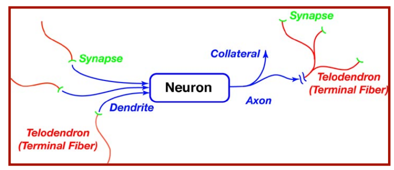
Neural Networks - 1

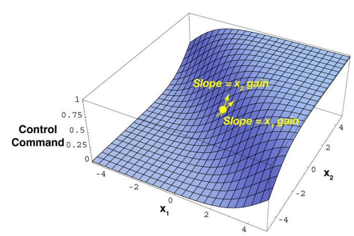
Robert Stengel
Robotics and Intelligent Systems,
MAE 345, Princeton University, 2013

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



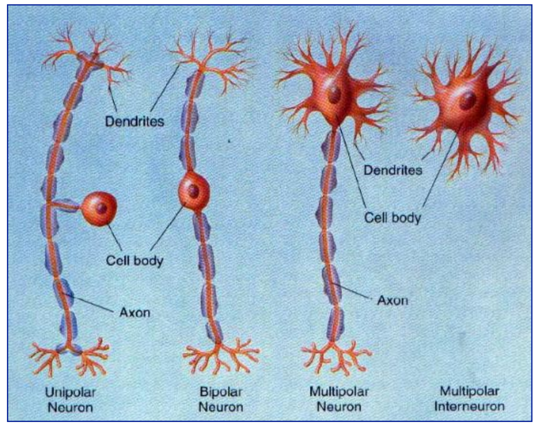
Applications of Computational Neural Networks



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

Neurons

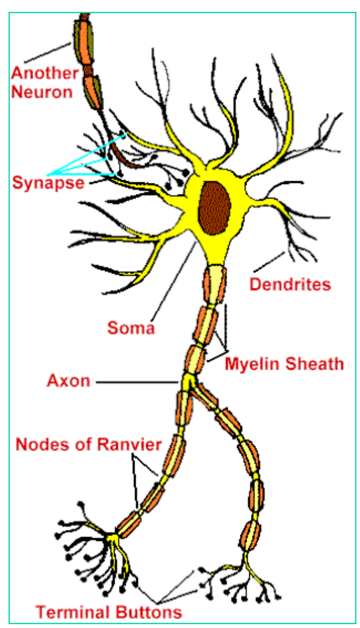
- Biological cells with significant electrochemical activity
- ~10-100 billion neurons in the brain
- Inputs from thousands of other neurons
- Output is scalar, but may have thousands of branches

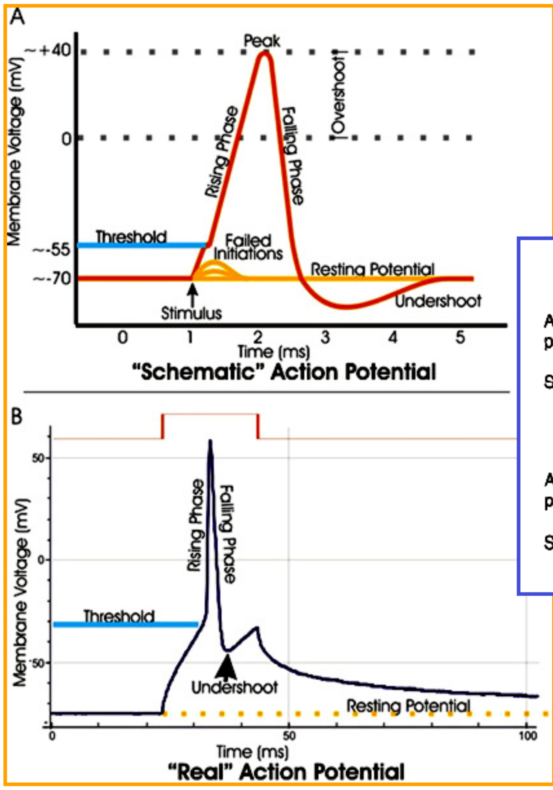


- Afferent (unipolar) neurons send signals from organs and the periphery to the central nervous system
- Efferent (multipolar) neurons issue commands from the CNS to effector (e.g., muscle) cells
- Interneurons (multipolar) send signals between neurons in the central nervous system
- Signals are ionic,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

