Neural Networks - 1
Robert Stengel
Robotics and Intelligent Systems, MAE 345, Princeton University, 2013

Learning Objectives

• Natural and artificial neurons
• Natural and computational neural networks
  – Linear network
  – Perceptron
  – Sigmoid network
• Applications of neural networks
• Supervised training
  – Left pseudoinverse
  – Steepest descent
  – Back-propagation
  – Exact algebraic fit

Copyright 2013 by Robert Stengel. All rights reserved. For educational use only.
http://www.princeton.edu/~stengel/MAE345.html

Applications of Computational Neural Networks

• Classification of data sets
• Nonlinear function approximation
  • Efficient data storage and retrieval
  • System identification
• Nonlinear and adaptive control systems
Neurons

- Biological cells with significant electrochemical activity
- ~10-100 billion neurons in the brain
- Inputs from thousands of other neurons
- Output is scalar, but may have thousands of branches

- Afferent (unipolar) neurons send signals from organs and the periphery to the central nervous system
- Efferent (multipolar) neurons issue commands from the CNS to effector (e.g., muscle) cells
- Interneurons (multipolar) send signals between neurons in the central nervous system
- Signals are ionic, i.e., chemical (neurotransmitter atoms and molecules) and electrical (potential)

Activation Input to Soma Causes Change in Output Potential

- Stimulus from
  - Other neurons
  - Muscle cells
  - Pacemakers (c.g. cardiac sino-atrial node)
- When membrane potential of neuronal cell exceeds a threshold
  - Cell is polarized
  - Action potential pulse is transmitted from the cell
  - Activity measured by amplitude and firing frequency of pulses
- Cell depolarizes and potential returns to rest
Neural Action Potential

- **Maximum Firing Rate**: 500/sec
- **Refractory Period**: Minimum time increment between action potential firing ~ 1-2 msec

Some Recorded Action Potential Pulse Trains
Impulse, Pulse-Train, and Step Response of a LTI 2nd-Order Neural Model

In the limit, neglecting absolute refractory period

Multipolar Neuron
Mathematical Model of Neuron Components

Synapse effects represented by weights (gains or multipliers)
Neuron firing frequency is modeled by linear gain or nonlinear element

The Neuron Function

- Vector input, \( u \), to a single neuron
  - Sensory input or output from upstream neurons
  - Linear operation produces scalar, \( r \)
  - Add bias, \( b \), for zero adjustment
- Scalar output, \( u \), of a single neuron (or node)
  - Scalar linear or nonlinear operation, \( s(r) \)

\[
\begin{align*}
  r &= w^T u + b \\
  u &= s(r)
\end{align*}
\]
Layout of a Neural Network

Layered, parallel structure for computation

Input-Output Characteristics of a Neural Network Layer

- **Single layer**
  - Number of inputs = $n$
    - $\dim(u) = (n \times 1)$
  - Number of nodes = $m$
    - $\dim(r) = \dim(b) = \dim(s) = (m \times 1)$

$$r = Wu + b$$

$$u = s(r)$$

$W = \begin{bmatrix} w_1^T \\
       w_2^T \\
       \vdots \\
       w_n^T \end{bmatrix}$
Two-Layer Network

- **Two layers**
  - Number of nodes in each layer need not be the same
  - Node functions may be different, e.g.,
    - Sigmoid hidden layer
    - Linear output layer

\[
y = u_2 \\
= s_2(\mathbf{r}_2) = s_2(\mathbf{W}_2u_1 + \mathbf{b}_2) \\
= s_2\left[\mathbf{W}_2 s_1(\mathbf{r}_1) + \mathbf{b}_2\right] \\
= s_2\left[\mathbf{W}_2 s_1(\mathbf{W}_1u_0 + \mathbf{b}_1) + \mathbf{b}_2\right] \\
= s_2\left[\mathbf{W}_2 s_1(\mathbf{W}_1x + \mathbf{b}_1) + \mathbf{b}_2\right]
\]

---

**Is a Neural Network Serial or Parallel?**

**3rd-degree power series**

4 coefficients

Express as a neural network?

\[
y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\
= a_0' + a_1' r + a_2' r^2 + a_3' r^3 \\
= a_0' + a_1'(c_1 x + b_1) + a_2'(c_1 x + b_2)^2 + a_3'(c_1 x + b_3)^3 \\
= w_0 + w_1 s_1(u) + w_2 s_2(u) + w_3 s_3(u)
\]
Is a Neural Network Serial or Parallel?

