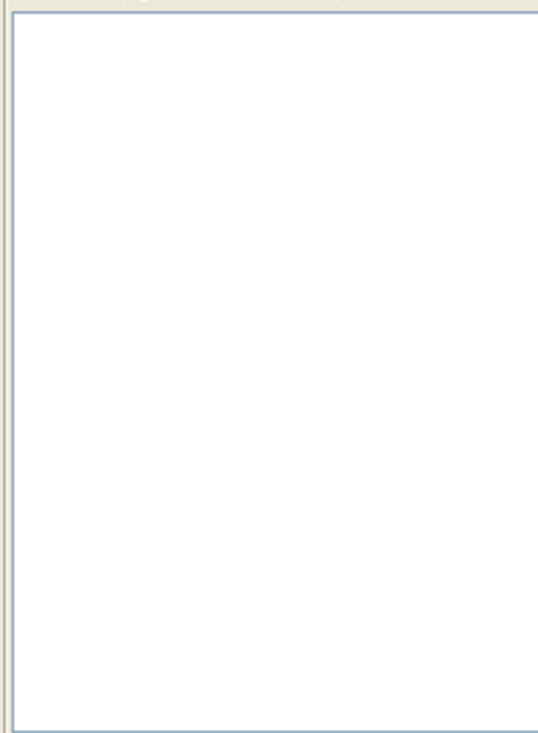
- Use training set
- Supplied test set
- Cross-validation Folds
- Percentage split %
-

Classifier output

**right-click**

(Nom) class

Result list (right-click for options)



Status

OK

x 0

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

- Use training set
 Supplied test set
 Cross-validation
 Percentage split

Mo

(Nom) class

Start

Result list (right-click f

weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About

This neural network uses backpropagation to train. [More](#)

GUI	False
autoBuild	True
debug	False
decay	False
hiddenLayers	a
learningRate	0.3
momentum	0.2
nominalToBinaryFilter	True
normalizeAttributes	True
normalizeNumericClass	True
randomSeed	0
reset	True
trainingTime	500
validationSetSize	0
validationThreshold	20

Open... Save... OK Cancel

Status

OK

Log

x 0

Classifier: **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options:

Use training set for testing

Supplied test set

Cross-validation

Percent

(Nom) class:

Step:

Result list (right):

weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About

This neural network uses backpropagation to train.

GUI

autoBuild

debug

decay

hiddenLayers

learningRate

momentum

nominalToBinaryFilter

normalizeAttributes

normalizeNumericClass

randomSeed

reset

trainingTime

validationSetSize

validationThreshold

Information

NAME
weka.classifiers.functions.MultilayerPerceptron

SYNOPSIS
This neural network uses backpropagation to train.

OPTIONS
GUI -- Brings up a gui interface. This will allow the pausing and altering of the neural network during training.

* To add a node left click (this node will be automatically selected, ensure no other nodes were selected).

* To select a node left click on it either while no other node is selected or while holding down the control key (this toggles that node as being selected and not selected).

NAME

weka.classifiers.functions.MultilayerPerceptron

SYNOPSIS

This neural network uses backpropagation to train.

OPTIONS

GUI -- Brings up a gui interface. This will allow the pausing and altering of the neural network during training.

- * To add a node left click (this node will be automatically selected, ensure no other nodes were selected).
- * To select a node left click on it either while no other node is selected or while holding down the control key (this toggles that node as being selected and not selected).
- * To connect a node, first have the start node(s) selected, then click either the end node or on an empty space (this will create a new node that is connected with the selected nodes). The selection status of nodes will stay the same after the connection. (Note these are directed connections, also a connection between two nodes will not be established more than once and certain connections that are deemed to be invalid will not be made).
- * To remove a connection select one of the connected node(s) in the connection and then right click the other node (it does not matter whether the node is the start or end the connection will be removed).
- * To remove a node right click it while no other nodes (including it) are selected. (This will also remove all connections to it)
- * To deselect a node either left click it while holding down control, or right click on empty space.
- * The raw inputs are provided from the labels on the left.
- * The red nodes are hidden layers.
- * The orange nodes are the output nodes.
- * The labels on the right show the class the output node represents. Note that with a numeric class the output node will automatically be made into an unthresholded linear unit.

Alterations to the neural network can only be done while the network is not running, This also applies to the learning rate and other fields on the control panel.

- * You can accept the network as being finished at any time.
- * The network is automatically paused at the beginning.
- * There is a running indication of what epoch the network is up to and what the (rough) error for that epoch was (or for the validation if that is being used). Note that this error value is based on a network that changes as the value is computed. (also depending on whether the class is normalized will effect the error reported for numeric classes.
- * Once the network is done it will pause again and either wait to be accepted or trained more.

Note that if the gui is not set the network will not require any interaction.

autoBuild -- Adds and connects up hidden layers in the network.

debug -- If set to true, classification over output additional info to the console

`debug` -- If set to true, classifier may output additional info to the console.

`decay` -- This will cause the learning rate to decrease. This will divide the starting learning rate by the epoch number, to determine what the current learning rate should be. This may help to stop the network from diverging from the target output, as well as improve general performance. Note that the decaying learning rate will not be shown in the gui, only the original learning rate. If the learning rate is changed in the gui, this is treated as the starting learning rate.

`hiddenLayers` -- This defines the hidden layers of the neural network. This is a list of positive whole numbers. 1 for each hidden layer. Comma seperated. To have no hidden layers put a single 0 here. This will only be used if autobuild is set. There are also wildcard values 'a' = (attribs + classes) / 2, 'i' = attribs, 'o' = classes, 't' = attribs + classes.

`learningRate` -- The amount the weights are updated.

`momentum` -- Momentum applied to the weights during updating.

`nominalToBinaryFilter` -- This will preprocess the instances with the filter. This could help improve performance if there are nominal attributes in the data.

`normalizeAttributes` -- This will normalize the attributes. This could help improve performance of the network. This is not reliant on the class being numeric. This will also normalize nominal attributes as well (after they have been run through the nominal to binary filter if that is in use) so that the nominal values are between -1 and 1

`normalizeNumericClass` -- This will normalize the class if it's numeric. This could help improve performance of the network, It normalizes the class to be between -1 and 1. Note that this is only internally, the output will be scaled back to the original range.

`randomSeed` -- Seed used to initialise the random number generator. Random numbers are used for setting the initial weights of the connections between nodes, and also for shuffling the training data.

`reset` -- This will allow the network to reset with a lower learning rate. If the network diverges from the answer this will automatically reset the network with a lower learning rate and begin training again. This option is only available if the gui is not set. Note that if the network diverges but isn't allowed to reset it will fail the training process and return an error message.

`trainingTime` -- The number of epochs to train through. If the validation set is non-zero then it can terminate the network early

`validationSetSize` -- The percentage size of the validation set. (The training will continue until it is observed that the error on the validation set has been consistently getting worse, or if the training time is reached).

If This is set to zero no validation set will be used and instead the network will train for the specified number of epochs.

`validationThreshold` -- Used to terminate validation testing. The value here dictates how many times in a row the validation set error can get worse before training is terminated.

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

- Use train
 Supplied
 Cross-validated
 Percent

(Nom) class

Step

Result list (right)

weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About

This neural network uses backpropagation to train. [More](#)

GUI False

autoBuild True

debug False

decay False

hiddenLayers

learningRate

momentum

nominalToBinaryFilter True

normalizeAttributes True

normalizeNumericClass True

randomSeed

reset True

trainingTime

validationSetSize

validationThreshold

Information

NAME
weka.classifiers.functions.MultilayerPerceptron

SYNOPSIS
This neural network uses backpropagation to train.

OPTIONS
GUI -- Brings up a gui interface. This will allow the pausing and altering of the neural network during training.

* To add a node left click (this node will be automatically selected, ensure no other nodes were selected).

* To select a node left click on it either while no other node is selected or while holding down the control key (this toggles that node as being selected and not selected).

Status

OK

Log

x 0

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

 Use training set Supplied test set

Set...

 Cross-validation

Folds

10

 Percentage split

%

66

More options...

Classifier output

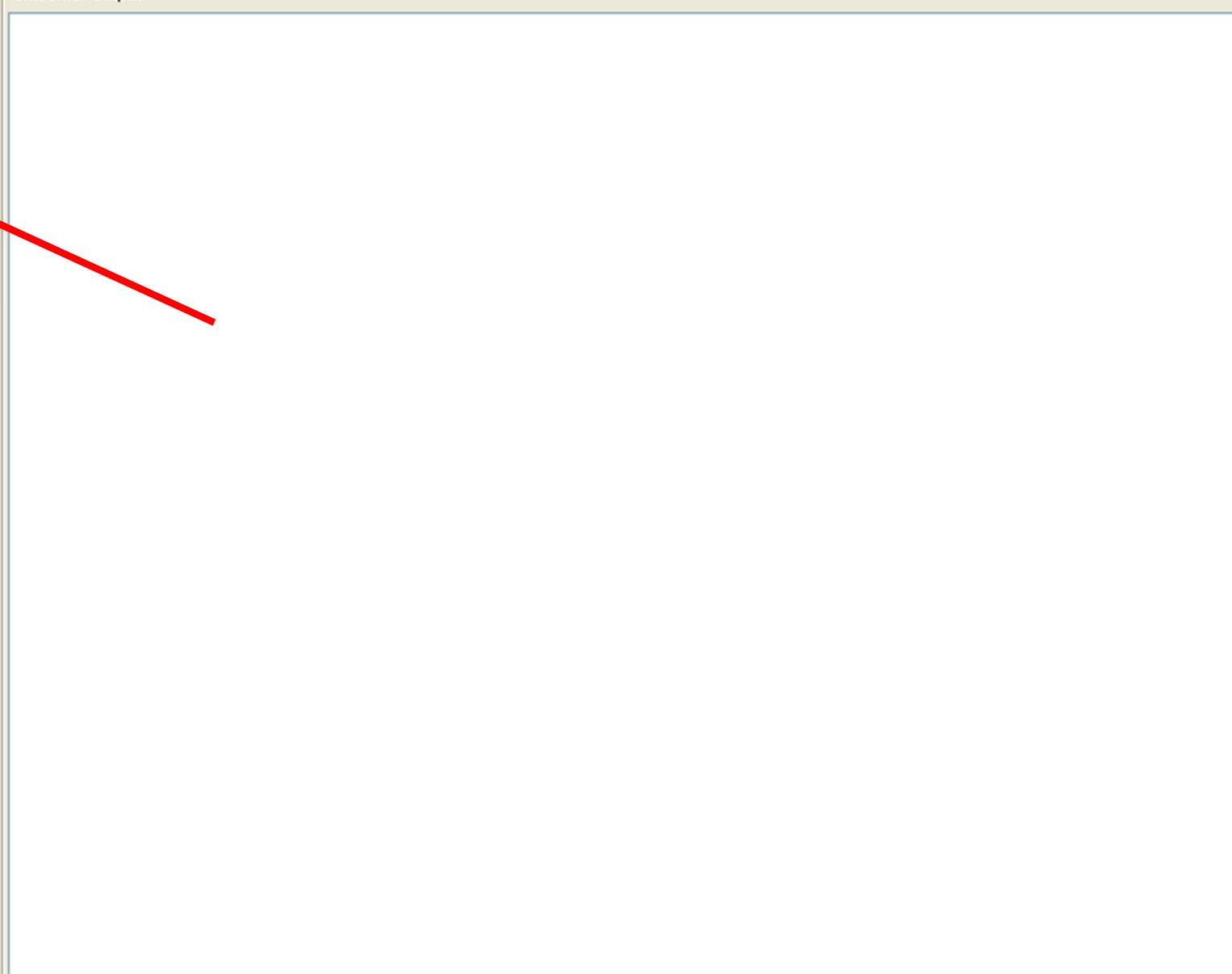
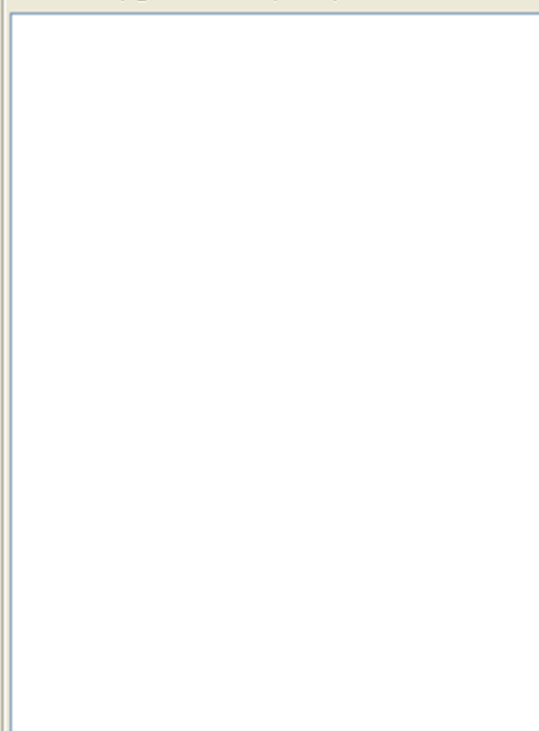
(Nom) class



Start

Stop

Result list (right-click for options)



Status

OK

Log

 x 0

Classifier

Choose **MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a**

Test options

 Use training set Supplied test set

Set...

 Cross-validation

Folds

10

 Percentage split

%

66

More options...

Classifier output

=== Run information ===

```

Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:    iris
Instances:   150
Attributes:  5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:   split 66% train, remainder test

```

=== Classifier model (full training set) ===

Sigmoid Node 0

Inputs	Weights
Threshold	-3.5015971588434005
Node 3	-1.0058110853859954
Node 4	9.07503844669134
Node 5	-4.107780453339232

Sigmoid Node 1

Inputs	Weights
Threshold	1.0692845992273172
Node 3	3.898873687789399
Node 4	-9.768910360340262
Node 5	-8.59913449315134

Sigmoid Node 2

Inputs	Weights
Threshold	-1.0071762383436442
Node 3	-4.21840613382704
Node 4	-3.6260596863211187

(Nom) class

Start

Stop

Result list (right-click for options)

09:17:47 - functions.MultilayerPerceptron

Status

OK

Log

x 0

Classifier Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

Use training set

Supplied test set

Cross-validation Folds

Percentage split %

(Nom) class

Result list (right-click for options)

09:17:47 - functions.MultilayerPerceptron

Classifier output

```

=== Run information ===
Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:    iris
Instances:   150
Attributes:  5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:   split 66% train, remainder test

=== Classifier model (full training set) ===

Sigmoid Node 0
  Inputs  Weights
  Threshold  -3.5015971588434005
  Node 3    -1.0058110853859954
  Node 4     9.07503844669134
  Node 5    -4.107780453339232

Sigmoid Node 1
  Inputs  Weights
  Threshold  1.0692845992273172
  Node 3     3.898873687789399
  Node 4    -9.768910360340262
  Node 5    -8.59913449315134

Sigmoid Node 2
  Inputs  Weights
  Threshold -1.0071762383436442
  Node 3   -4.21840613382704
  Node 4   -3.6260596863211187
  
```

right-click

Classifier: **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a -G -R

Test options:
 Use training data
 Supplied test data
 Cross-validation
 Percent of training data
[]

(Nom) class: []

Start

Result list (right-click to open):
09:17:47 - fu

weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About: This neural network uses backpropagation to train. [More](#)

GUI: **True** (highlighted with a red arrow)

autoBuild: True

debug: False

decay: False

hiddenLayers: a

learningRate: 0.3

momentum: 0.2

nominalToBinaryFilter: True

normalizeAttributes: True

normalizeNumericClass: True

randomSeed: 0

reset: False

trainingTime: 500

validationSetSize: 0

validationThreshold: 20

Buttons: Open... Save... **OK** Cancel

er test
t) ===

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a -G -R

Test options

 Use training set

 Supplied test set

Set...