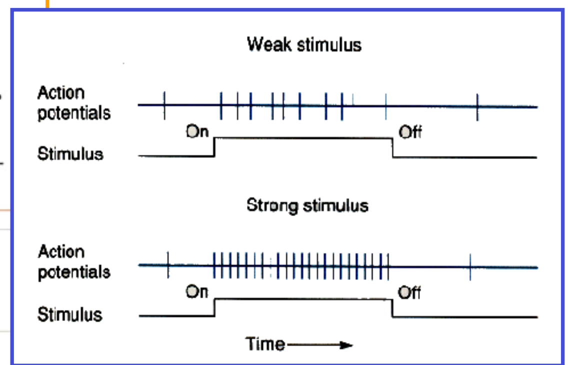
Activation Input to Soma Causes Change in Output Potential

- 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



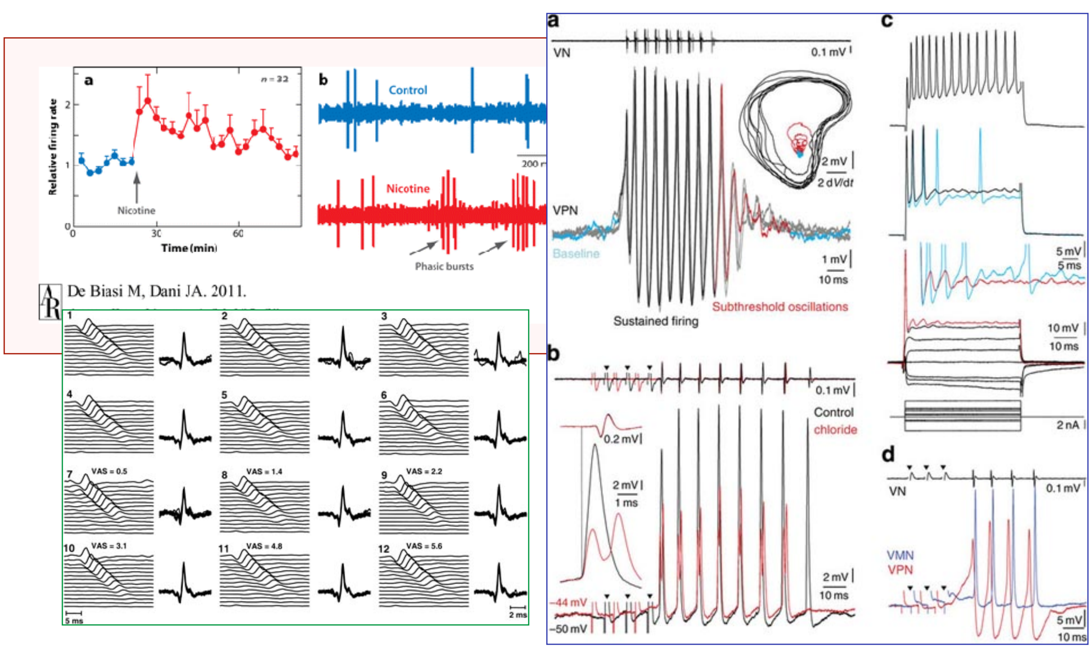


Neural Action Potential

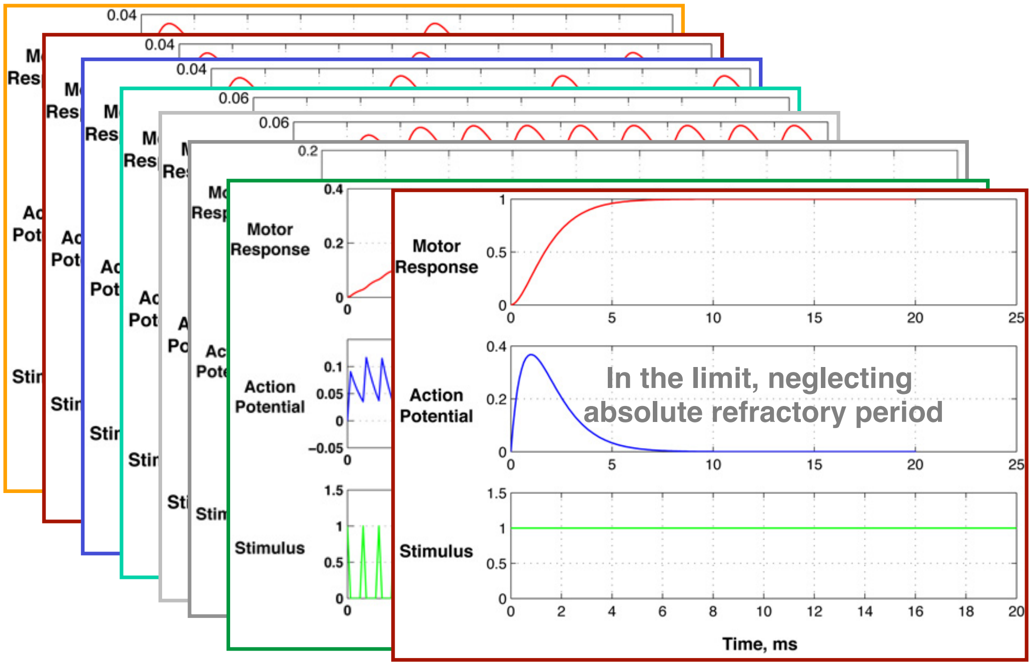


- **Maximum Firing Rate**: 500/sec
- **Refractory Period**: Minimum time increment between action potential firing ~ 1-2 msec