Power series is serial, but it can be expressed as a parallel neural network (with dissimilar nodes)

MATLAB Neural Network Toolbox

• Implementation of many neural network architectures
• Common calling sequences
• Pre- and post-processing
• Command-line and GUI
MATLAB Training and Evaluation of “Backpropagation” Neural Networks

- Backpropagation (Ch. 5)
- Preprocessing to normalize data (5-62)
- Architecture (5-8)
- Simulation (5-14)
- Training algorithms (5-15, 5-52)

Linear Neural Network

- Outputs provide linear scaling of inputs
- Equivalent to matrix transformation of a vector, $y = Wx + b$
- Therefore, linear network is easy to train (left pseudoinverse)
- MATLAB symbology

\[ a = \text{purelin}(W*p+b) \]

Where...
- $R = \# \text{ Inputs}$
- $S = \# \text{ Neurons}$
Idealizations of Nonlinear Neuron Input-Output Characteristic

**Step function ("Perceptron")**

\[ u = s(r) = \begin{cases} 
1, & r > 0 \\
0, & r \leq 0 
\end{cases} \]

**Logistic sigmoid function**

\[ u = s(r) = \frac{1}{1 + e^{-r}} \]

**Sigmoid with two inputs, one output**

\[ u = s(r) = \frac{1}{1 + e^{-(w_1 r_1 + w_2 r_2 + b)}} \]

---

**Perceptron Neural Network**

Each node is a step function
Weighted sum of features is fed to each node
Each node produces a linear classification of the input space
Perceptron Neural Network

Weights adjust slopes
Biases adjust zero crossing points

Single-Layer, Single-Node Perceptron Discriminants

\[ u = s(w^T x + b) = \begin{cases} 1, & (w^T x + b) > 0 \\ 0, & (w^T x + b) \leq 0 \end{cases} \]

Two inputs, single step function Discriminant

\[ w_1 x_1 + w_2 x_2 + b = 0 \]
\[ \text{or } x_2 = \frac{-1}{w_2} (w_1 x_1 + b) \]

Three inputs, single step function Discriminant

\[ w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0 \]
\[ \text{or } x_3 = \frac{-1}{w_3} (w_1 x_1 + w_2 x_2 + b) \]
Single-Layer, Multi-Node Perceptron Discriminants

\[ u = s(Wx + b) \]

- Multiple inputs, nodes, and outputs
  - More inputs lead to more dimensions in discriminants
  - More outputs lead to more discriminants

Multi-Layer Perceptrons Can Classify With Boundaries or Clusters

Classification capability of multi-layer perceptrons
Classifications of classifications
Open or closed regions

<table>
<thead>
<tr>
<th>STRUCTURE</th>
<th>TYPES OF DECISION REGIONS</th>
<th>EXCLUSIVE OR PROBLEM</th>
<th>CLASSES WITH MESHED REGIONS</th>
<th>MOST GENERAL REGION SHAPES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLE LAYER</td>
<td>HALF PLANE BOUNDED BY HYPERPLANE</td>
<td>A B</td>
<td>B A</td>
<td>A B</td>
</tr>
<tr>
<td>TWO LAYER</td>
<td>CONVEX OPEN OR CLOSED REGIONS</td>
<td>B A</td>
<td>B A</td>
<td>B A</td>
</tr>
<tr>
<td>THREE LAYER</td>
<td>ARBITRARY (Complexity Limited By Number of Nodes)</td>
<td>B A</td>
<td>B A</td>
<td>B A</td>
</tr>
</tbody>
</table>
Sigmoid Activation Functions

- Alternative sigmoid functions
  - Logistic function: 0 to 1
  - Hyperbolic tangent: –1 to 1
  - Augmented ratio of squares: 0 to 1
- Smooth nonlinear functions

\[
u = s(r) = \frac{1}{1 + e^{-r}}
\]

\[
u = s(r) = \tanh r = \frac{1 - e^{-2r}}{1 + e^{-2r}}
\]

\[
u = s(r) = \frac{r^2}{1 + r^2}
\]

Sigmoid Neural Network

\[a = \text{logsig}(W*p + b)\]

Where...

- \(R\) = # Inputs
- \(S\) = # Neurons
Single Sigmoid Layer is Sufficient ...

- Sigmoid network with single hidden layer can approximate any continuous function
- Therefore, additional sigmoid layers are unnecessary
- Typical sigmoid network contains
  - Single sigmoid hidden layer (nonlinear fit)
  - Single linear output layer (scaling)

Typical Sigmoid Neural Network Output

Classification is not limited to linear discriminants

Sigmoid network can approximate a continuous nonlinear function to arbitrary accuracy with a single hidden layer
Thresholded Neural Network Output

Threshold gives “yes/no” output

Training Error and Cost for a Single Linear Neuron

- Training error: difference between network output and target output
- Quadratic error cost

\[ \varepsilon = \hat{y} - y_T \]
\[ J = \frac{1}{2} \varepsilon^2 = \frac{1}{2} (\hat{y} - y_T)^2 = \frac{1}{2} (\hat{y}^2 - 2\hat{y} y_T + y_T^2) \]

\[ \hat{y} = r = \hat{w}^T x + \hat{b} \]
**Linear Neuron Gradient**

\[ \hat{y} = r = w^T x + b \]
\[ \frac{d\hat{y}}{dr} = 1 \]
\[ \epsilon = \hat{y} - y_T \]
\[ J = \frac{1}{2} \epsilon^2 = \frac{1}{2}(\hat{y} - y_T)^2 = \frac{1}{2}(\hat{y}^2 - 2\hat{y}y_T + y_T^2) \]