 Cross-validation

Folds

10

 Percentage split

%

66

More options...

Classifier output

```
=== Run information ===
```

```
Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
```

```
Relation:    iris
```

```
Instances:   150
```

```
Attributes:  5
```

```
  sepallength
```

```
  sepalwidth
```

```
  petallength
```

```
  petalwidth
```

```
  class
```

```
Test mode:   split 66% train, remainder test
```

```
=== Classifier model (full training set) ===
```

```
Sigmoid Node 0
```

```
Inputs      Weights
```

```
Threshold   -3.5015971588434005
```

```
Node 3      -1.0058110853859954
```

```
Node 4      9.07503844669134
```

```
Node 5      -4.107780453339232
```

```
Sigmoid Node 1
```

```
Inputs      Weights
```

```
Threshold   1.0692845992273172
```

```
Node 3      3.898873687789399
```

```
Node 4      -9.768910360340262
```

```
Node 5      -8.59913449315134
```

```
Sigmoid Node 2
```

```
Inputs      Weights
```

```
Threshold   -1.0071762383436442
```

```
Node 3      -4.21840613382704
```

```
Node 4      -3.6260596863211187
```

(Nom) class

Start

Stop

Result list (right-click for options)

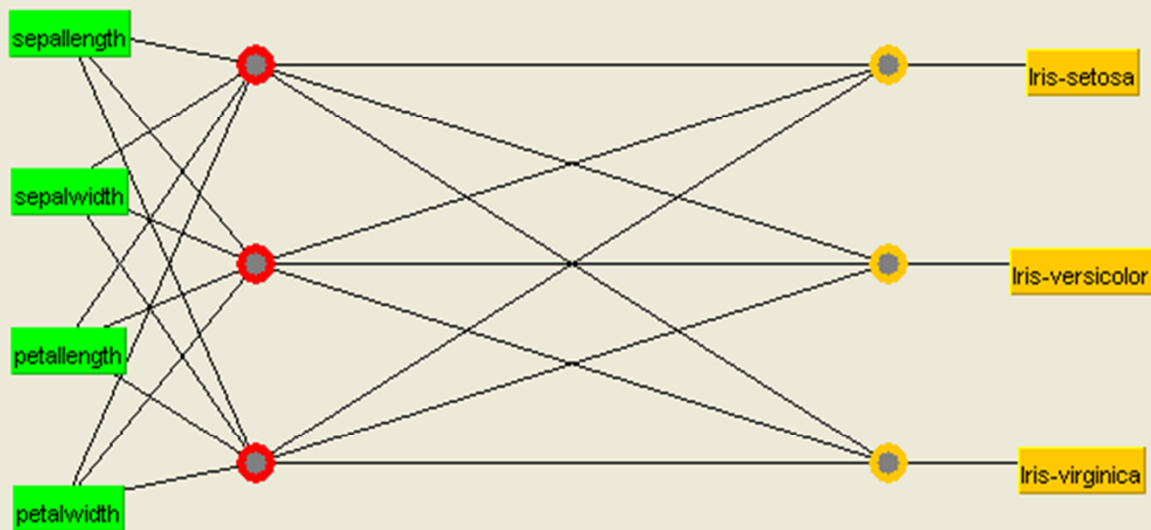
09:22:56 - functions.MultilayerPerceptron

Status

OK

Log

x 0



Controls

Epoch 0

Learning Rate = 0.3

Num Of Epochs 500

Momentum = 0.2

Error per Epoch = 0

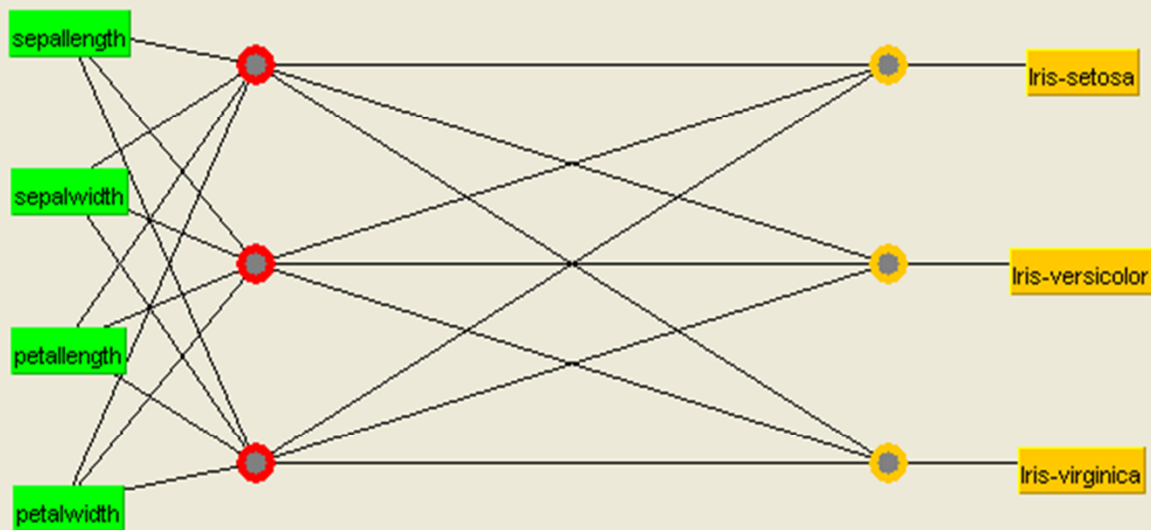
```
con -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a -G -R
```

Status

Building model on training data...



x 1



Controls

Start

Epoch 0

Accept

Num Of Epochs

500

Error per Epoch = 0

Learning Rate = 0.3

Momentum = 0.2

Class Iris-versicolor

Input

Node 1

Class Iris-virginica

Input

Node 2

Time taken to build model: 92.3 seconds

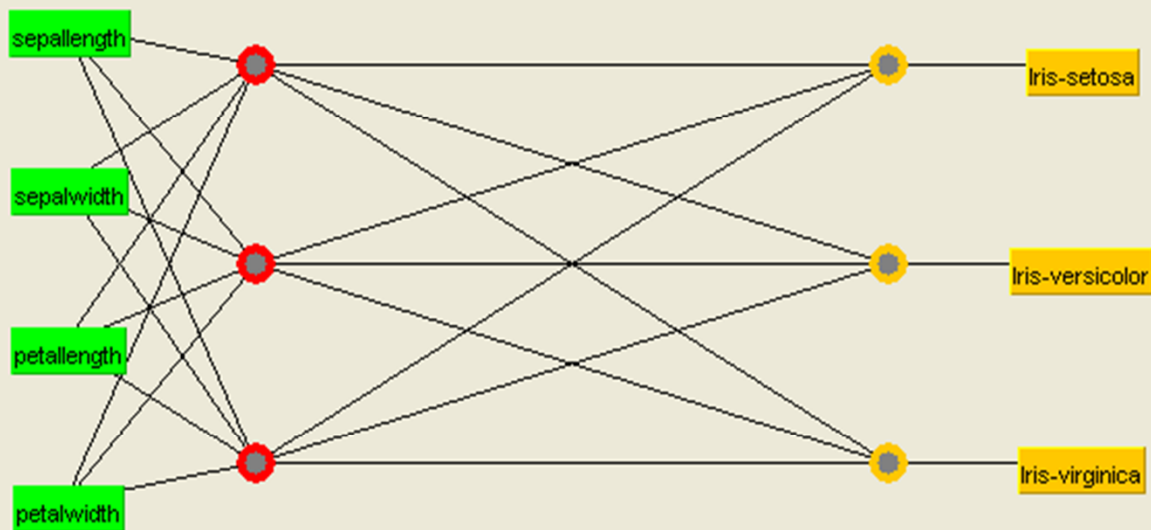
Status

Building model on training split...

Log



x 1



Controls

Epoch 500

Num Of Epochs

Learning Rate =

Error per Epoch = 0.0082655

Momentum =

Class Iris-versicolor

Input

Node 1

Class Iris-virginica

Input

Node 2

Time taken to build model: 92.3 seconds

Status

Building model on training split...



x 1

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a -G -R

Test options

 Use training set

 Supplied test set

Set...

 Cross-validation Folds
 Percentage split %

More options...

Classifier output

Time taken to build model: 92.3 seconds

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	50	98.0392 %
Incorrectly Classified Instances	1	1.9608 %
Kappa statistic	0.9704	
Mean absolute error	0.0239	
Root mean squared error	0.1101	
Relative absolute error	5.3594 %	
Root relative squared error	23.2952 %	
Total Number of Instances	51	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
1	0	1	1	1	Iris-setosa
1	0.031	0.95	1	0.974	Iris-versicolor
0.941	0	1	0.941	0.97	Iris-virginica

=== Confusion Matrix ===

```

a b c <-- classified as
15 0 0 | a = Iris-setosa
0 19 0 | b = Iris-versicolor
0 1 16 | c = Iris-virginica

```

(Nom) class

Start

Stop

Result list (right-click for options)

10:07:52 - functions.MultilayerPerceptron
 10:08:05 - functions.MultilayerPerceptron

Status

OK

Log

x 0

Try with different network topologies

Other Neural Network Software

- SNNS- Stuttgart Neural Network Simulator

<http://www-ra.informatik.uni-tuebingen.de/SNNS/>

- Alyuda NeuroIntelligence

<http://www.alyuda.com/neural-networks-software.htm>

- “backprop.cpp” from V. Rao and H. Rao

C++ Neural Network and Fuzzy Logic

Backpropagation Simulation version 1

Using “backprop.cpp”

- Download the executable file as well as the example training and testing data sets
- The training and data set were obtained from the UCI Machine Learning Repository’s Wine Recognition Database
- The original data set was randomly divided to create the training (2/3) and testing (1/3) data sets

Using "backprop.cpp"

- The training data set contains the input and output values.

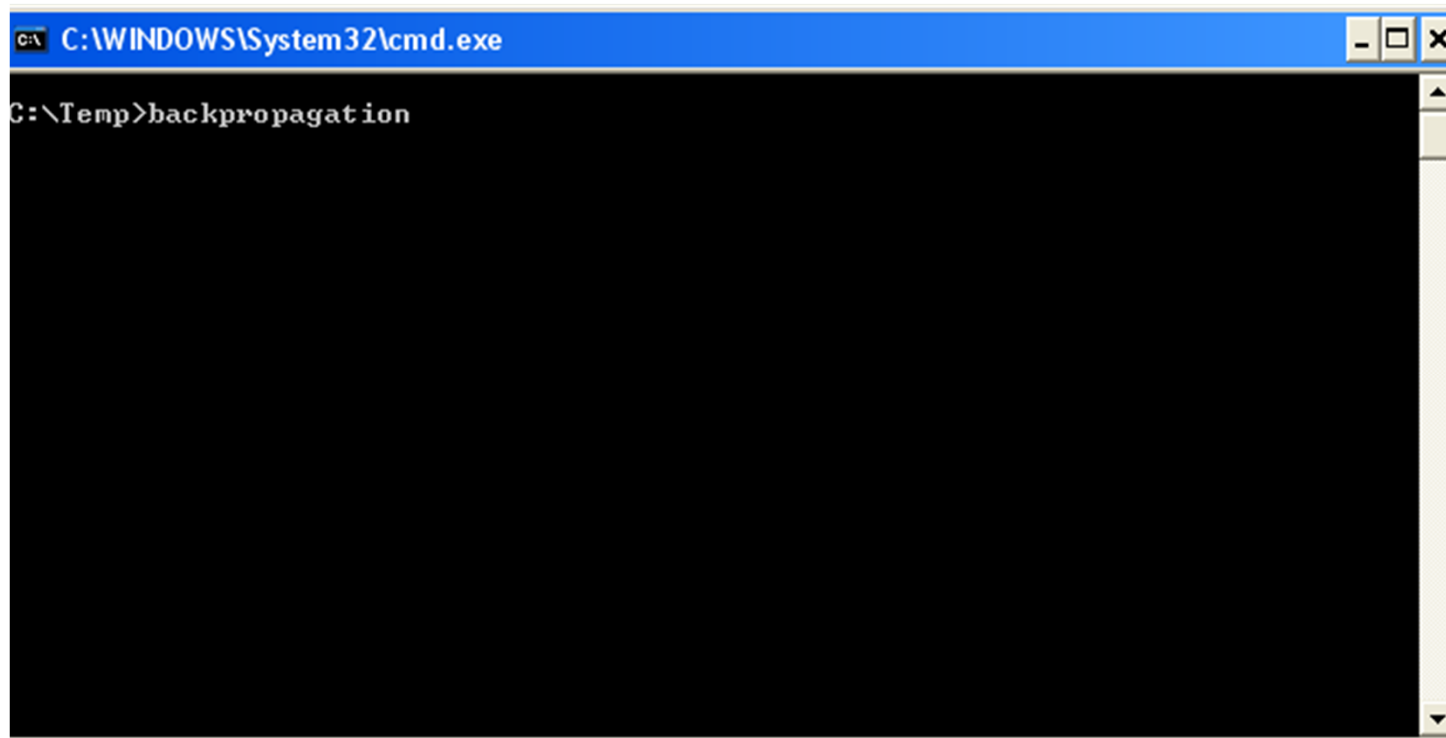
- The testing data set contains **only the input** values

Using "backprop.cpp"

- Please note the changes made to the output values for this particular example.
 - Class 1: output node 1 = 0; output node 2 = 0
 - Class 2: output node 1 = 0; output node 2 = 1
 - Class 3: output node 1 = 1; output node 2 = 1

Using “backprop.cpp” (TRAINING)