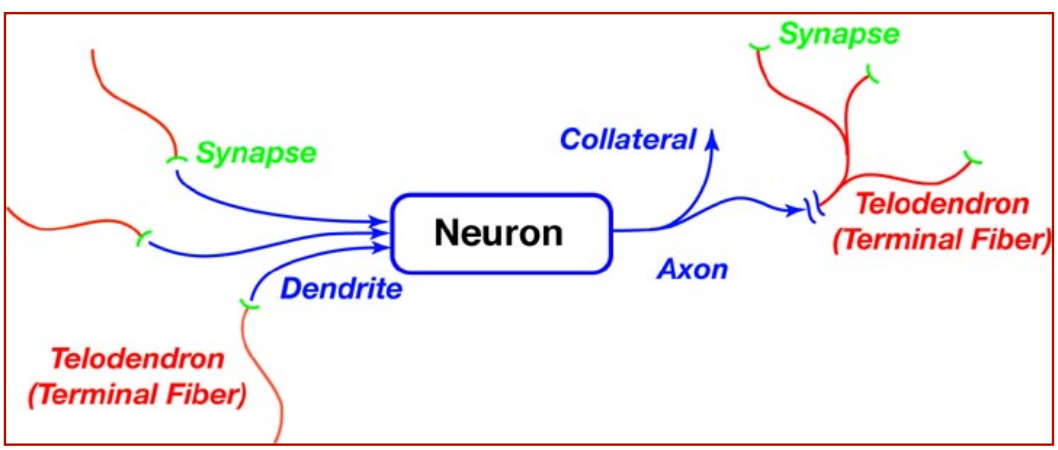
Some Recorded Action Potential Pulse Trains



Impulse, Pulse-Train, and Step Response of a LTI 2nd-Order Neural Model



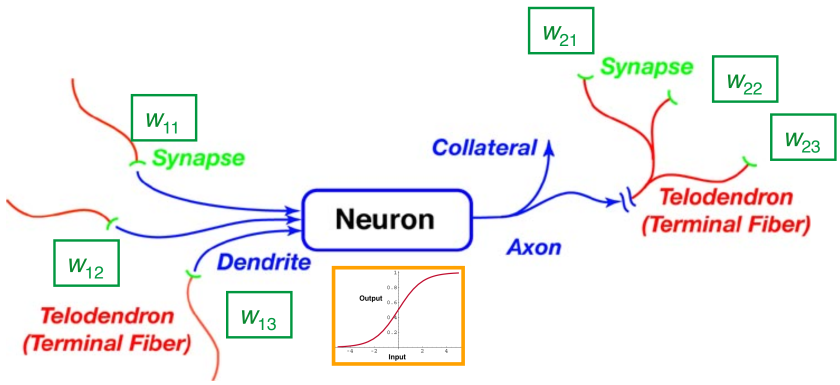
Multipolar Neuron



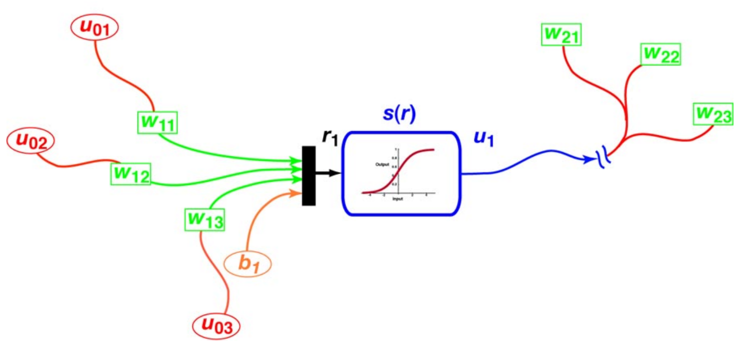
Mathematical Model of Neuron Components

Synapse effects represented by weights (gains or multipliers)