- **Training (control) parameter, \( p \)**
  - Input weights, \( w \) \((n \times 1)\)
  - Bias, \( b \) \((1 \times 1)\)

- **Optimality condition**

\[ \frac{\partial J}{\partial p} = 0 \]

- **Gradient**

\[
\frac{\partial J}{\partial p} = (\hat{y} - y_T) \frac{\partial y}{\partial p} = (\hat{y} - y_T) \frac{\partial y}{\partial r} \frac{\partial r}{\partial p}
\]

where

\[
\frac{\partial r}{\partial p} = \begin{bmatrix}
\frac{\partial r}{\partial p_1} & \frac{\partial r}{\partial p_2} & \cdots & \frac{\partial r}{\partial p_{n+1}}
\end{bmatrix}
\]

\[
\frac{\partial (w^T x + b)}{\partial p} = \begin{bmatrix} x^T & 1 \end{bmatrix}
\]

**Steepest-Descent Learning for a Single Linear Neuron**

**Gradient**

\[
\frac{\partial J}{\partial p} = (\hat{y} - y_T) \begin{bmatrix} x^T & 1 \end{bmatrix} = \left[(w^T x + b) - y_T\right] \begin{bmatrix} x^T & 1 \end{bmatrix}
\]

**Steepest-descent algorithm**

\[ \eta = \text{learning rate} \]
\[ k = \text{iteration index(epoch)} \]

\[
p_{k+1} = p_k - \eta \left( \frac{\partial J}{\partial p} \right)_k = p_k - \eta \left( \hat{y}_k - y_{T_k} \right) \begin{bmatrix} x_k \ & 1 \end{bmatrix}
\]

\[
\begin{bmatrix}
w \\
b
\end{bmatrix}_{k+1} = \begin{bmatrix} w \\
b \\
b
\end{bmatrix}_k - \eta \left[(w_k^T x_k + b_k) - y_{T_k}\right] \begin{bmatrix} x_k \\
1 \\
1 \\
\end{bmatrix}
\]
Backpropagation for a Single Linear Neuron

- **Training set** \((n \text{ members})\)
  - Target outputs, \(y_T (1 \times n)\)
  - Feature set, \(X (m \times n)\)

\[
\begin{bmatrix}
    y_T \\
    X
\end{bmatrix} =
\begin{bmatrix}
    y_{T_1} & y_{T_2} & \ldots & y_{T_n} \\
    x_1 & x_2 & \ldots & x_n
\end{bmatrix}
\]

- **Initialize** \(w\) and \(b\)
  - Random set
  - Prior training result
- **Estimate** \(w\) and \(b\) recursively
  - **Off line** (random or repetitive sequence)
  - **On line** (measured training features and target)
- … until \(\frac{\partial J}{\partial \theta} \approx 0\)

Steepest-Descent Algorithm for a Single-Step Perceptron

- **Neuron output is discontinuous**

\[
y = s(r) = \begin{cases} 
1, & r > 0 \\
0, & r \leq 0 
\end{cases}
\]

- **Binary target output**
  - \(y_T = 0\) or \(1\), for classification

\[
\hat{y}_{k} - y_{T_{k}} = \begin{cases} 
1, & y_{k} = 1, \ y_{T_{k}} = 0 \\
0, & y_{k} = y_{T_{k}} \\
-1, & y_{k} = 0, \ y_{T_{k}} = 1
\end{cases}
\]
Training Variables for a Single Sigmoid Neuron

**Input-output characteristic and 1st derivative**

\[ y = s(r) = \frac{1}{1 + e^{-r}} \]

\[ \frac{dy}{dr} = \frac{ds(r)}{dr} = \frac{e^{-r}}{(1 + e^{-r})^2} = e^{-r}s^2(r) \]

\[ = \left[(1 + e^{-r}) - 1\right]s^2(r) = \left(\frac{1}{s(r)} - 1\right)s^2(r) \]

\[ = \left[\frac{1 - s(r)}{s(r)}\right]s^2(r) = [1 - s(r)]s(r) = (1 - y)y \]

**Training error and quadratic error cost**

\[ \varepsilon = \hat{y} - y_T \]

\[ J = \frac{1}{2} \varepsilon^2 = \frac{1}{2} (\hat{y} - y_T)^2 = \frac{1}{2} (\hat{y}^2 - 2\hat{y}y_T + y_T^2) \]

**Control parameter**

\[ p = \begin{bmatrix} w & b \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n+1} \end{bmatrix} \]