- On the DOS prompt, type “backpropagation” to run the application

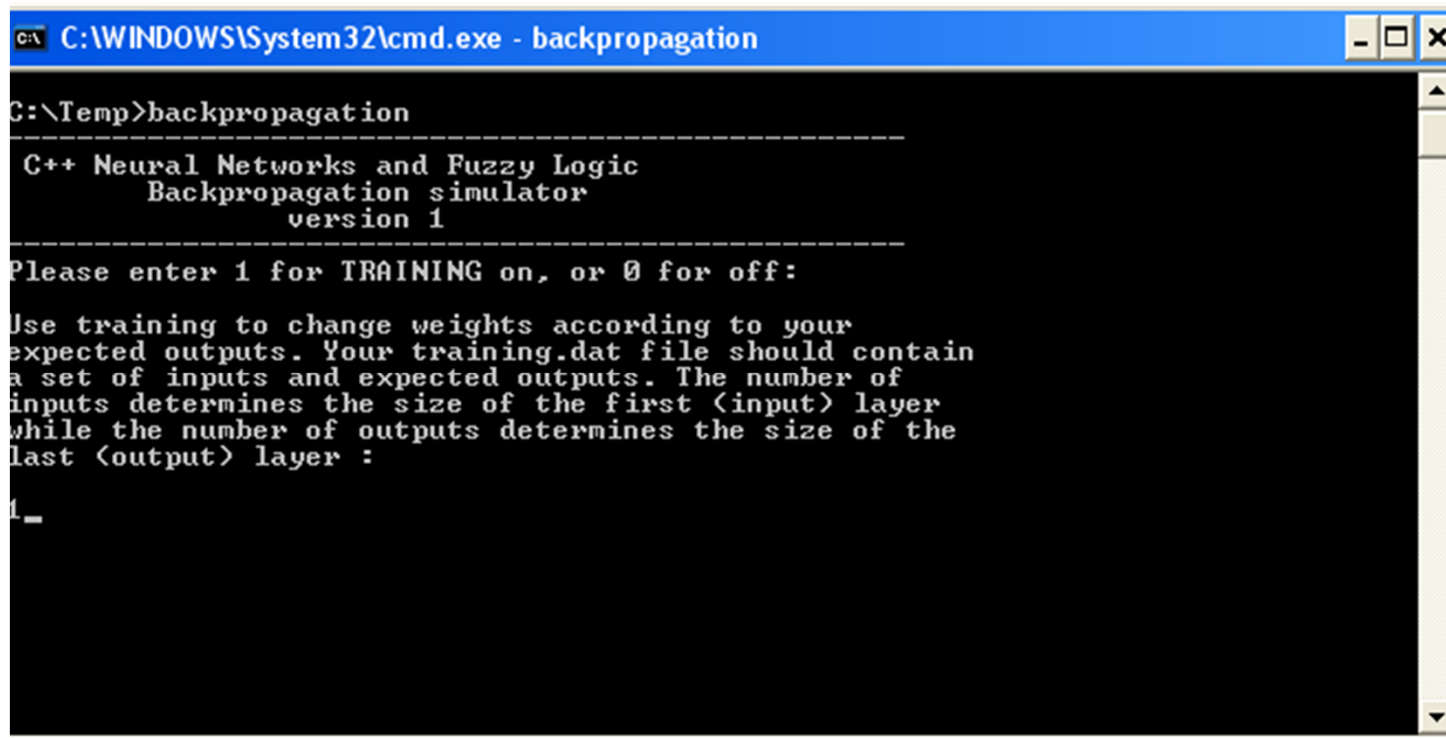


```
C:\WINDOWS\System32\cmd.exe
C:\Temp>backpropagation
```

The image shows a screenshot of a Windows command prompt window. The title bar at the top reads "C:\WINDOWS\System32\cmd.exe". The main area of the window is black with white text. The prompt "C:\Temp>" is visible, followed by the command "backpropagation" which has been entered. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Using "backprop.cpp" (TRAINING)

- Enter "1" for TRAINING



```
C:\WINDOWS\System32\cmd.exe - backpropagation

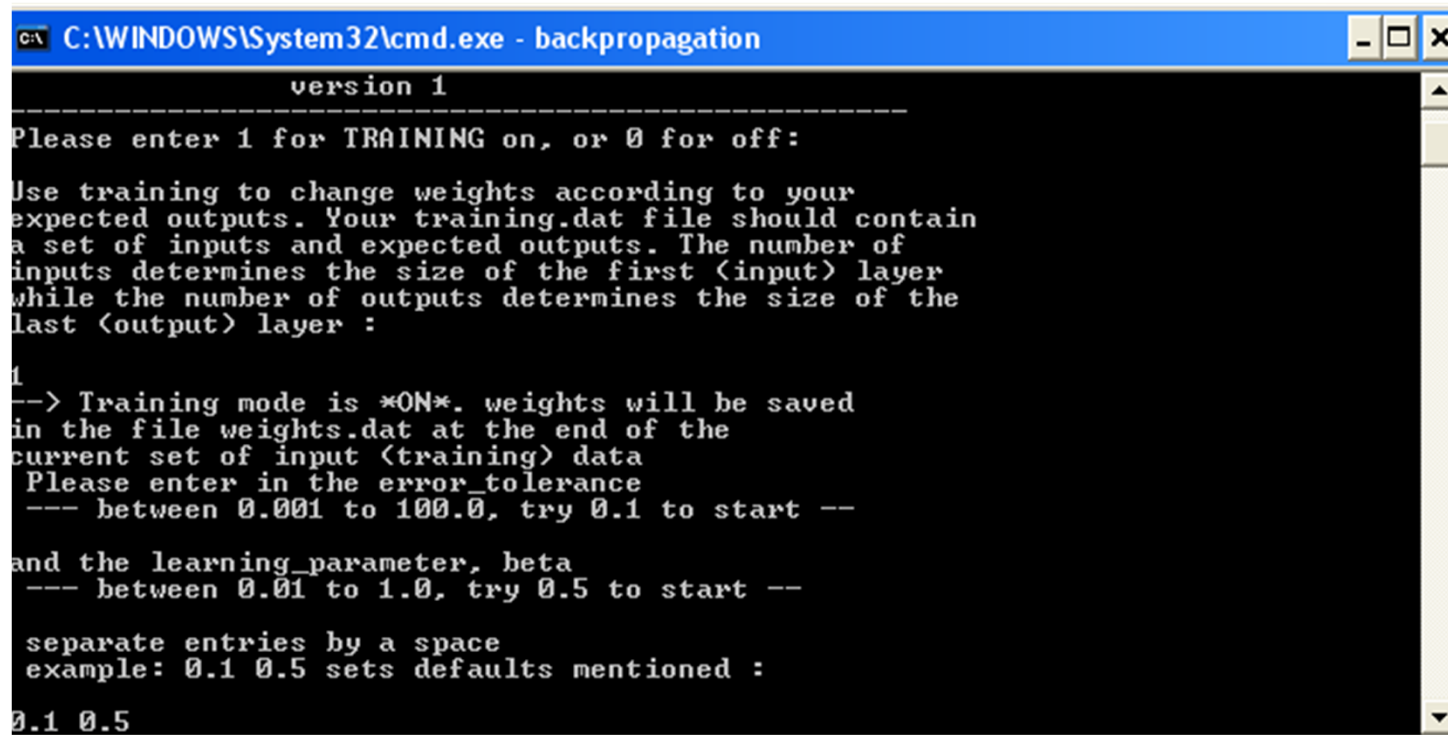
C:\Temp>backpropagation
-----
C++ Neural Networks and Fuzzy Logic
  Backpropagation simulator
    version 1
-----
Please enter 1 for TRAINING on, or 0 for off:

Use training to change weights according to your
expected outputs. Your training.dat file should contain
a set of inputs and expected outputs. The number of
inputs determines the size of the first (input) layer
while the number of outputs determines the size of the
last (output) layer :

1_
```


Using "backprop.cpp" (TRAINING)

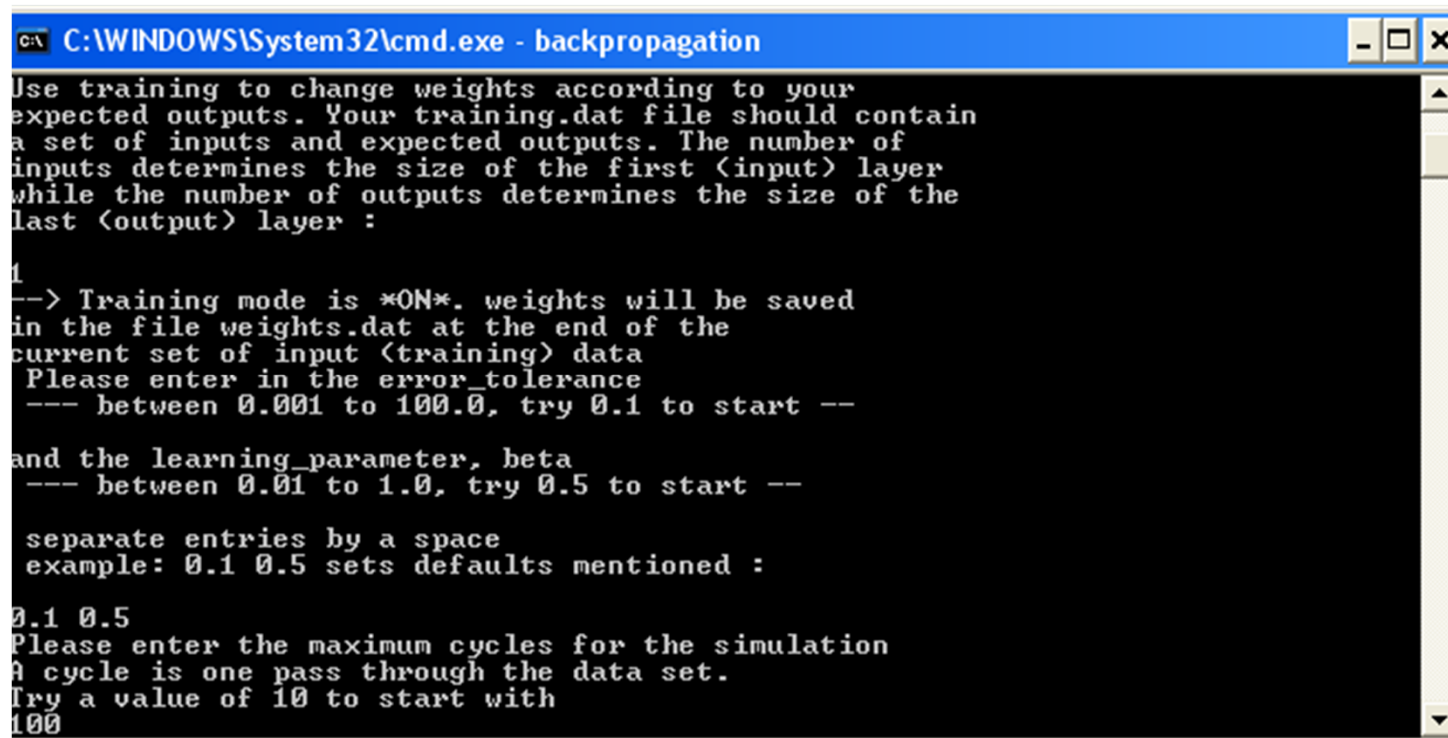
- Enter the error tolerance and the learning rate



```
C:\WINDOWS\System32\cmd.exe - backpropagation
version 1
-----
Please enter 1 for TRAINING on, or 0 for off:
1
Use training to change weights according to your
expected outputs. Your training.dat file should contain
a set of inputs and expected outputs. The number of
inputs determines the size of the first <input> layer
while the number of outputs determines the size of the
last <output> layer :
--> Training mode is *ON*. weights will be saved
in the file weights.dat at the end of the
current set of input <training> data
Please enter in the error_tolerance
--- between 0.001 to 100.0, try 0.1 to start ---
and the learning_parameter, beta
--- between 0.01 to 1.0, try 0.5 to start ---
separate entries by a space
example: 0.1 0.5 sets defaults mentioned :
0.1 0.5
```

Using "backprop.cpp" (TRAINING)

- Enter the maximum number of cycles. One cycle is one pass through the data set.



```
C:\WINDOWS\System32\cmd.exe - backpropagation
Use training to change weights according to your
expected outputs. Your training.dat file should contain
a set of inputs and expected outputs. The number of
inputs determines the size of the first <input> layer
while the number of outputs determines the size of the
last <output> layer :

1
--> Training mode is *ON*. weights will be saved
in the file weights.dat at the end of the
current set of input <training> data
Please enter in the error_tolerance
--- between 0.001 to 100.0, try 0.1 to start --

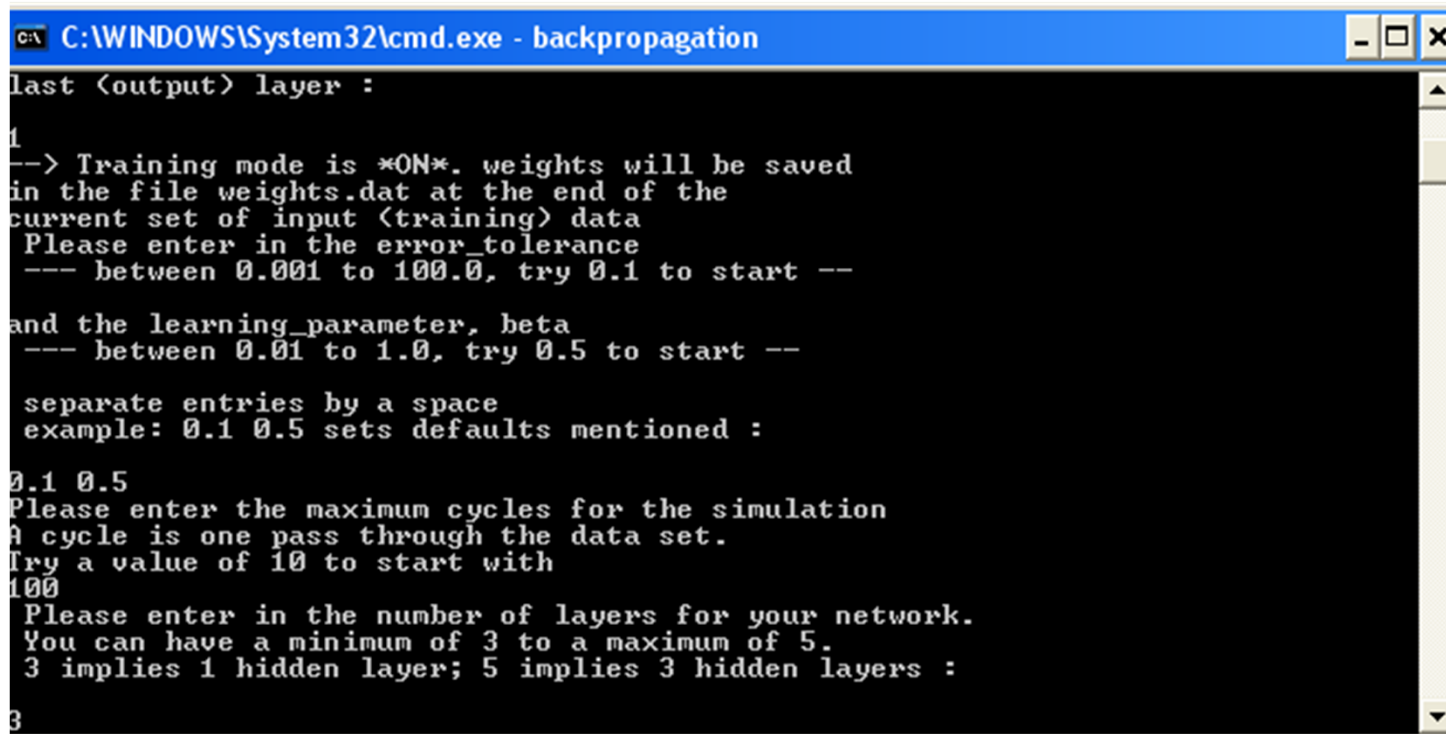
and the learning_parameter, beta
--- between 0.01 to 1.0, try 0.5 to start --

separate entries by a space
example: 0.1 0.5 sets defaults mentioned :

0.1 0.5
Please enter the maximum cycles for the simulation
A cycle is one pass through the data set.
Try a value of 10 to start with
100
```