Neuron firing frequency is modeled by linear gain or nonlinear element



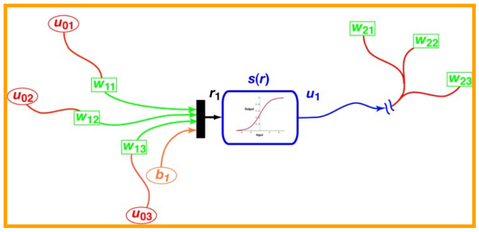
The Neuron Function



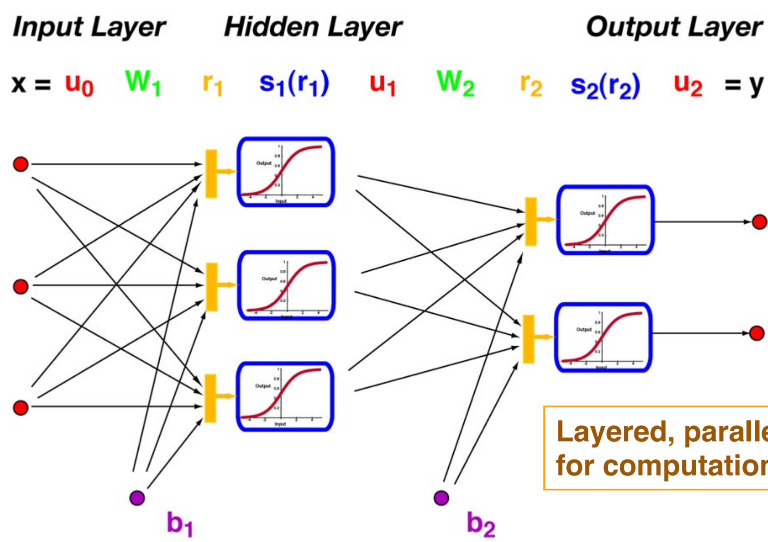
- **Vector input, u, to a single neuron**
  - Sensory input or output from upstream neurons
  - Linear operation produces scalar, r
  - Add bias, b, for zero adjustment
- **Scalar output, u, of a single neuron (or node)**
  - Scalar linear or nonlinear operation, $s(r)$

$$r = \mathbf{w}^T \mathbf{u} + b$$

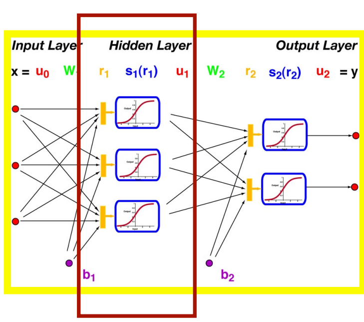
$$u = s(r)$$



Layout of a Neural Network



Layered, parallel structure for computation



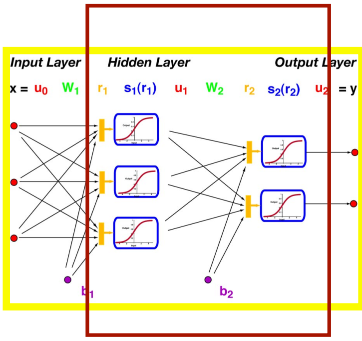
Input-Output Characteristics of a Neural Network Layer

- Single layer
  - Number of inputs = n
    - dim(u) = (n x 1)
  - Number of nodes = m
    - dim(r) = dim(b) = dim(s) = (m x 1)

$$\mathbf{r} = \mathbf{W}\mathbf{u} + \mathbf{b}$$

$$\mathbf{u} = \mathbf{s}(\mathbf{r})$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \\ \mathbf{w}_n^T \end{bmatrix}$$



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

$$
\begin{aligned}
\mathbf{y} &= \mathbf{u}_2 \\
&= \mathbf{s}_2(\mathbf{r}_2) = \mathbf{s}_2(\mathbf{W}_2\mathbf{u}_1 + \mathbf{b}_2) \\
&= \mathbf{s}_2\left[\mathbf{W}_2\ \mathbf{s}_1(\mathbf{r}_1) + \mathbf{b}_2\right] \\
&= \mathbf{s}_2\left[\mathbf{W}_2\ \mathbf{s}_1(\mathbf{W}_1\mathbf{u}_0 + \mathbf{b}_1) + \mathbf{b}_2\right] \\
&= \mathbf{s}_2\left[\mathbf{W}_2\ \mathbf{s}_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2\right]
\end{aligned}
$$

Is a Neural Network Serial or Parallel?