Training a Single Sigmoid Neuron

\[ \frac{\partial J}{\partial \mathbf{p}} = (\hat{y} - y_T) \frac{\partial y}{\partial \mathbf{p}} = (\hat{y} - y_T) \frac{\partial \hat{y}}{\partial r} \frac{\partial r}{\partial \mathbf{p}} \]

where

\[ r = \mathbf{w}^T \mathbf{x} + b \]

\[ \frac{\partial \hat{y}}{\partial r} = (1 - \hat{y})\hat{y} \]

\[ \frac{\partial r}{\partial \mathbf{p}} = \begin{bmatrix} \mathbf{x}^T & 1 \end{bmatrix} \]

\[ \frac{\partial J}{\partial \mathbf{p}} = (\hat{y} - y_T)(1 - \hat{y})\hat{y} \begin{bmatrix} \mathbf{x}^T & 1 \end{bmatrix} \]

\[ \mathbf{p}_{k+1} = \mathbf{p}_k - \eta \frac{\partial J}{\partial \mathbf{p}} \]

or

\[ \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_k - \eta (\hat{y}_k - y_T)(1 - \hat{y})\hat{y}_k \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} \]

**Diagram:**

- Input variable \( x_k \)
- Network output \( s(k) \)
- Target output \( y_T \)
- Adaptation algorithm
- Backpropagation
- Error calculation
- Parameter update
Training a Sigmoid Network

Two parameter vectors for 2-layer network

\[
p_{1,2} = \begin{bmatrix} w \ b \end{bmatrix}_{1,2} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n+1} \end{bmatrix}_{1,2}
\]

Output vector

\[
\hat{y} = u_2 = s_2(r_2) = s_2(W_2u_1 + b_2) \\
= s_2\left[ W_2 s_1(r_1) + b_2 \right] \\
= s_2\left[ W_2 s_1(W_1u_0 + b_1) + b_2 \right] \\
= s_2\left[ W_2 s_1(W_1x + b_1) + b_2 \right]
\]

Training a Sigmoid Network

\[
p_{1,2k} = p_{1,2k} - \eta \left( \frac{\partial J}{\partial p_{1,2}} \right)^T
\]

where

\[
\frac{\partial J}{\partial p_{1,2}} = (\hat{y} - y_1) \frac{\partial \hat{y}}{\partial p_{1,2}} = (\hat{y} - y_1) \frac{\partial \hat{y}}{\partial r_{12}} \frac{\partial r_{12}}{\partial p_{1,2}}
\]

where

\[
r_{1,2} = W_{1,2}u_{0,1} + b_{1,2}
\]

\[
\frac{\partial \hat{y}}{\partial r_2} = \mathbf{I}; \quad \frac{\partial \hat{y}}{\partial r_1} = \begin{bmatrix} (1 - \hat{y}_1) & 0 & \ldots & 0 \\ 0 & (1 - \hat{y}_2) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & (1 - \hat{y}_n) \end{bmatrix}
\]

\[
\frac{\partial r_2}{\partial p_1} = \mathbf{x}^T; \quad \frac{\partial r_2}{\partial p_2} = \mathbf{u}_1^T
\]
Small, Round Blue-Cell Tumor Classification Example

- Childhood cancers, including
  - Ewing’s sarcoma (EWS)
  - Burkitt’s Lymphoma (BL)
  - Neuroblastoma (NB)
  - Rhabdomyosarcoma (RMS)

  - 96 transcripts chosen from 2,308 probes for training
  - 63 EWS, BL, NB, and RMS training samples

- Source of data for my analysis

---

Overview of Present SRBCT Analysis

- **Transcript selection by t test**
  - 96 transcripts, 12 highest and lowest t values for each class
  - Overlap with Khan set: 32 transcripts

- **Ensemble averaging** of highest and lowest t values for each class

- **Cross-plot** of ensemble averages

- **Classification by sigmoidal neural network**

- **Validation** of neural network
  - Novel set simulation
  - Leave-one-out simulation
Clustering of SRBCT
Ensemble Averages

SRBCT Neural Network

Expression
Level Measurement

Class Prediction

Predicted Class

Input Layer
Hidden Layer
Output Layer

EWS (+) Average
EWS (-) Average
BL (+) Average
BL (-) Average
NB (+) Average
NB (-) Average
RMS (+) Average
RMS (-) Average

Class Comparison (Training)

Actual Class

Binary vector output (0,1) after rounding
Neural Network Training Set

Each input row is an ensemble average for a transcript set, normalized in \((-1,+1)\)

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Target Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
<td>EWS</td>
</tr>
<tr>
<td>Sample 2</td>
<td>EWS</td>
</tr>
<tr>
<td>Sample 3</td>
<td>EWS</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Sample 62</td>
<td>RMS</td>
</tr>
<tr>
<td>Sample 63</td>
<td>RMS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>EWS(+Average)</th>
<th>EWS(+Average)</th>
<th>EWS(+Average)</th>
<th>EWS(+Average)</th>
<th>EWS(+Average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWS(-Average)</td>
<td>EWS(-Average)</td>
<td>EWS(-Average)</td>
<td>EWS(-Average)</td>
<td>EWS(-Average)</td>
<td>EWS(-Average)</td>
</tr>
<tr>
<td>BL(+Average)</td>
<td>BL(+Average)</td>
<td>BL(+Average)</td>
<td>BL(+Average)</td>
<td>BL(+Average)</td>
<td>BL(+Average)</td>
</tr>
<tr>
<td>BL(-Average)</td>
<td>BL(-Average)</td>
<td>BL(-Average)</td>
<td>BL(-Average)</td>
<td>BL(-Average)</td>
<td>BL(-Average)</td>
</tr>
<tr>
<td>NB(+Average)</td>
<td>NB(+Average)</td>
<td>NB(+Average)</td>
<td>NB(+Average)</td>
<td>NB(+Average)</td>
<td>NB(+Average)</td>
</tr>
<tr>
<td>NB(-Average)</td>
<td>NB(-Average)</td>
<td>NB(-Average)</td>
<td>NB(-Average)</td>
<td>NB(-Average)</td>
<td>NB(-Average)</td>
</tr>
<tr>
<td>RMS(+Average)</td>
<td>RMS(+Average)</td>
<td>RMS(+Average)</td>
<td>RMS(+Average)</td>
<td>RMS(+Average)</td>
<td>RMS(+Average)</td>
</tr>
<tr>
<td>RMS(-Average)</td>
<td>RMS(-Average)</td>
<td>RMS(-Average)</td>
<td>RMS(-Average)</td>
<td>RMS(-Average)</td>
<td>RMS(-Average)</td>
</tr>
</tbody>
</table>