Using "backprop.cpp" (TRAINING)

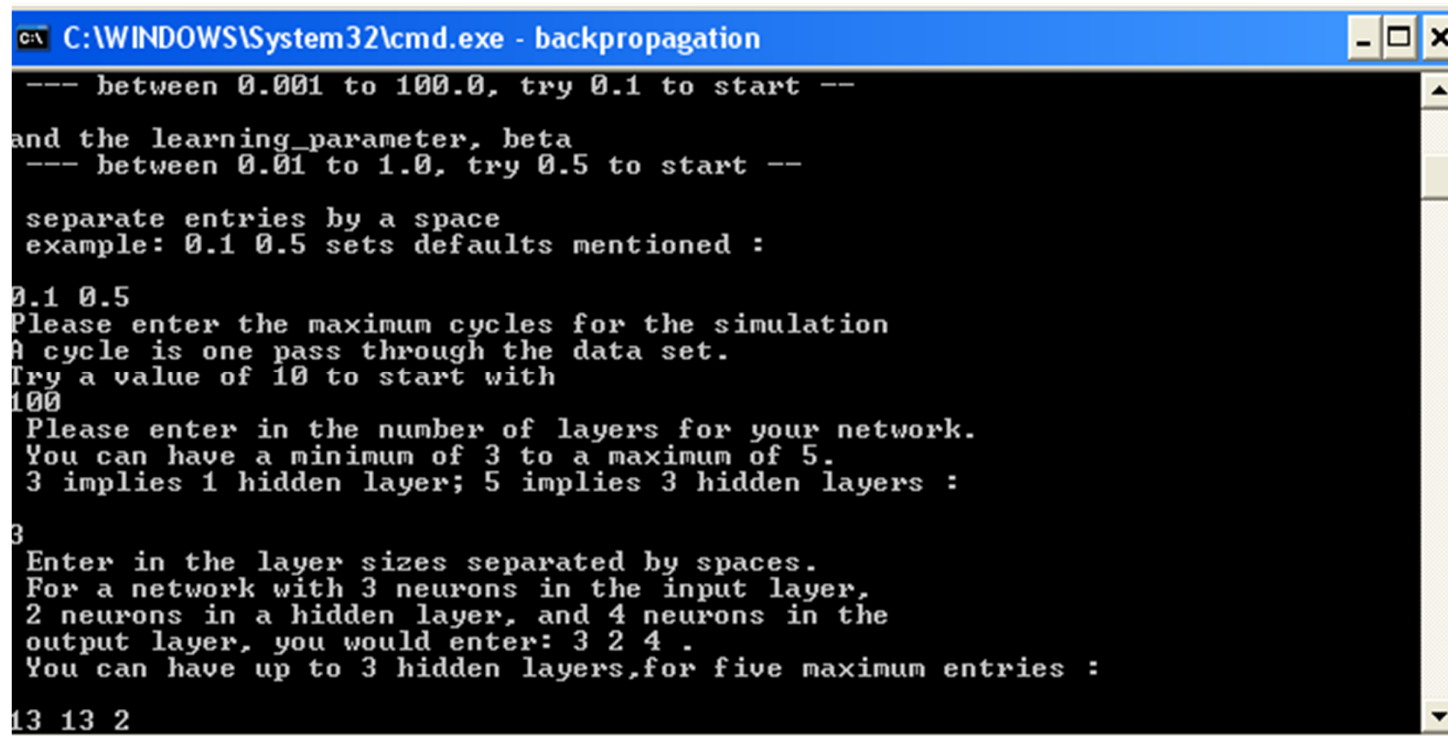
- Enter the number of layers for your network.



```
C:\WINDOWS\System32\cmd.exe - backpropagation
last (output) layer :
1
--> Training mode is *ON*. weights will be saved
in the file weights.dat at the end of the
current set of input (training) data
Please enter in the error_tolerance
--- between 0.001 to 100.0, try 0.1 to start --
and the learning_parameter, beta
--- between 0.01 to 1.0, try 0.5 to start --
separate entries by a space
example: 0.1 0.5 sets defaults mentioned :
0.1 0.5
Please enter the maximum cycles for the simulation
A cycle is one pass through the data set.
Try a value of 10 to start with
100
Please enter in the number of layers for your network.
You can have a minimum of 3 to a maximum of 5.
3 implies 1 hidden layer; 5 implies 3 hidden layers :
3
```

Using “backprop.cpp” (TRAINING)

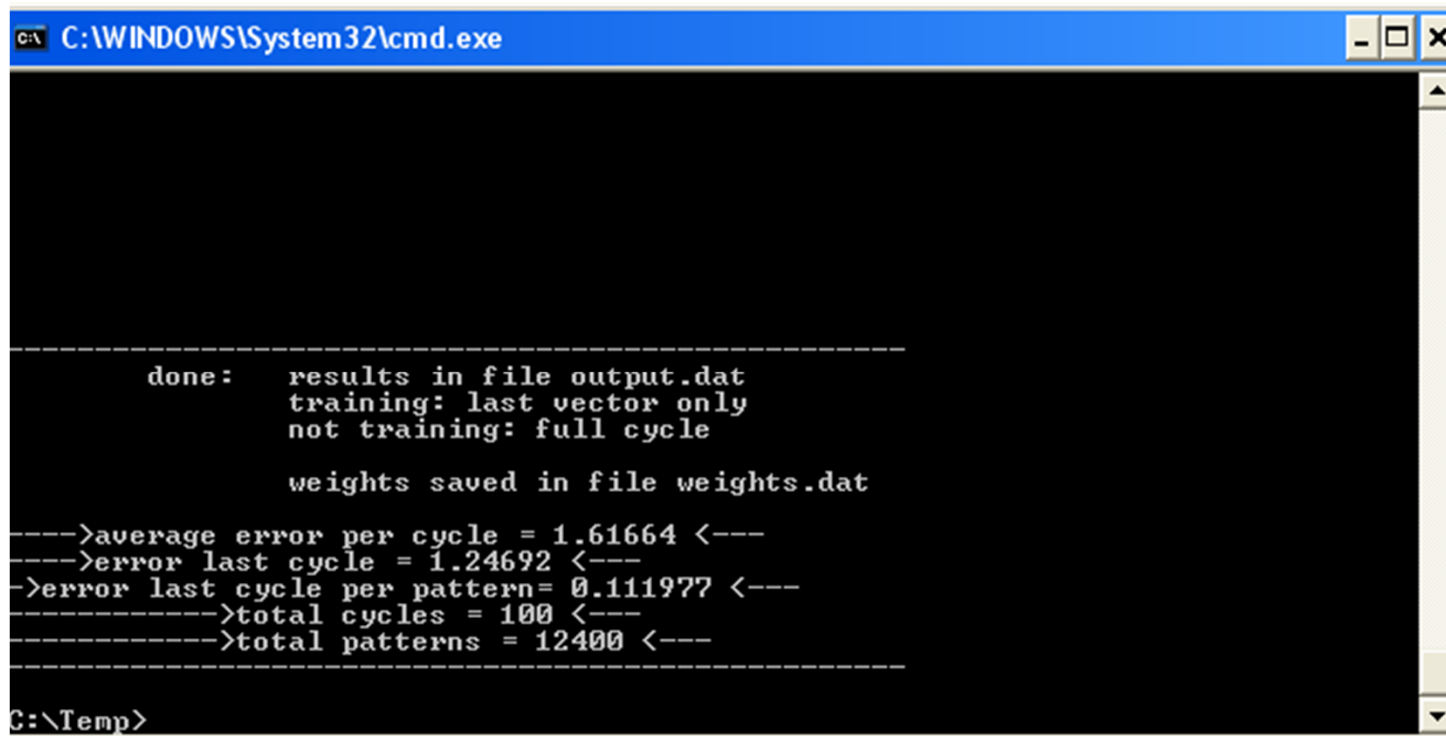
- Enter the layer sizes separated by spaces.
- **IMPORTANT:** the number of input and output nodes must match the training data (on the “training.dat” file)



```
C:\WINDOWS\System32\cmd.exe - backpropagation
--- between 0.001 to 100.0, try 0.1 to start --
and the learning_parameter, beta
--- between 0.01 to 1.0, try 0.5 to start --
separate entries by a space
example: 0.1 0.5 sets defaults mentioned :
0.1 0.5
Please enter the maximum cycles for the simulation
A cycle is one pass through the data set.
Try a value of 10 to start with
100
Please enter in the number of layers for your network.
You can have a minimum of 3 to a maximum of 5.
3 implies 1 hidden layer; 5 implies 3 hidden layers :
3
Enter in the layer sizes separated by spaces.
For a network with 3 neurons in the input layer,
2 neurons in a hidden layer, and 4 neurons in the
output layer, you would enter: 3 2 4 .
You can have up to 3 hidden layers, for five maximum entries :
13 13 2
```

Using “backprop.cpp” (TRAINING)

- Run the neural network. Weights were saved on the “weights.dat” file. They will be used later during testing



```
C:\WINDOWS\System32\cmd.exe

-----
done:   results in file output.dat
        training: last vector only
        not training: full cycle

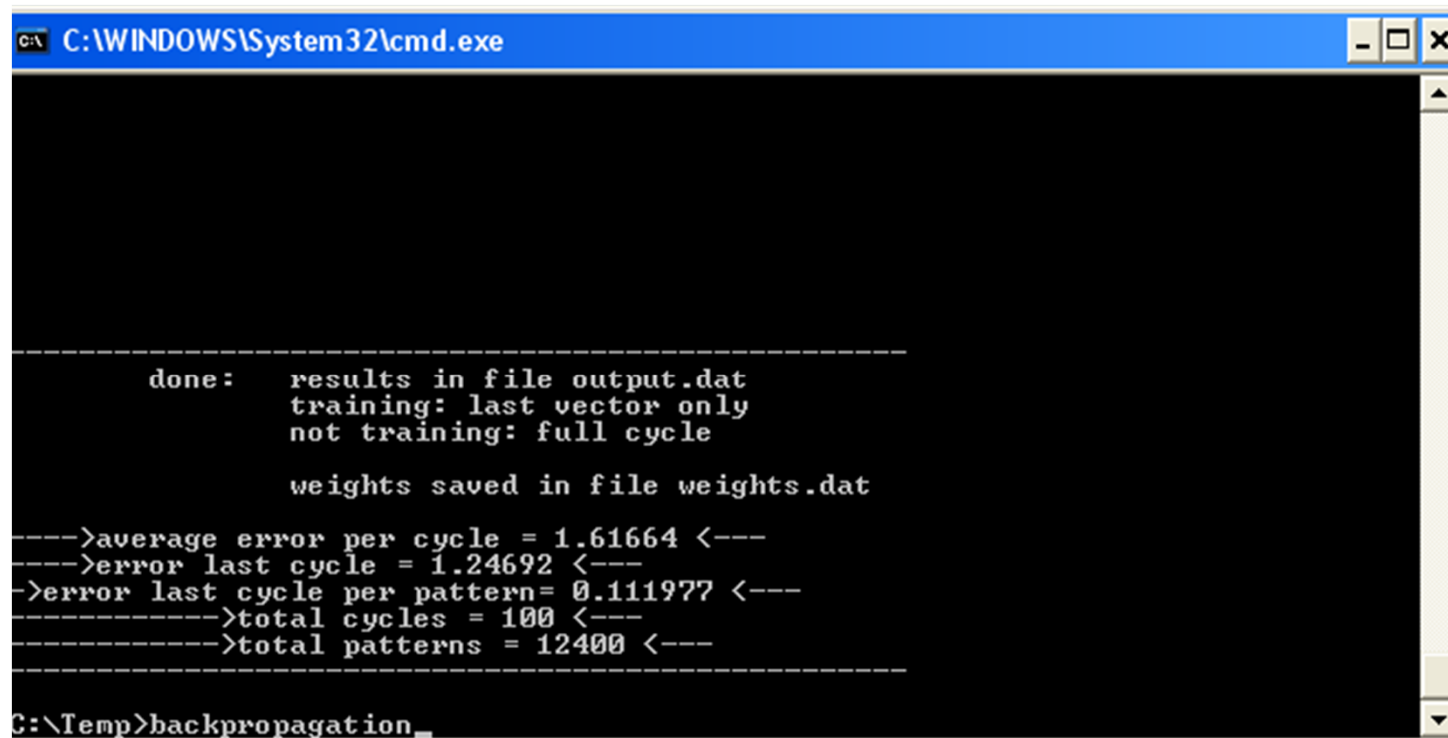
        weights saved in file weights.dat

----->average error per cycle = 1.61664 <---
----->error last cycle = 1.24692 <---
----->error last cycle per pattern= 0.111977 <---
----->total cycles = 100 <---
----->total patterns = 12400 <---
-----

C:\Temp>
```

Using "backprop.cpp" (TESTING)