3rd-degree power series

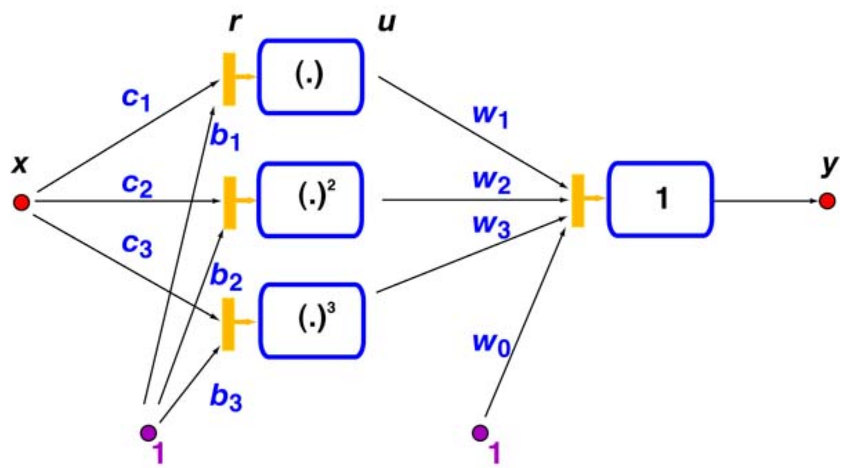
4 coefficients

Express as a neural network?

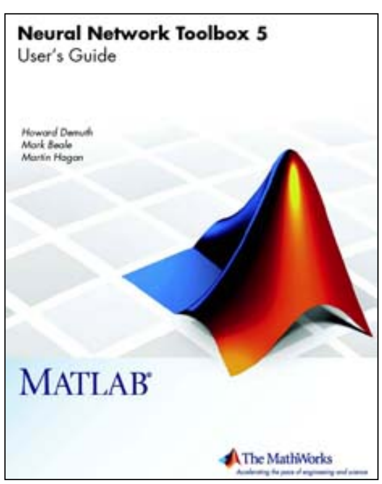
$$
\begin{aligned}
y &= a_0 + a_1x + a_2x^2 + a_3x^3 \\
&= a_0' + a_1'r + a_2'r^2 + a_3'r^3 \\
&= a_0' + a_1'(c_1x + b_1) + a_2'(c_1x + b_2)^2 + a_3'(c_1x + b_3)^3 \\
&= w_0 + w_1s_1(u) + w_2s_2(u) + w_3s_3(u)
\end{aligned}
$$

Is a Neural Network Serial or Parallel?

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



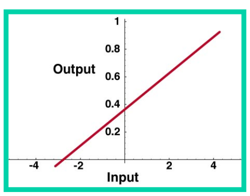
MATLAB Neural Network Toolbox



- **Implementation of many neural network architectures**
- **Common calling sequences**
- **Pre- and post-processing**
- **Command-line and GUI**

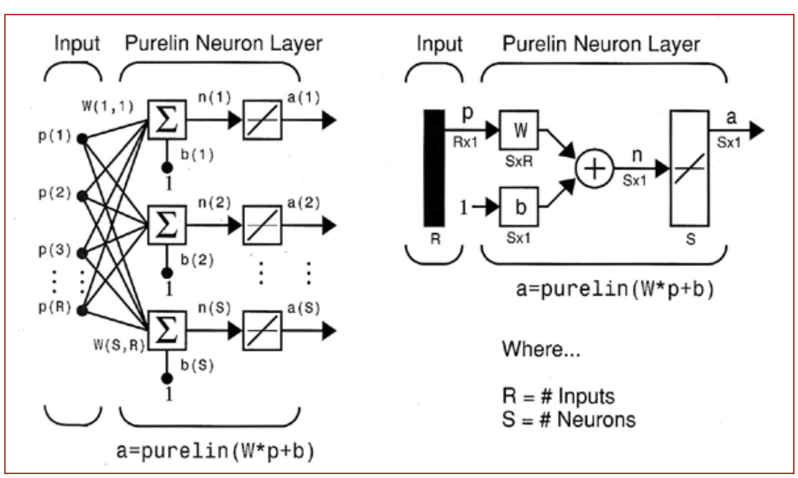
MATLAB Training and Evaluation of "Backpropagation" Neural Networks