SRBCT Neural Network Training

- **Neural network**
  - 8 ensemble-average inputs
  - various # of sigmoidal neurons
  - 4 linear neurons
  - 4 outputs

- **Training accuracy**
  - Train on all 63 samples
  - Test on all 63 samples

- **100% accuracy**
Leave-One-Out Validation of SRBCT Neural Network

- Remove a single sample
- Train on remaining samples (125 times)
- Evaluate class of the removed sample
- Repeat for each of 63 samples
- 6 sigmoids: 99.96% accuracy (3 errors in 7,875 trials)
- 12 sigmoids: 99.99% accuracy (1 error in 7,875 trials)

Novel-Set Validation of SRBCT Neural Network

- Network always chooses one of four classes (i.e., “unknown” is not an option)
- Test on 25 novel samples (400 times each)
  - 5 EWS
  - 5 BL
  - 5 NB
  - 5 RMS
  - 5 samples of unknown class
- 99.96% accuracy on first 20 novel samples (3 errors in 8,000 trials)
- 0% accuracy on unknown classes
Observations of SRBCT Classification using Ensemble Averages

- t test identified strong features for classification in this data set
- Neural networks easily classified the four data types
- Caveat: Small, round blue-cell tumors occur in different tissue types
  - Ewing’s sarcoma: Bone tissue
  - Burkitt’s Lymphoma: Lymph nodes
  - Neuroblastoma: Nerve tissue
  - Rhabdomyosarcoma: Soft tissue

Gene expression (i.e., mRNA) variation may be linked to tissue differences as well as tumor genetics

Next Time:
Neural Networks - 2
Impulse, Pulse-Train, and Step Response of a LTI 2\textsuperscript{nd}-Order Neural Model
Cardiac Pacemaker and EKG Signals

Electrochemical Signaling at Axon Hillock and Synapse
Synaptic Strength Can Be Increased or Decreased by Externalities

- **Synapses**: learning elements of the nervous system
  - Action potentials enhanced or inhibited
  - Chemicals can modify signal transfer
  - Potentiation of pre- and post-synaptic cells

- **Adaptation/Learning (potentiation)**
  - Short-term
  - Long-term

---

Microarray Training Set

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>...</th>
<th>Sample n - 1</th>
<th>Sample n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene A Level</td>
<td>Tumor</td>
<td>Gene A Level</td>
<td>Tumor</td>
<td>...</td>
<td>Gene A Level</td>
<td>Normal</td>
</tr>
<tr>
<td>Gene B Level</td>
<td>Gene B Level</td>
<td>Gene B Level</td>
<td>Gene B Level</td>
<td>...</td>
<td>Gene B Level</td>
<td>Gene B Level</td>
</tr>
<tr>
<td>Gene m Level</td>
<td>Gene m Level</td>
<td>Gene m Level</td>
<td>Gene m Level</td>
<td>...</td>
<td>Gene m Level</td>
<td>Gene m Level</td>
</tr>
</tbody>
</table>
Microarray Training Data

- First row: Target classification
- 2nd-5th rows: Up-regulated genes
- 6th-10th rows: Down-regulated genes

Lab Analysis of Tissue Samples

Tumor = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0];

Normalized Data: Up-Regulated in Tumor

U22055 = [138 68 93 62 30 81 121 7 82 24 -2 -48 38 ... 82 118 55 103 102 87 62 69 14 101 25 47 48 75 ... 59 62 116 54 96 90 130 70 75 35 149 97 21 ... 14 15 25 19 -21 3 19 34];

Normalized Data: Up-Regulated in Normal

M96839 = [3 -23 3 12 -22 0 4 29 -73 32 5 -13 -16 14 ... 2 24 18 19 9 -13 -20 -3 -22 6 -5 -12 9 28 ... 20 -9 30 -15 18 1 -16 12 -9 3 -35 23 3 5 ... 33 29 47 19 32 34 20 55 49 20 10 36 70 50 ... 15 45 56 41 31 40];