- On the DOS prompt, type "backpropagation" to run the application again



```
C:\WINDOWS\System32\cmd.exe

-----
done:   results in file output.dat
        training: last vector only
        not training: full cycle

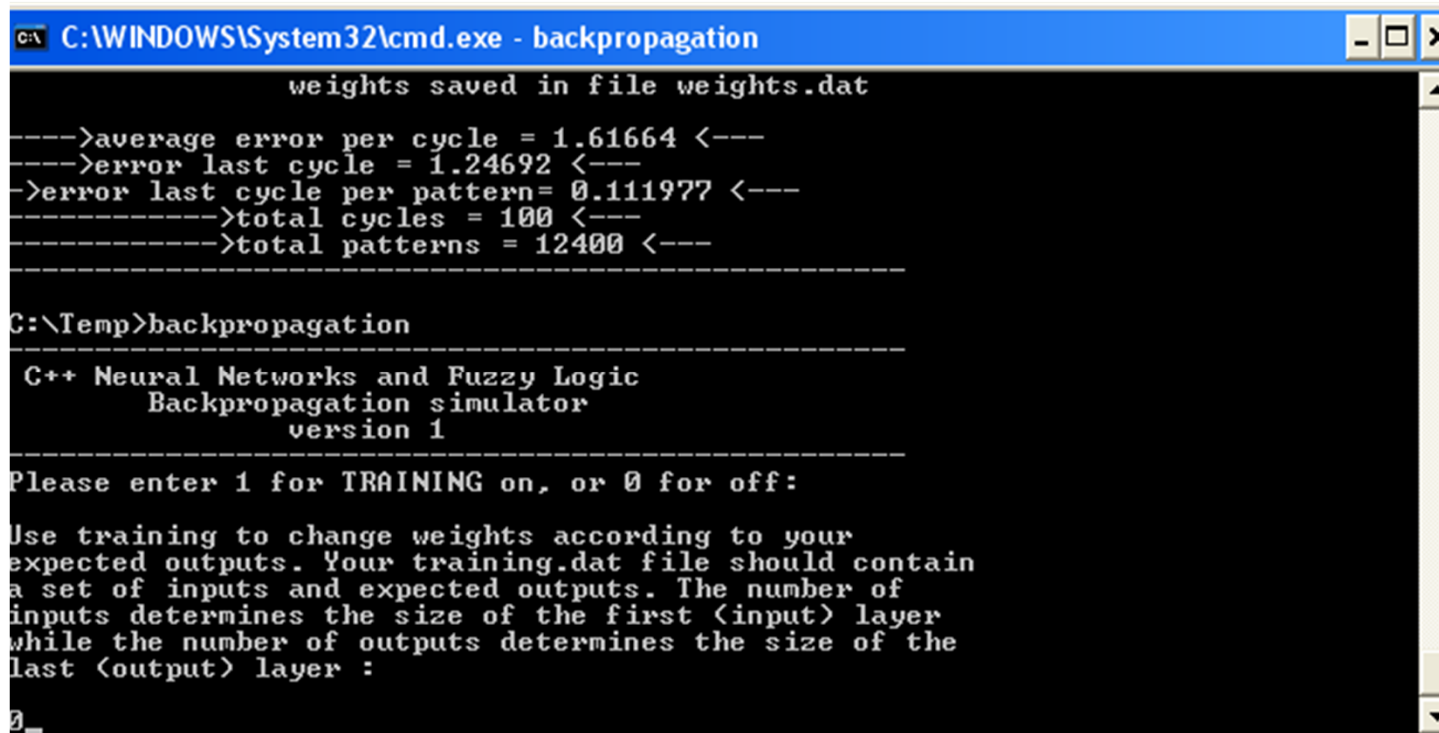
        weights saved in file weights.dat

----->average error per cycle = 1.61664 <---
----->error last cycle = 1.24692 <---
----->error last cycle per pattern= 0.111977 <---
----->total cycles = 100 <---
----->total patterns = 12400 <---
-----

C:\Temp>backpropagation_
```

Using "backprop.cpp" (TESTING)

- Enter "0" for TESTING

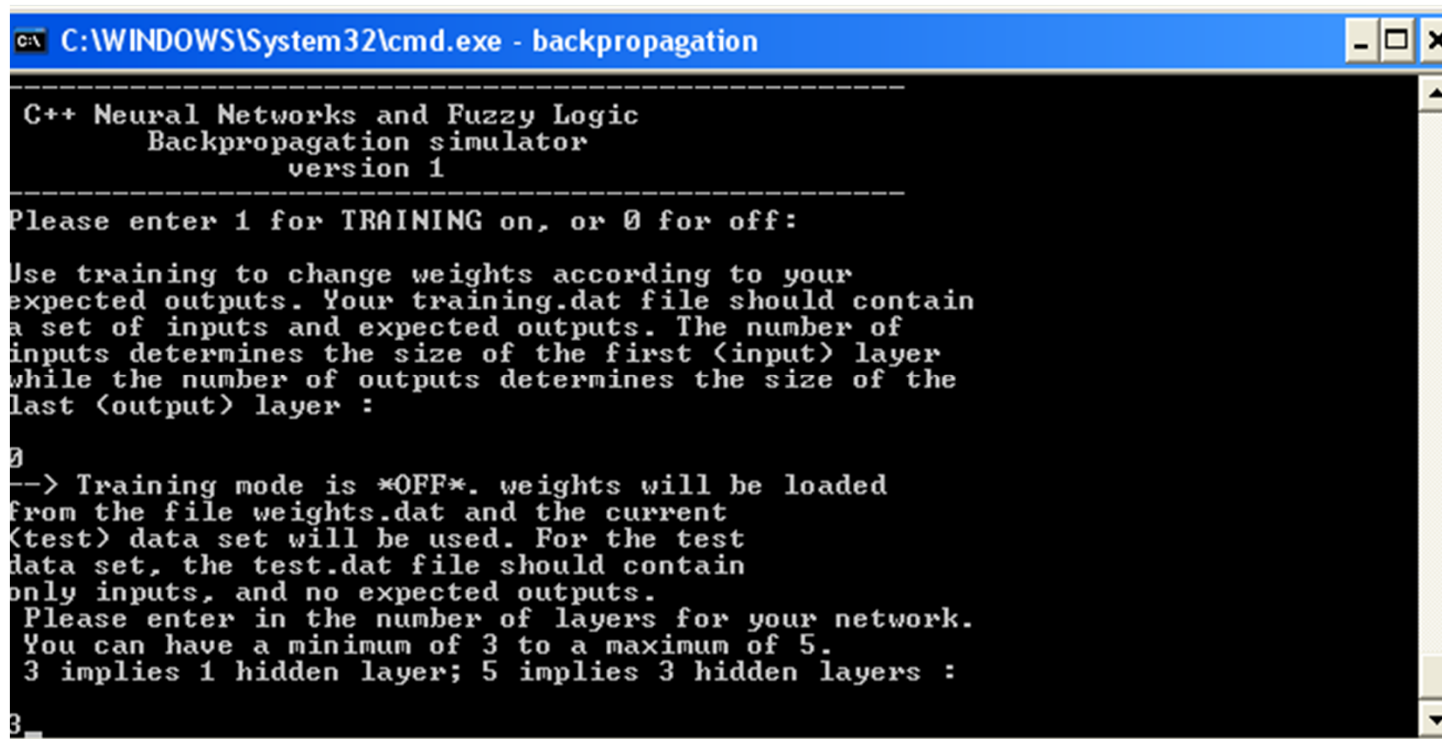


```
C:\WINDOWS\System32\cmd.exe - backpropagation
weights saved in file weights.dat
---->average error per cycle = 1.61664 <---
---->error last cycle = 1.24692 <---
->error last cycle per pattern= 0.111977 <---
----->total cycles = 100 <---
----->total patterns = 12400 <---
-----
C:\Temp>backpropagation
-----
C++ Neural Networks and Fuzzy Logic
Backpropagation simulator
version 1
-----
Please enter 1 for TRAINING on, or 0 for off:
Use training to change weights according to your
expected outputs. Your training.dat file should contain
a set of inputs and expected outputs. The number of
inputs determines the size of the first <input> layer
while the number of outputs determines the size of the
last <output> layer :
0_
```

Using "backprop.cpp" (TESTING)

- Enter the number of layers for your network.

IMPORTANT: This must be the same used for training



```
C:\WINDOWS\System32\cmd.exe - backpropagation
-----
C++ Neural Networks and Fuzzy Logic
  Backpropagation simulator
    version 1
-----
Please enter 1 for TRAINING on, or 0 for off:

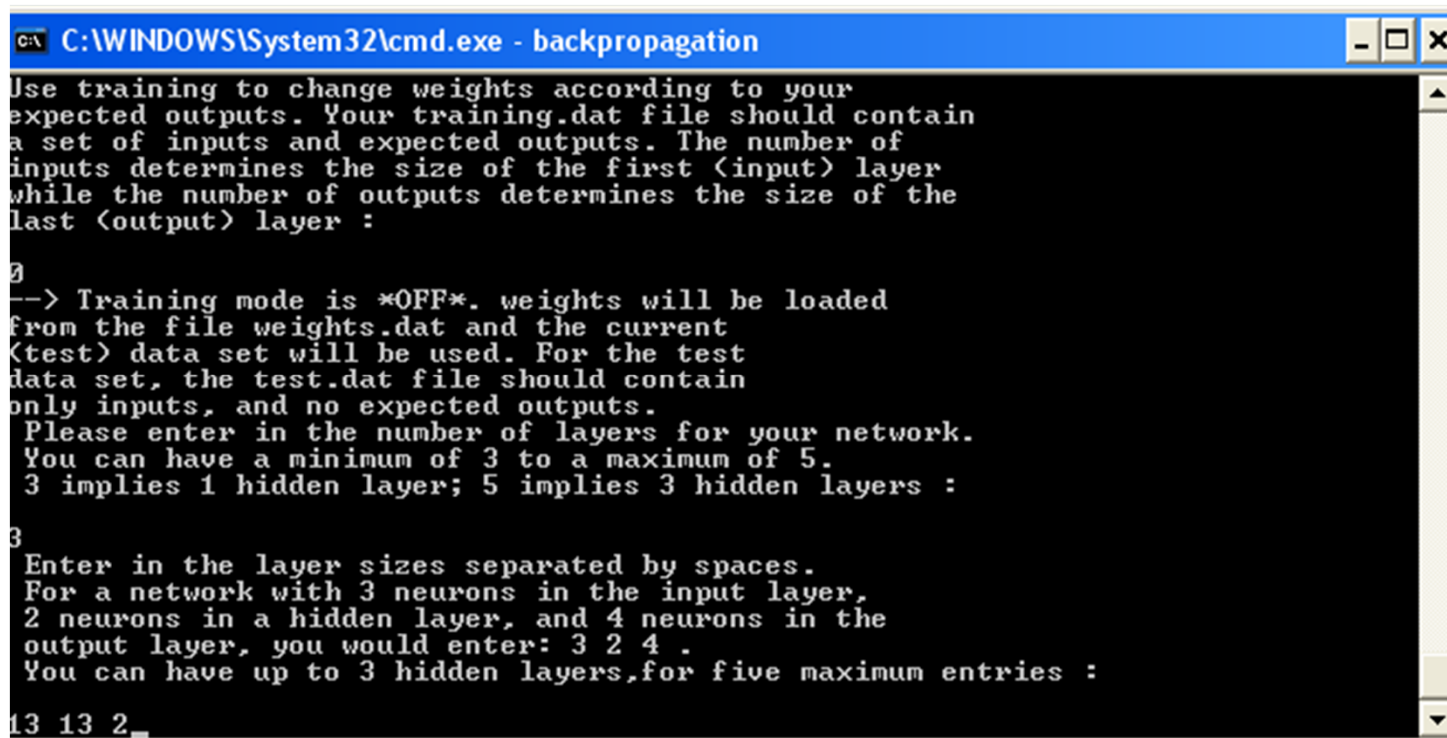
Use training to change weights according to your
expected outputs. Your training.dat file should contain
a set of inputs and expected outputs. The number of
inputs determines the size of the first (input) layer
while the number of outputs determines the size of the
last (output) layer :

0
--> Training mode is *OFF*. weights will be loaded
from the file weights.dat and the current
(test) data set will be used. For the test
data set, the test.dat file should contain
only inputs, and no expected outputs.
Please enter in the number of layers for your network.
You can have a minimum of 3 to a maximum of 5.
3 implies 1 hidden layer; 5 implies 3 hidden layers :
3_
```


Using "backprop.cpp" (TESTING)

- Enter the layer sizes separated by spaces.

IMPORTANT: This must be the same used for training



```
C:\WINDOWS\System32\cmd.exe - backpropagation
Use training to change weights according to your
expected outputs. Your training.dat file should contain
a set of inputs and expected outputs. The number of
inputs determines the size of the first (input) layer
while the number of outputs determines the size of the
last (output) layer :

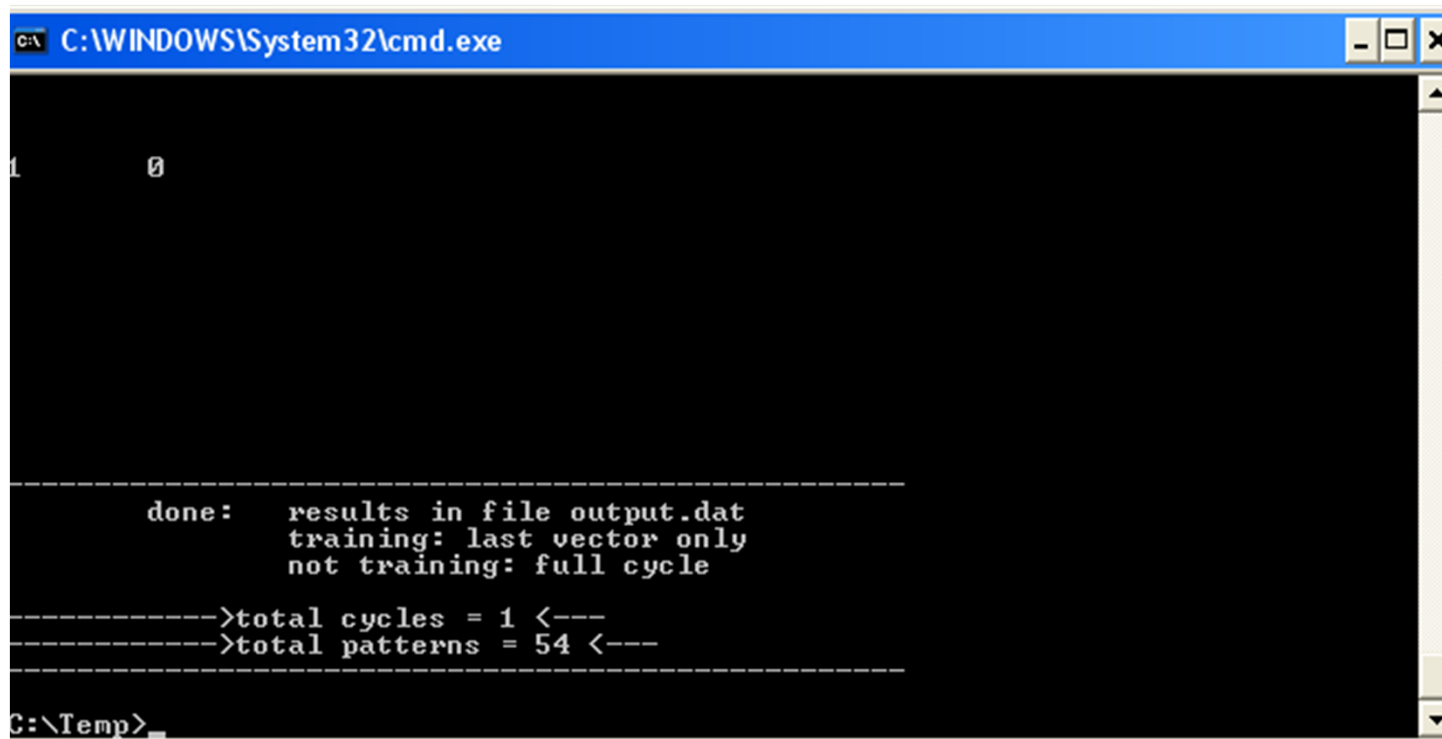
0
--> Training mode is *OFF*. weights will be loaded
from the file weights.dat and the current
(test) data set will be used. For the test
data set, the test.dat file should contain
only inputs, and no expected outputs.
Please enter in the number of layers for your network.
You can have a minimum of 3 to a maximum of 5.
3 implies 1 hidden layer; 5 implies 3 hidden layers :

3
Enter in the layer sizes separated by spaces.
For a network with 3 neurons in the input layer,
2 neurons in a hidden layer, and 4 neurons in the
output layer, you would enter: 3 2 4 .
You can have up to 3 hidden layers, for five maximum entries :

13 13 2_
```