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



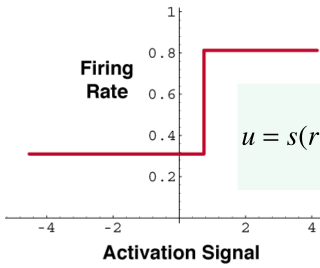
Linear Neural Network

- 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



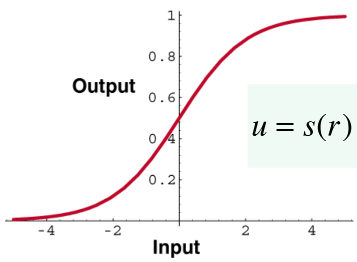
Idealizations of Nonlinear Neuron Input-Output Characteristic

Step function ("Perceptron")



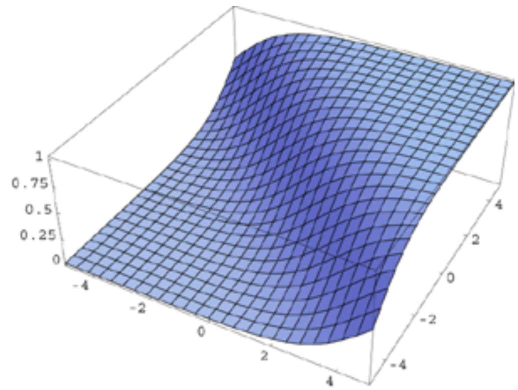
$$u = s(r) = \begin{cases} 1, & r > 0 \\ 0, & r \le 0 \end{cases}$$