Neural Network Classification Example

- ~7000 genes expressed in 62 microarray samples
  - Tumor = 1
  - Normal = 0

- 8 genes in strong feature set
  - 4 with Mean Tumor/Normal > 20:1
  - 4 with Mean Normal/Tumor > 20:1
  - and minimum variance within up-regulated set

Dukes Stages: A -> B -> C -> D
Neural Network Training Results:
Tumor/Normal Classification, 8 Genes, 4 Nodes

- Training begins with a random set of weights
- Adjustable parameters
  - Learning rate
  - Target error
  - Maximum # of epochs
- Non-unique sets of trained weights

Classification =
Columns 1 through 13
1 1 1 1 1 1 1 1 1 1 1 1 1 1
Columns 14 through 26
1 1 1 1 1 1 1 1 1 1 1 1 1 1
Columns 27 through 39
1 1 1 1 1 1 1 1 1 1 1 1 1 1
Columns 40 through 52
1 0 0 0 0 0 0 0 0 0 0 0 0 0
Columns 53 through 62
0 0 0 0 0 0 0 0 0 0 0 0 0 0

Binary network output (0,1) after rounding

Zero classification errors

Neural Network Training Results:
Tumor Stage/Normal Classification
8 Genes, 16 Nodes

- Colon cancer classification
  - 0 = Normal
  - 1 = Adenoma
  - 2 = A Tumor
  - 3 = B Tumor
  - 4 = C Tumor
  - 5 = D Tumor

Target =
[2 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5
5 5 5 5 5 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

One classification error

Scalar network output with varying magnitude
**Ranking by EWS t Values (Top and Bottom 12)**

- 24 transcripts selected from 12 highest and lowest t values for EWS vs. remainder

<table>
<thead>
<tr>
<th>Image ID</th>
<th>Transcript Description</th>
<th>EWS t Value</th>
<th>BL t Value</th>
<th>NB t Value</th>
<th>RMS t Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>110384</td>
<td>Fc fragment of IgG, receptor, transporter, alpha</td>
<td>12.04</td>
<td>-6.67</td>
<td>-6.17</td>
<td>-4.79</td>
</tr>
<tr>
<td>377461</td>
<td>caveolin 1, caveolae protein, 26kD</td>
<td>8.82</td>
<td>-5.97</td>
<td>-4.91</td>
<td>-4.78</td>
</tr>
<tr>
<td>812560</td>
<td>follicular lymphoma variant translocation 1</td>
<td>8.17</td>
<td>-4.31</td>
<td>-2.70</td>
<td>-5.48</td>
</tr>
<tr>
<td>491565</td>
<td>Cbp/p300-interacting transactivator, with Glu/Asp-rich carboxy-terminal domain</td>
<td>7.60</td>
<td>-5.82</td>
<td>-2.52</td>
<td>-3.68</td>
</tr>
<tr>
<td>841041</td>
<td>cyclin D1 (PRAD1: parathyroid adenomatosis 1)</td>
<td>6.84</td>
<td>-9.93</td>
<td>0.56</td>
<td>-4.30</td>
</tr>
<tr>
<td>1471841</td>
<td>ATPase, Na+/K+ transporting, alpha 1 polypeptide</td>
<td>6.65</td>
<td>-3.56</td>
<td>-2.72</td>
<td>-4.69</td>
</tr>
<tr>
<td>766702</td>
<td>protein tyrosine phosphatase, non-receptor type 13</td>
<td>6.54</td>
<td>-3.36</td>
<td>-2.07</td>
<td>-4.94</td>
</tr>
<tr>
<td>719922</td>
<td>glutathione S-transferase M1</td>
<td>6.17</td>
<td>-5.61</td>
<td>-5.16</td>
<td>-1.97</td>
</tr>
<tr>
<td>308497</td>
<td>NGFI-A binding protein 2 (ERG1 binding protein 2)</td>
<td>5.93</td>
<td>-6.74</td>
<td>-3.88</td>
<td>-1.21</td>
</tr>
<tr>
<td>345323</td>
<td>lymphotoxin alpha (TNF superfamily, member 1)</td>
<td>5.61</td>
<td>-8.05</td>
<td>-2.49</td>
<td>-1.19</td>
</tr>
<tr>
<td>786084</td>
<td>chromobox homolog 1 (Drosophila HP1 beta)</td>
<td>-5.04</td>
<td>-1.05</td>
<td>9.65</td>
<td>-0.62</td>
</tr>
<tr>
<td>796235</td>
<td>sarcooglobin, alpha (50kD dystrophin-associated glycoprotein)</td>
<td>-5.04</td>
<td>-3.31</td>
<td>-3.86</td>
<td>6.83</td>
</tr>
<tr>
<td>431307</td>
<td>N-acetylglucosamine receptor 1 (thyroid)</td>
<td>-5.04</td>
<td>2.64</td>
<td>2.19</td>
<td>0.64</td>
</tr>
<tr>
<td>859559</td>
<td>quinone oxidoreductase homolog</td>
<td>-5.23</td>
<td>-7.72</td>
<td>0.78</td>
<td>5.40</td>
</tr>
<tr>
<td>78254</td>
<td>cysteine and glycine-rich protein 2 (LIM domain only, smooth muscle)</td>
<td>-5.30</td>
<td>-4.11</td>
<td>2.20</td>
<td>3.68</td>
</tr>
<tr>
<td>448386</td>
<td>cyclin E1</td>
<td>-5.38</td>
<td>-0.42</td>
<td>3.76</td>
<td>0.14</td>
</tr>
<tr>
<td>68950</td>
<td>cyclin E1</td>
<td>-5.80</td>
<td>0.03</td>
<td>-1.58</td>
<td>5.10</td>
</tr>
<tr>
<td>774502</td>
<td>protein tyrosine phosphatase, non-receptor type 12</td>
<td>-5.80</td>
<td>-5.56</td>
<td>3.76</td>
<td>3.66</td>
</tr>
<tr>
<td>842920</td>
<td>inducible poly(A)-binding protein</td>
<td>-6.14</td>
<td>0.60</td>
<td>0.66</td>
<td>3.80</td>
</tr>
<tr>
<td>214772</td>
<td>ESTs</td>
<td>-6.39</td>
<td>-0.08</td>
<td>-0.22</td>
<td>4.56</td>
</tr>
<tr>
<td>295985</td>
<td>ESTs</td>
<td>-9.26</td>
<td>-0.13</td>
<td>3.24</td>
<td>2.95</td>
</tr>
</tbody>
</table>