Using “backprop.cpp” (TESTING)

- Run the neural network. Output results from the testing data set were saved on the “output.dat” file.



```
C:\WINDOWS\System32\cmd.exe

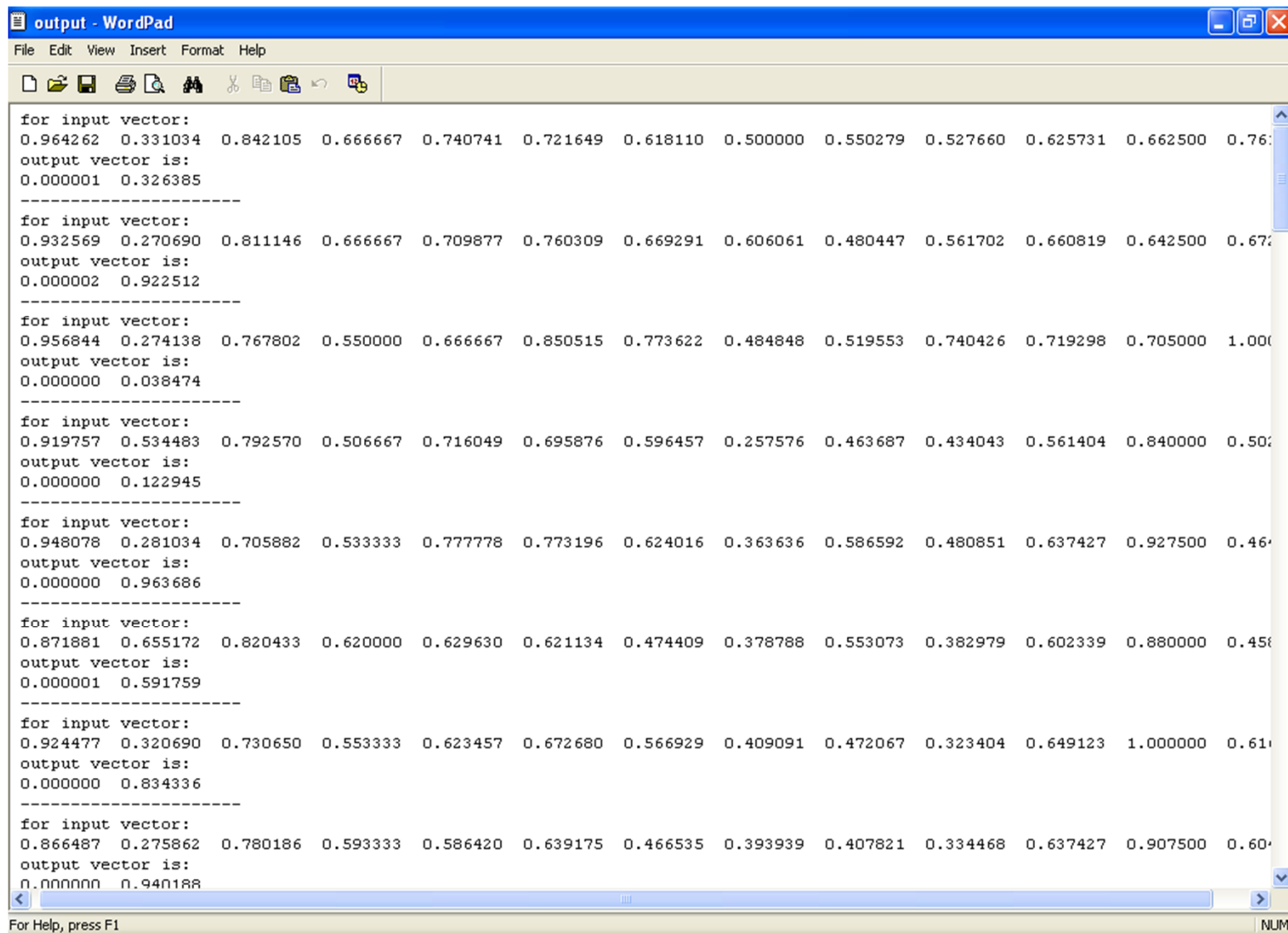
1      0

-----
done:   results in file output.dat
       training: last vector only
       not training: full cycle

----->total cycles = 1 <---
----->total patterns = 54 <---
-----

C:\Temp>
```

Using "backprop.cpp" (OUTPUT)




```
output - WordPad
File Edit View Insert Format Help
for input vector:
0.964262 0.331034 0.842105 0.666667 0.740741 0.721649 0.618110 0.500000 0.550279 0.527660 0.625731 0.662500 0.76
output vector is:
0.000001 0.326385
-----
for input vector:
0.932569 0.270690 0.811146 0.666667 0.709877 0.760309 0.669291 0.606061 0.480447 0.561702 0.660819 0.642500 0.67
output vector is:
0.000002 0.922512
-----
for input vector:
0.956844 0.274138 0.767802 0.550000 0.666667 0.850515 0.773622 0.484848 0.519553 0.740426 0.719298 0.705000 1.00
output vector is:
0.000000 0.038474
-----
for input vector:
0.919757 0.534483 0.792570 0.506667 0.716049 0.695876 0.596457 0.257576 0.463687 0.434043 0.561404 0.840000 0.50
output vector is:
0.000000 0.122945
-----
for input vector:
0.948078 0.281034 0.705882 0.533333 0.777778 0.773196 0.624016 0.363636 0.586592 0.480851 0.637427 0.927500 0.46
output vector is:
0.000000 0.963686
-----
for input vector:
0.871881 0.655172 0.820433 0.620000 0.629630 0.621134 0.474409 0.378788 0.553073 0.382979 0.602339 0.880000 0.45
output vector is:
0.000001 0.591759
-----
for input vector:
0.924477 0.320690 0.730650 0.553333 0.623457 0.672680 0.566929 0.409091 0.472067 0.323404 0.649123 1.000000 0.61
output vector is:
0.000000 0.834336
-----
for input vector:
0.866487 0.275862 0.780186 0.593333 0.586420 0.639175 0.466535 0.393939 0.407821 0.334468 0.637427 0.907500 0.60
output vector is:
0.000000 0.940188
For Help, press F1 NUM
```

SVM-Light Support Vector Machine - Microsoft Internet Explorer


File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites Media Print Mail Stop

Address <http://svmlight.joachims.org/> Go



SVM^{light}



Support Vector Machine

Author: [Thorsten Joachims](mailto:thorsten@joachims.org) <thorsten@joachims.org>
[Cornell University](#)
[Department of Computer Science](#)

Developed at:
[University of Dortmund](#), [Informatik, AI-Unit](#)
[Collaborative Research Center on 'Complexity Reduction in Multivariate Data' \(SFB475\)](#)

Version: 6.01
Date: 02.09.2004

Overview

SVM^{light} is an implementation of Support Vector Machines (SVMs) in C. The main features of the program are the following:

- fast optimization algorithm
 - working set selection based on steepest feasible descent
 - "shrinking" heuristic
 - caching of kernel evaluations
 - use of folding in the linear case
- solves classification and regression problems. For multivariate and structured outputs use [SVM^{struct}](#).
- solves ranking problems (e. g. learning retrieval functions in [STRIVER](#) search engine).
- computes XiAlpha-estimates of the error rate, the precision, and the recall
- efficiently computes Leave-One-Out estimates of the error rate, the precision, and the recall
- includes algorithm for approximately training large transductive SVMs (TSVMs) (see also [Spectral Graph Transducer](#))

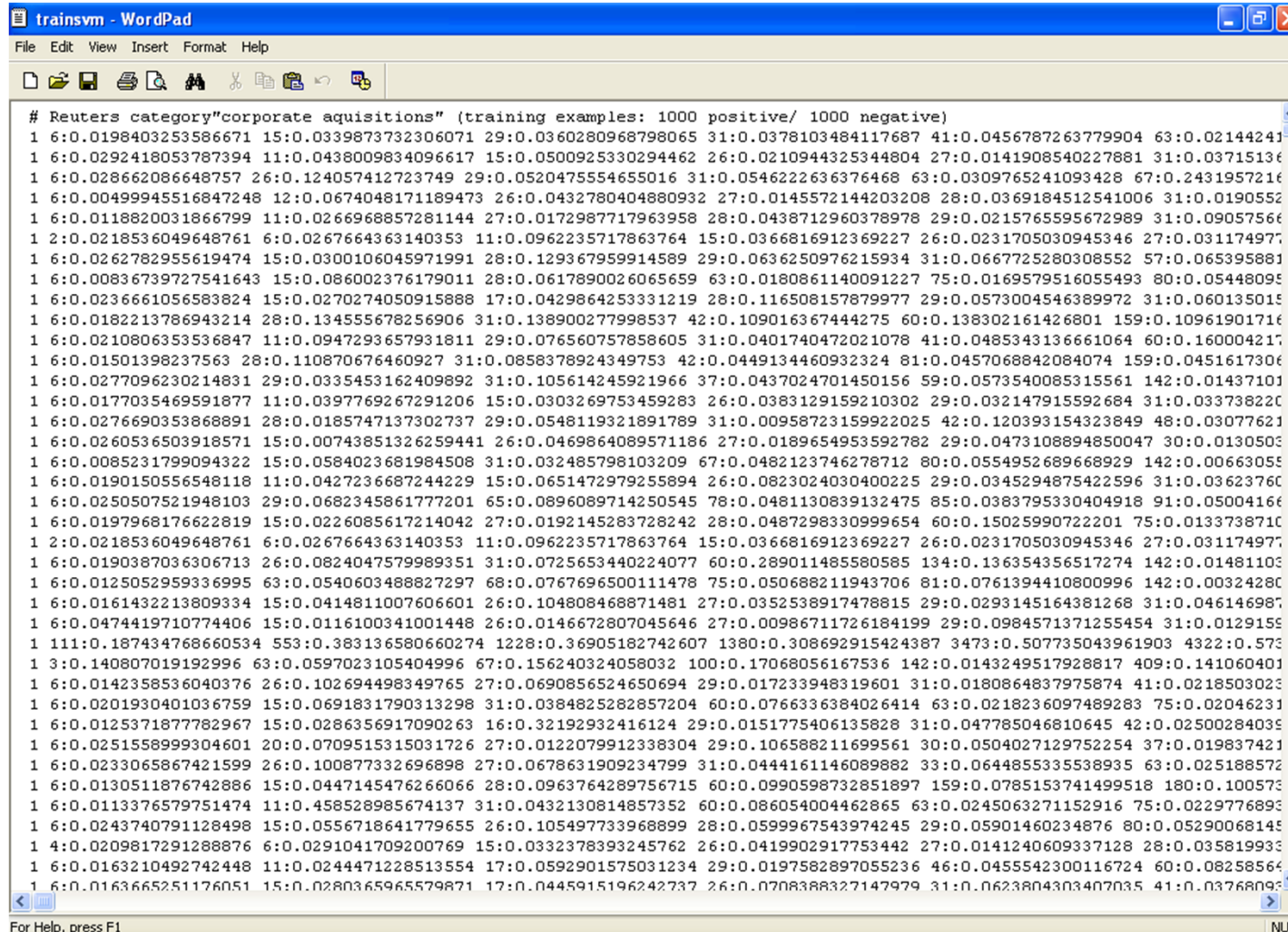
[SVM^{light} \(http://svmlight.joachims.org/\)](http://svmlight.joachims.org/)

Using "SVM^{light}"

- Download the executable file as well as the example training and testing data sets
 - Reference : T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

Using "SVM^{light}"