Logistic sigmoid function



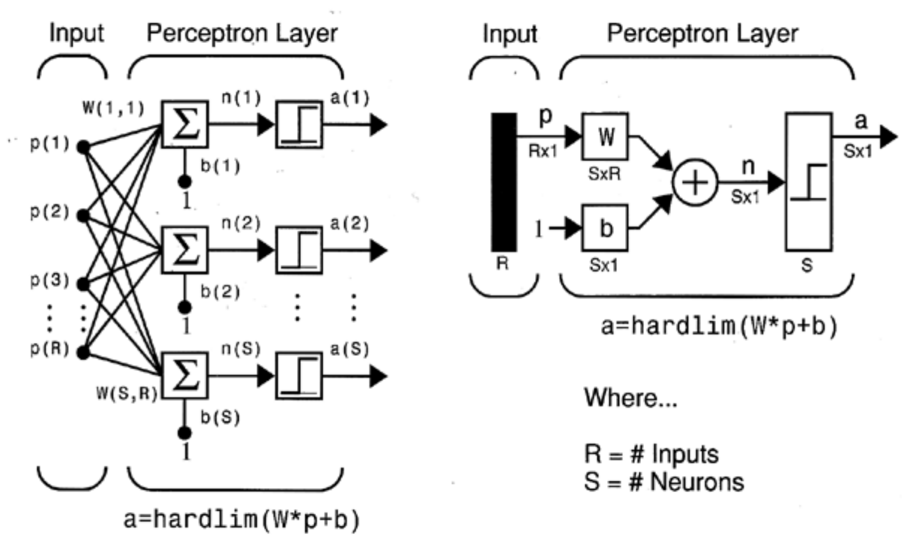
$$u = s(r) = \frac{1}{1 + e^{-r}}$$

Sigmoid with two inputs, one output



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

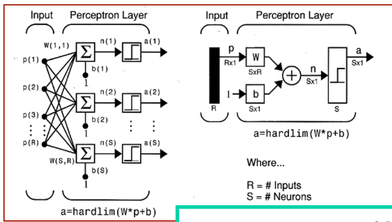
Perceptron Neural Network



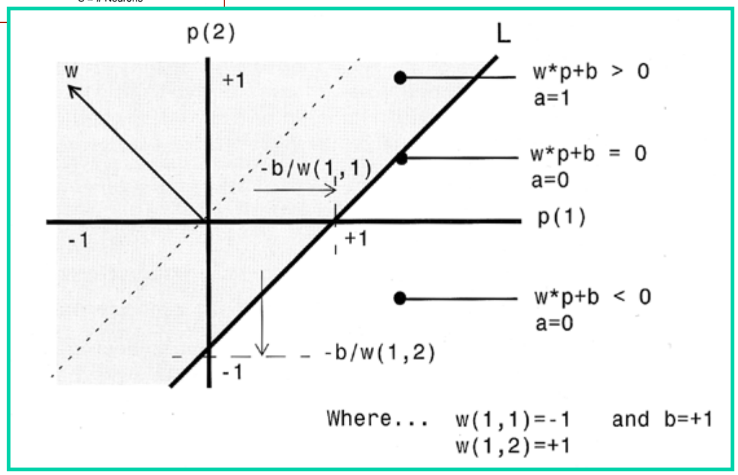
Each node is a step function

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

$$u = s(\mathbf{w}^T\mathbf{x}+b) = \begin{cases} 1, & (\mathbf{w}^T\mathbf{x}+b) > 0 \\ 0, & (\mathbf{w}^T\mathbf{x}+b) \le 0 \end{cases}$$

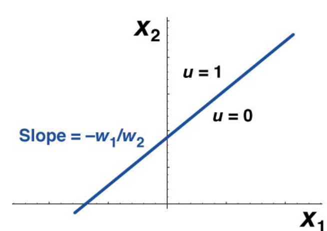
Two inputs, single step function

Discriminant

$$w_1x_1 + w_2x_2 + b = 0$$

$$or \quad x_2 = \frac{-1}{w_2}(w_1x_1 + b)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



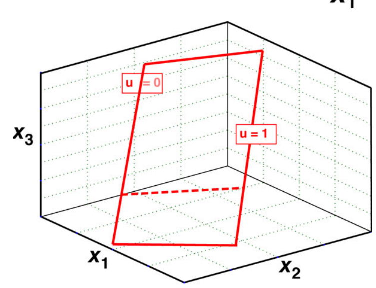
Three inputs, single step function

Discriminant

$$w_1x_1 + w_2x_2 + w_3x_3 + b = 0$$

$$or \quad x_3 = \frac{-1}{w_3}(w_1x_1 + w_2x_2 + b)$$

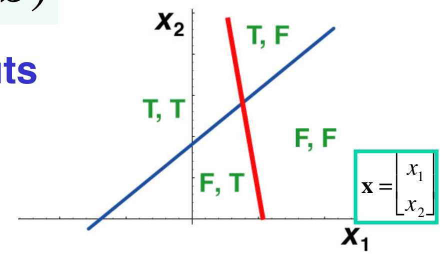
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

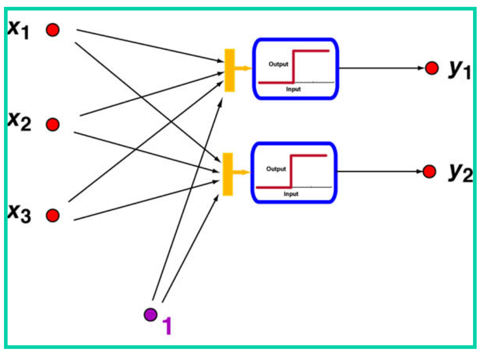


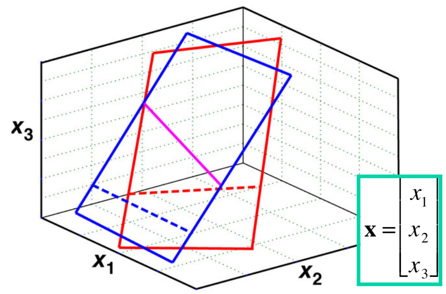
Single-Layer, Multi-Node Perceptron Discriminants