Repeated for BL vs. remainder, NB vs. remainder, and RMS vs. remainder

**Comparison of Present SRBCT Set with Khan Top 10**

<table>
<thead>
<tr>
<th>Image ID</th>
<th>Gene Description</th>
<th>EWS Student t Value</th>
<th>BL Student t Value</th>
<th>NB Student t Value</th>
<th>RMS Student t Value</th>
<th>Most Significant Student t Value</th>
<th>Khan Gene Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>296448</td>
<td>insulin-like growth factor 2 (somatomedin A)</td>
<td>-4.789</td>
<td>-5.226</td>
<td>-1.185</td>
<td>5.998</td>
<td>RMS</td>
<td>RMS</td>
</tr>
<tr>
<td>207274</td>
<td>Human DNA for insulin-like growth factor II (IGF-2); exon 7 and additional ORF</td>
<td>-4.377</td>
<td>-5.424</td>
<td>-1.639</td>
<td>5.708</td>
<td>RMS</td>
<td>RMS</td>
</tr>
<tr>
<td>841641</td>
<td>cyclin D1 (PRAD1): parathyroid adenomatosis 1</td>
<td>6.841</td>
<td>-9.932</td>
<td>0.565</td>
<td>-4.300</td>
<td>BL (-)</td>
<td>EWS/NB</td>
</tr>
<tr>
<td>365826</td>
<td>growth arrest-specific 1</td>
<td>3.551</td>
<td>-8.438</td>
<td>-6.995</td>
<td>1.583</td>
<td>BL (-)</td>
<td>EWS/RMS</td>
</tr>
<tr>
<td>486787</td>
<td>calponin 3, acidic</td>
<td>-4.335</td>
<td>-6.354</td>
<td>2.446</td>
<td>2.605</td>
<td>BL (-)</td>
<td>RMS/NB</td>
</tr>
<tr>
<td>770394</td>
<td>Fc fragment of IgG, receptor, transporter, alpha</td>
<td>12.037</td>
<td>-6.673</td>
<td>-6.173</td>
<td>-4.792</td>
<td>EWS</td>
<td>EWS</td>
</tr>
<tr>
<td>244618</td>
<td>ESTs</td>
<td>-4.174</td>
<td>-4.822</td>
<td>-3.484</td>
<td>5.986</td>
<td>RMS</td>
<td>RMS</td>
</tr>
<tr>
<td>233721</td>
<td>insulin-like growth factor</td>
<td>0.058</td>
<td>-7.487</td>
<td>-1.599</td>
<td>2.184</td>
<td>BL (-)</td>
<td>Not BL</td>
</tr>
<tr>
<td>43733</td>
<td>glycoconjugate 2</td>
<td>4.715</td>
<td>-4.576</td>
<td>-3.834</td>
<td>-3.524</td>
<td>EWS</td>
<td>Not BL</td>
</tr>
<tr>
<td>295985</td>
<td>ESTs</td>
<td>-9.260</td>
<td>-0.133</td>
<td>3.237</td>
<td>2.948</td>
<td>EWS (-)</td>
<td>Not EWS</td>
</tr>
</tbody>
</table>

**Legend:**
- Red: both sets
- Black: Khan set only
MATLAB Program for Neural Network Analysis with Leave-One-Out Validation - Initialization(1)

'Measure-One-Out Neural Network Analysis of Khan Data'

% Neural Network with Vector Output
% Based on 63 Samples of 8 Positive and Negative t-Value Metagenes
% 12/5/2007

clear

Target = [ones(1,23) zeros(1,40)
    zeros(1,23) ones(1,8) zeros(1,32)
    zeros(1,31) ones(1,12) zeros(1,20)
    zeros(1,43) ones(1,20)];