- Training file ("trainsvm.dat"):



```
# Reuters category "corporate aquisitions" (training examples: 1000 positive/ 1000 negative)
1 6:0.0198403253586671 15:0.0339873732306071 29:0.0360280968798065 31:0.0378103484117687 41:0.0456787263779904 63:0.02144241
1 6:0.0292418053787394 11:0.0438009834096617 15:0.0500925330294462 26:0.0210944325344804 27:0.0141908540227881 31:0.03715136
1 6:0.028662086648757 26:0.124057412723749 29:0.0520475554655016 31:0.0546222636376468 63:0.0309765241093428 67:0.2431957216
1 6:0.00499945516847248 12:0.0674048171189473 26:0.0432780404880932 27:0.0145572144203208 28:0.0369184512541006 31:0.0190552
1 6:0.0118820031866799 11:0.0266968857281144 27:0.0172987717963958 28:0.0438712960378978 29:0.0215765595672989 31:0.09057566
1 2:0.0218536049648761 6:0.0267664363140353 11:0.0962235717863764 15:0.0366816912369227 26:0.0231705030945346 27:0.031174977
1 6:0.0262782955619474 15:0.0300106045971991 28:0.129367959914589 29:0.0636250976215934 31:0.0667725280308552 57:0.0653958881
1 6:0.00836739727541643 15:0.086002376179011 28:0.0617890026065659 63:0.0180861140091227 75:0.0169579516055493 80:0.05448095
1 6:0.0236661056583824 15:0.0270274050915888 17:0.0429864253331219 28:0.116508157879977 29:0.0573004546389972 31:0.060135015
1 6:0.0182213786943214 28:0.134555678256906 31:0.138900277998537 42:0.109016367444275 60:0.138302161426801 159:0.10961901716
1 6:0.0210806353536847 11:0.0947293657931811 29:0.076560757858605 31:0.0401740472021078 41:0.0485343136661064 60:0.160004217
1 6:0.01501398237563 28:0.110870676460927 31:0.0858378924349753 42:0.0449134460932324 81:0.0457068842084074 159:0.0451617306
1 6:0.0277096230214831 29:0.0335453162409892 31:0.105614245921966 37:0.0437024701450156 59:0.0573540085315561 142:0.01437101
1 6:0.0177035469591877 11:0.0397769267291206 15:0.0303269753459283 26:0.0383129159210302 29:0.032147915592684 31:0.033738220
1 6:0.0276690353868891 28:0.0185747137302737 29:0.0548119321891789 31:0.00958723159922025 42:0.120393154323849 48:0.03077621
1 6:0.0260536503918571 15:0.00743851326259441 26:0.0469864089571186 27:0.0189654953592782 29:0.0473108894850047 30:0.0130503
1 6:0.0085231799094322 15:0.0584023681984508 31:0.032485798103209 67:0.0482123746278712 80:0.0554952689668929 142:0.00663055
1 6:0.0190150556548118 11:0.0427236687244229 15:0.0651472979255894 26:0.0823024030400225 29:0.0345294875422596 31:0.03623760
1 6:0.0250507521948103 29:0.0682345861777201 65:0.0896089714250545 78:0.0481130839132475 85:0.0383795330404918 91:0.05004166
1 6:0.0197968176622819 15:0.0226085617214042 27:0.0192145283728242 28:0.0487298330999654 60:0.15025990722201 75:0.0133738710
1 2:0.0218536049648761 6:0.0267664363140353 11:0.0962235717863764 15:0.0366816912369227 26:0.0231705030945346 27:0.031174977
1 6:0.0190387036306713 26:0.0824047579989351 31:0.0725653440224077 60:0.289011485580585 134:0.136354356517274 142:0.01481103
1 6:0.0125052959336995 63:0.0540603488827297 68:0.0767696500111478 75:0.050688211943706 81:0.0761394410800996 142:0.00324280
1 6:0.0161432213809334 15:0.0414811007606601 26:0.104808468871481 27:0.0352538917478815 29:0.0293145164381268 31:0.046146987
1 6:0.0474419710774406 15:0.0116100341001448 26:0.0146672807045646 27:0.00986711726184199 29:0.0984571371255454 31:0.0129155
1 11:0.187434768660534 553:0.383136580660274 1228:0.36905182742607 1380:0.308692915424387 3473:0.507735043961903 4322:0.573
1 3:0.140807019192996 63:0.0597023105404996 67:0.156240324058032 100:0.17068056167536 142:0.0143249517928817 409:0.141060401
1 6:0.0142358536040376 26:0.102694498349765 27:0.0690856524650694 29:0.017233948319601 31:0.0180864837975874 41:0.0218503023
1 6:0.0201930401036759 15:0.0691831790313298 31:0.0384825282857204 60:0.0766336384026414 63:0.0218236097489283 75:0.02046231
1 6:0.0125371877782967 15:0.0286356917090263 16:0.32192932416124 29:0.0151775406135828 31:0.047785046810645 42:0.02500284035
1 6:0.0251558999304601 20:0.0709515315031726 27:0.0122079912338304 29:0.106588211699561 30:0.0504027129752254 37:0.019837421
1 6:0.0233065867421599 26:0.100877332696898 27:0.0678631909234799 31:0.0444161146089882 33:0.0644855335538935 63:0.025188572
1 6:0.0130511876742886 15:0.0447145476266066 28:0.0963764289756715 60:0.0990598732851897 159:0.0785153741499518 180:0.100573
1 6:0.0113376579751474 11:0.458528985674137 31:0.0432130814857352 60:0.086054004462865 63:0.0245063271152916 75:0.0229776893
1 6:0.0243740791128498 15:0.0556718641779655 26:0.105497733968899 28:0.0599967543974245 29:0.05901460234876 80:0.05290068145
1 4:0.0209817291288876 6:0.0291041709200769 15:0.0332378393245762 26:0.0419902917753442 27:0.0141240609337128 28:0.035819933
1 6:0.0163210492742448 11:0.0244471228513554 17:0.0592901575031234 29:0.0197582897055236 46:0.0455542300116724 60:0.08258564
1 6:0.0163665251176051 15:0.0280365965579871 17:0.0445915196242737 26:0.0708388327147979 31:0.0623804303407035 41:0.03768093
```

Using "SVM^{light}"

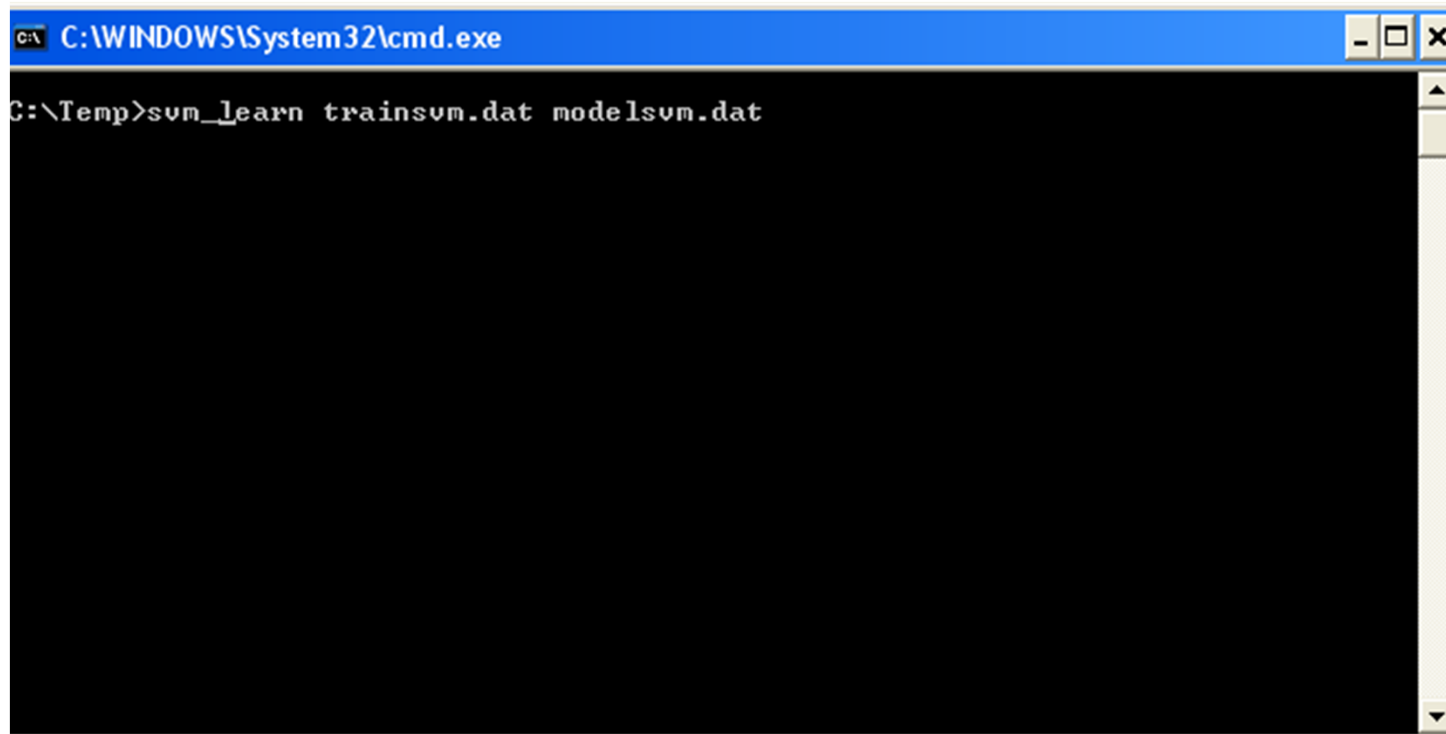
- Testing file ("testsvm.dat"):

```
testsvm - WordPad
File Edit View Insert Format Help
# Reuters category "corporate acquisitions" (test examples: 300 positive/ 300 negative)
+1 6:0.0342598670723747 26:0.148286149621374 27:0.0570037235976456 31:0.0373086482671729 33:0.0270832794680822 63:0.03173684
+1 179:0.183380155787579 199:0.0.157247710191107 408:0.198088894016017 587:0.42606443224553 1990:0.447885839011823 2134:0.458
+1 6:0.00482709726364712 17:0.0526067936384763 26:0.10446503857735 27:0.0281106990768854 28:0.0356456751828835 63:0.03130128
+1 6:0.00797462873810959 26:0.0345163916642208 27:0.0696606189381528 33:0.0441290005061936 67:0.0451094300865861 142:0.01654
+1 6:0.020398477445271 11:0.0458319874879689 15:0.0349435129583541 26:0.0441451169688967 28:0.0753162264530517 29:0.03704164
+1 6:0.0331457691319788 31:0.126333924126773 60:0.251579838716322 68:0.203480917932947 204:0.198980295178862 252:0.239210120
+1 6:0.0218491019698381 18:0.0584788160942556 26:0.0472844683977855 27:0.0159048362045602 28:0.0403361455817294 31:0.0208192
+1 6:0.00348466932721003 26:0.0452478285578935 27:0.0710256494147685 29:0.0506225585902376 33:0.0578490784540854 64:0.090670
+1 6:0.00785445606167515 15:0.0538201516085403 28:0.0580011908235646 29:0.0570517063228698 40:0.0490761508769594 63:0.033954
+1 6:0.00783355108261519 15:0.0536769069161587 28:0.057846818112062 29:0.0568998607059805 40:0.0489455325504489 63:0.0338644
+1 31:0.121578360639455 174:0.23382059176088 232:0.283129708150668 389:0.271848845786714 506:0.17845240482837 719:0.34333381
+1 6:0.0261488826243853 15:0.044794216690105 26:0.16976935928035 29:0.047483821936342 63:0.0282603811429679 67:0.07395712272
+1 26:0.15620989586763 28:0.266510561058506 63:0.0780096811503992 142:0.00935880799034978 158:0.259041653526988 204:0.216656
+1 26:0.176202309110135 27:0.118536549536136 60:0.308989992539676 80:0.265064145238512 142:0.0527829421196014 153:0.35308467
+1 6:0.0282619337466596 15:0.0322759791183498 26:0.0407751468824368 29:0.0342139445339461 31:0.0359064528910632 67:0.1065775
+1 6:0.023702409591198 12:0.0798915346740287 27:0.0345078660644096 29:0.0860825307648993 31:0.0451704469881664 37:0.05607366
+1 6:0.0263446335928295 15:0.120345458501861 26:0.152035782680804 27:0.0511395031541379 28:0.129694541815601 29:0.0318928577
+1 111:0.215994569070182 186:0.195992437000748 229:0.180872475933477 426:0.310925435082124 586:0.253890450118932 800:0.30457
+1 6:0.0414181490451389 15:0.0709511595535961 26:0.0896345837153026 27:0.0602998583070711 28:0.07646302772154 29:0.075211321
+1 6:0.0300323798317241 15:0.0205787291063404 26:0.129988419388934 27:0.0174894174677238 31:0.0228934702338641 63:0.01298296
+1 6:0.0142894732787166 26:0.0618487797440109 28:0.0527602713605547 29:0.0259482905819441 31:0.0272319104410711 63:0.0154433
+1 6:0.0149480029785652 29:0.0271441163254978 31:0.0569737834901793 63:0.0161550406404948 68:0.0458826789511475 75:0.0151473
+1 6:0.0176610820154908 15:0.0302542309798681 26:0.152884063717583 27:0.0257124175665849 31:0.0673145880687707 60:0.06702472
+1 6:0.0576644726785321 26:0.124793867491993 28:0.212911502547412 31:0.219786093578729 63:0.062320826465597 67:0.16309295292
+1 6:0.0309109297810553 15:0.0423614548000294 29:0.134714964920907 31:0.0471263652636768 41:0.0284667086458094 63:0.06681391
+1 6:0.0222985466329023 15:0.0327415005210145 26:0.0689387551446239 27:0.0278262942367657 28:0.0470467154727192 29:0.0809835
+1 186:0.143427256583034 252:0.182408394129025 527:0.241068649672439 586:0.185797019979149 965:0.252463123260795 1038:0.3355
+1 6:0.0149480029785652 29:0.0271441163254978 31:0.0569737834901793 63:0.0161550406404948 68:0.0458826789511475 75:0.0151473
+1 6:0.0146297124837151 26:0.0379928572932579 27:0.00851964281532518 29:0.042505810900465 37:0.013844026081929 41:0.01347285
+1 6:0.0117995828147134 26:0.0510718473903028 27:0.0114525184122277 28:0.0580893061722454 29:0.0571383792156861 31:0.0149912
+1 6:0.0248233972589892 12:0.0669359558643418 26:0.0644655040654039 27:0.101191691115903 28:0.109984950453772 29:0.036061495
+1 6:0.0233852875833601 28:0.0431721342934449 29:0.0212327013641281 31:0.0222830486711246 49:0.0528092931406353 60:0.0887483
+1 27:0.0824301952103377 65:0.202530970937398 111:0.177040474823198 142:0.0146820815679921 204:0.169946684905274 322:0.24534
+1 6:0.00932846504342859 15:0.0639203273874091 28:0.0688860026494896 40:0.0582860422576615 63:0.0403269204872715 74:0.229415
+1 6:0.0193999731994898 28:0.0716295016917865 29:0.0352284602828621 41:0.0446468959379451 63:0.0628995102379255 75:0.0589760
+1 6:0.0188827964772949 31:0.0359855547168811 81:0.0574846679931861 142:0.00489658413875352 199:0.0441326542263463 205:0.074
+1 6:0.0217372792609427 15:0.0248246272815232 29:0.0263151868382218 63:0.0313233930871678 75:0.0293695253621568 80:0.0943557
+1 6:0.0378781390022991 15:0.0432579750005485 26:0.136622523364546 27:0.0183820093962911 28:0.0466184874560938 29:0.0687830
```

Using "SVM^{light}" (TRAINING)

svm_learn is called with the following parameters:

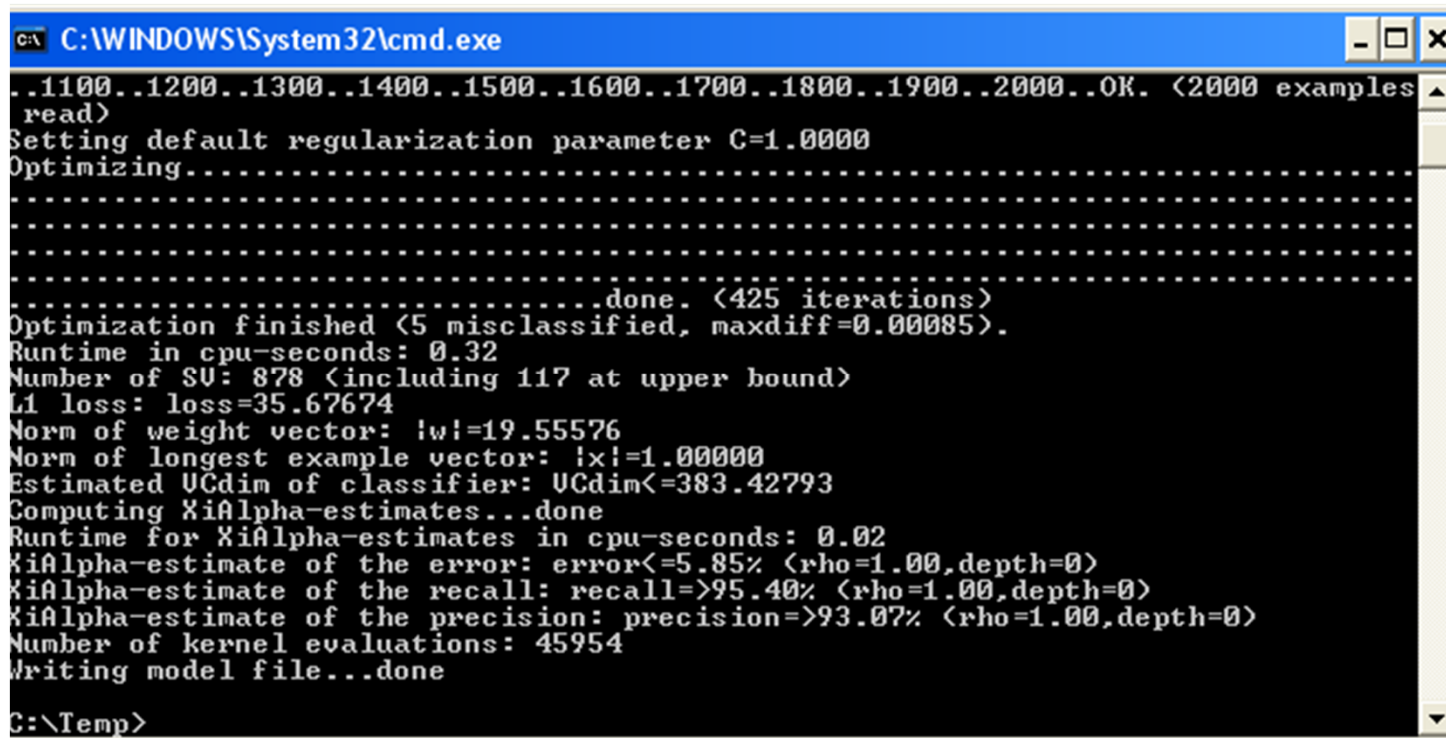
```
svm_learn [options] trainingexamples_file model_file
```



```
C:\WINDOWS\System32\cmd.exe
C:\Temp>svm_learn trainsvm.dat modelsvm.dat
```


Using "SVM^{light}" (TRAINING)

Run svm_learn. The model is saved at the "model_file".
Later, this file will be used for testing

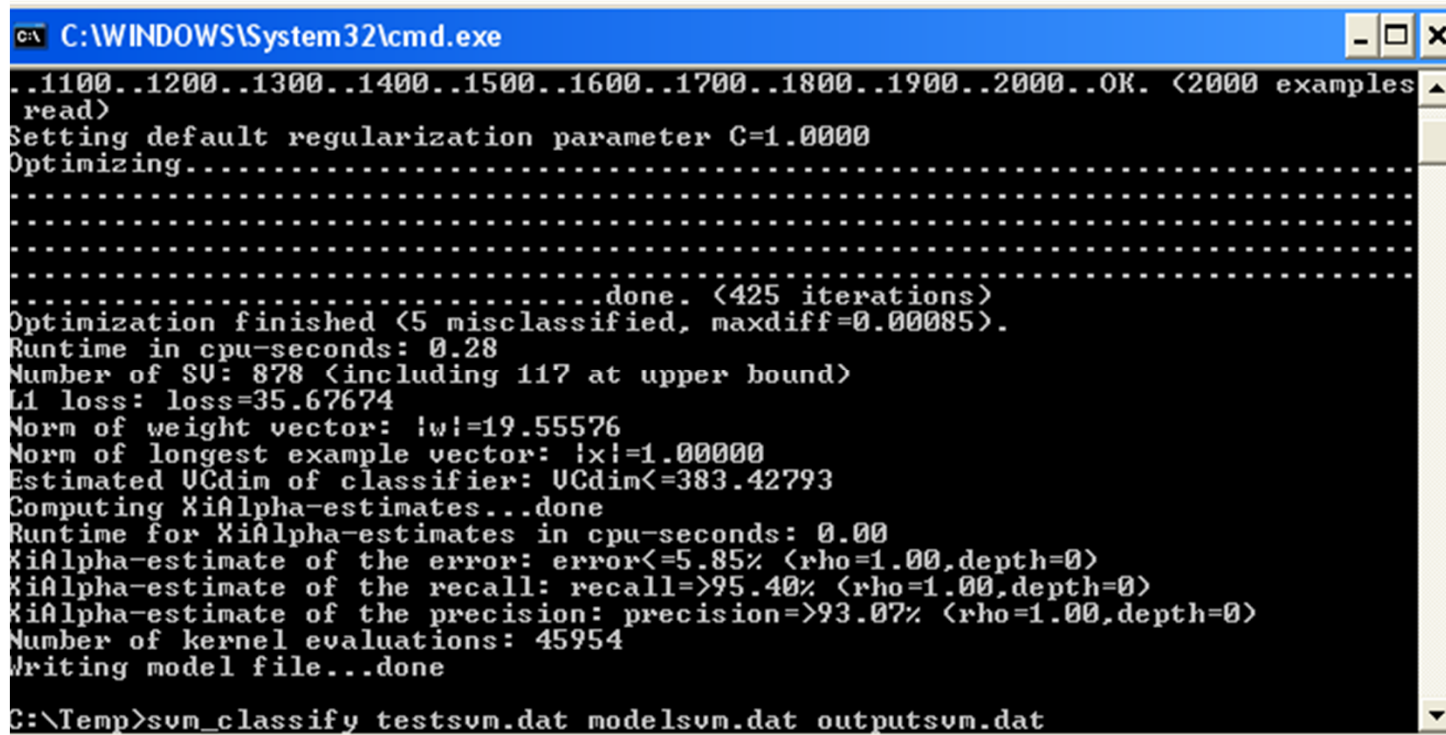


```
C:\WINDOWS\System32\cmd.exe
..1100..1200..1300..1400..1500..1600..1700..1800..1900..2000..OK. (2000 examples
read)
Setting default regularization parameter C=1.0000
Optimizing.....
.....
.....done. (425 iterations)
Optimization finished (5 misclassified, maxdiff=0.00085).
Runtime in cpu-seconds: 0.32
Number of SU: 878 (including 117 at upper bound)
L1 loss: loss=35.67674
Norm of weight vector: |w|=19.55576
Norm of longest example vector: |x|=1.00000
Estimated UCdim of classifier: UCdim<=383.42793
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.02
XiAlpha-estimate of the error: error<=5.85% (rho=1.00,depth=0)
XiAlpha-estimate of the recall: recall=>95.40% (rho=1.00,depth=0)
XiAlpha-estimate of the precision: precision=>93.07% (rho=1.00,depth=0)
Number of kernel evaluations: 45954
Writing model file...done
C:\Temp>
```

Using "SVM^{light}" (TESTING)

svm_classify is called with the following parameters:

svm_classify [options] testingexamples_file model_file output_file



```
C:\WINDOWS\System32\cmd.exe
..1100..1200..1300..1400..1500..1600..1700..1800..1900..2000..OK. <2000 examples
read>
Setting default regularization parameter C=1.0000
Optimizing.....
.....
.....done. <425 iterations>
Optimization finished <5 misclassified, maxdiff=0.00085>.
Runtime in cpu-seconds: 0.28
Number of SU: 878 <including 117 at upper bound>
L1 loss: loss=35.67674
Norm of weight vector: |w|=19.55576
Norm of longest example vector: |x|=1.00000
Estimated UCdim of classifier: UCdim<=383.42793
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.00
XiAlpha-estimate of the error: error<=5.85% <rho=1.00,depth=0>
XiAlpha-estimate of the recall: recall=>95.40% <rho=1.00,depth=0>
XiAlpha-estimate of the precision: precision=>93.07% <rho=1.00,depth=0>
Number of kernel evaluations: 45954
Writing model file...done
C:\Temp>svm_classify testsvm.dat modelsvm.dat outputsvm.dat
```

Using "SVM^{light}" (TESTING)

Run svm_classify. Results will be saved at the "output_file"

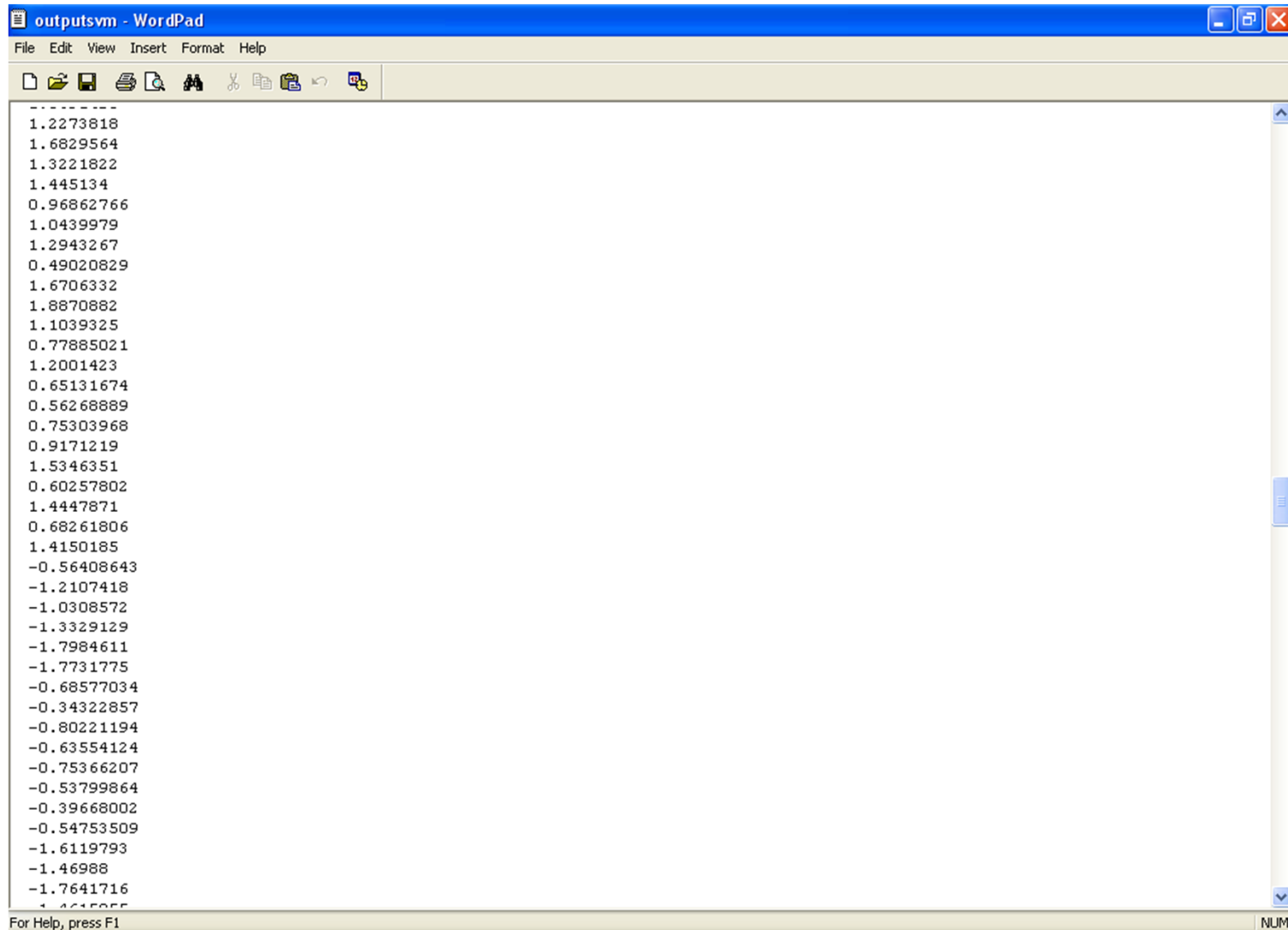
```
C:\WINDOWS\System32\cmd.exe
.....done. <425 iterations>
Optimization finished <5 misclassified, maxdiff=0.00085>.
Runtime in cpu-seconds: 0.32
Number of SV: 878 <including 117 at upper bound>
L1 loss: loss=35.67674
Norm of weight vector: |w|=19.55576
Norm of longest example vector: |x|=1.00000
Estimated UCdim of classifier: UCdim<=383.42793
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.02
XiAlpha-estimate of the error: error<=5.85% <rho=1.00,depth=0>
XiAlpha-estimate of the recall: recall=>95.40% <rho=1.00,depth=0>
XiAlpha-estimate of the precision: precision=>93.07% <rho=1.00,depth=0>
Number of kernel evaluations: 45954
Writing model file...done

C:\Temp>svm_classify testsvm.dat modelsvm.dat outputsvm.dat
Reading model...OK. <878 support vectors read>
Classifying test examples..100..200..300..400..500..600..done
Runtime <without IO> in cpu-seconds: 0.00
Accuracy on test set: 97.67% <586 correct, 14 incorrect, 600 total>
Precision/recall on test set: 96.43%/99.00%

C:\Temp>_
```

Using "SVM^{light}" (RESULTS)

- Output file ("outputsvm.dat"):



The screenshot shows a Windows WordPad window titled "outputsvm - WordPad". The window contains a list of 40 numerical values, each on a new line. The values are: 1.2273818, 1.6829564, 1.3221822, 1.445134, 0.96862766, 1.0439979, 1.2943267, 0.49020829, 1.6706332, 1.8870882, 1.1039325, 0.77885021, 1.2001423, 0.65131674, 0.56268889, 0.75303968, 0.9171219, 1.5346351, 0.60257802, 1.4447871, 0.68261806, 1.4150185, -0.56408643, -1.2107418, -1.0308572, -1.3329129, -1.7984611, -1.7731775, -0.68577034, -0.34322857, -0.80221194, -0.63554124, -0.75366207, -0.53799864, -0.39668002, -0.54753509, -1.6119793, -1.46988, -1.7641716, and -1.6150555. The window has a standard menu bar (File, Edit, View, Insert, Format, Help) and a toolbar with icons for file operations. The status bar at the bottom left says "For Help, press F1" and the bottom right shows "NUM".

```
-----  
1.2273818  
1.6829564  
1.3221822  
1.445134  
0.96862766  
1.0439979  
1.2943267  
0.49020829  
1.6706332  
1.8870882  
1.1039325  
0.77885021  
1.2001423  
0.65131674  
0.56268889  
0.75303968  
0.9171219  
1.5346351  
0.60257802  
1.4447871  
0.68261806  
1.4150185  
-0.56408643  
-1.2107418  
-1.0308572  
-1.3329129  
-1.7984611  
-1.7731775  
-0.68577034  
-0.34322857  
-0.80221194  
-0.63554124  
-0.75366207  
-0.53799864  
-0.39668002  
-0.54753509  
-1.6119793  
-1.46988  
-1.7641716  
-1.6150555
```