TrainingData = [2.489   2.725   2.597   2.831   ...
    .....   .....   .....   .....   .....   ...
    .....   .....   .....   .....   .....   ...
    .....   .....   .....   .....   .....   ...
    .....];

MATLAB Program for Neural Network Analysis with Leave-One-Out Validation - Initialization(2)

% Validation Sample and Leave-One-Out Training Set

MisClass = 0;
iSamLog = [];
iRepLog = [];
ErrorLog = [];
OutputLog = [];
SizeTarget = size(Target);
SizeTD = size(TrainingData);

% Preprocessing of Training Data

[TrainingData,minp,maxp,tn,mint,maxt] = premnmx(TrainingData,Target);

premnmx has been replaced by mapminmax in MATLAB
MATLAB Program for Neural Network Analysis with Leave-One-Out Validation - Initialization(3)

```matlab
for iSam = 1:SizeTD(2)
    ValidSample = TrainingData(:,iSam);
    ReducedData = TrainingData;
    ReducedData(:,iSam) = [];
    ReducedTarget = Target;
    ReducedTarget(:,iSam) = [];
    Repeats = 2;
```

MATLAB Program for Neural Network Analysis with Leave-One-Out Validation - Training(1)

```matlab
for i = 1:Repeats
    Range = minmax(ReducedData);
    Neurons = [12,4];
    Nodes = {'logsig', 'purelin'};
    Beta = 0.5;
    Epochs = 200;
    Trainer = 'trainbr';

    Net = newff(Range,Neurons,Nodes,Trainer);
    Net.trainParam.show = 100;
    Net.trainParam.lr = Beta;
    Net.trainParam.epochs = Epochs;
    Net.trainParam.goal = 0.001;

    [Net,TrainingRecord] = train(Net,ReducedData,ReducedTarget);
    NetOutput = sim(Net, ReducedData);
    Rounded = round(NetOutput);
    Error = ReducedTarget - Rounded;
```

Check calling sequence of `newff`
% Validation with Single Sample
% NovelOutput = sim(Net,ValidSample);
LengthNO = length(NovelOutput);
NovelRounded = round(NovelOutput);
NovelRounded = max(NovelRounded,zeros(LengthNO,1));
NovelRounded = min(NovelRounded,ones(LengthNO,1));

% If no actual output is greater than 0.5, choose the largest
% for k = 1:SizeNO(2)
if (isequal(NovelRounded,zeros(LengthNO,1)))
    [c,j] = max(NovelOutput);
    NovelRounded(j,1) = 1;
end
AbsDiff = abs(NovelOutput - NovelRounded);

% If two rounded outputs are "1", choose the one whose actual output is
% closest to "1"
for j = 1:(LengthNO - 1)
    if NovelRounded(j) == 1
        for k = (j + 1):LengthNO
            if NovelRounded(k) == 1
                if (AbsDiff(j) < AbsDiff(k))
                    NovelRounded(k) = 0;
                else
                    NovelRounded(j) = 0;
                end
            end
        end
    end
end

NovelError = Target(:,iSam) - NovelRounded;
MATLAB Program for Neural Network Analysis with Leave-One-Out Validation - Training(4)

if (~isequal(NovelError,zeros(LengthNO,1)))
    MisClass = MisClass + 1;
iSamLog = [iSamLog iSam];
iRepLog = [iRepLog i];
ErrorLog = [ErrorLog NovelError];
OutputLog = [OutputLog NovelOutput];
end
end
end

MisClass
iSamLog
iRepLog
ErrorLog
OutputLog

Trials = iSam * Repeats

Algebraic Training of a Neural Network
Algebraic Training for Exact Fit to a Smooth Function

- Smooth functions define equilibrium control settings at many operating points
- Neural network required to fit the functions

Algorithm for Network Training

\begin{align*}
\delta T_c &: \text{Throttle command} \\
\delta S_c &: \text{Spoiler command} \\
\delta A_c &: \text{Aileron command} \\
\delta R_c &: \text{Rudder command} \\
\delta \theta_c &: \text{Pitch angle command} \\
\delta \psi_c &: \text{Yaw rate command}
\end{align*}
Results for Network Training

- 45-node example
- Algorithm is considerably faster than search methods

<table>
<thead>
<tr>
<th>Algorithm:</th>
<th>Time (Scaled):</th>
<th>Flops:</th>
<th>Lines of code (MATLAB®):</th>
<th>Epochs:</th>
<th>Final error:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebraic</td>
<td>1</td>
<td>$2 \times 10^5$</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Levenberg-Marquardt</td>
<td>50</td>
<td>$5 \times 10^7$</td>
<td>150</td>
<td>6</td>
<td>$10^{-26}$</td>
</tr>
<tr>
<td>Resilient Backprop.</td>
<td>150</td>
<td>$1 \times 10^7$</td>
<td>100</td>
<td>150</td>
<td>0.006</td>
</tr>
</tbody>
</table>