$$\mathbf{u} = \mathbf{s}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

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





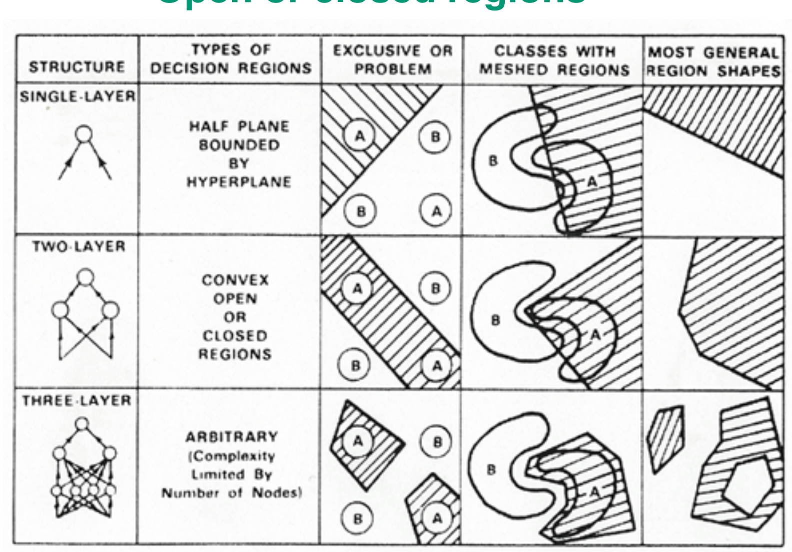


Multi-Layer Perceptrons Can Classify With Boundaries or Clusters

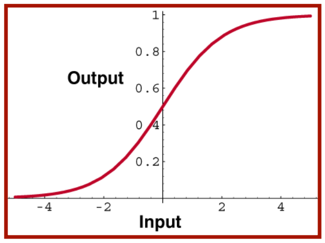
Classification capability of multi-layer perceptrons

Classifications of classifications

Open or closed regions



STRUCTURE	TYPES OF DECISION REGIONS	EXCLUSIVE OR PROBLEM	CLASSES WITH MESHED REGIONS	MOST GENERAL REGION SHAPES
SINGLE-LAYER	HALF PLANE BOUNDED BY HYPERPLANE	A B B A	B A	
TWO-LAYER	CONVEX OPEN OR CLOSED REGIONS	A B B A	B A	
THREE-LAYER	ARBITRARY (Complexity Limited By Number of Nodes)	A B B A	B A	



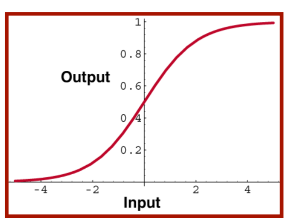
Sigmoid Activation Functions

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

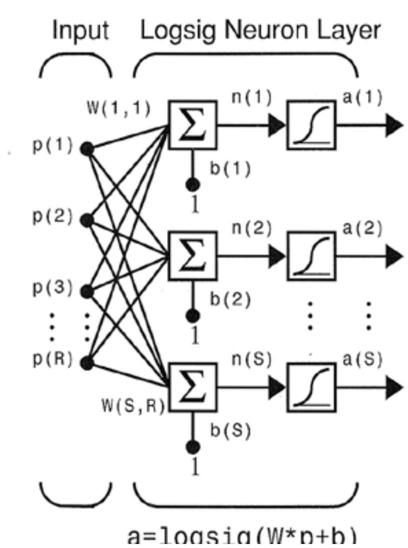
$$u = s(r) = \frac{1}{1+e^{-r}}$$

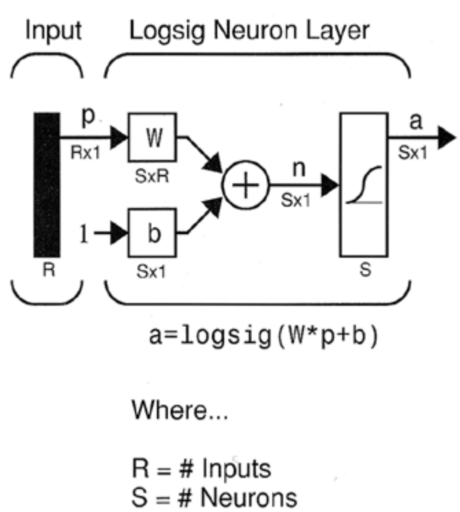
$$u = s(r) = \tanh r = \frac{1-e^{-2r}}{1+e^{-2r}}$$

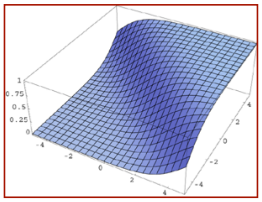
$$u = s(r) = \frac{r^2}{1+r^2}$$



Sigmoid Neural Network

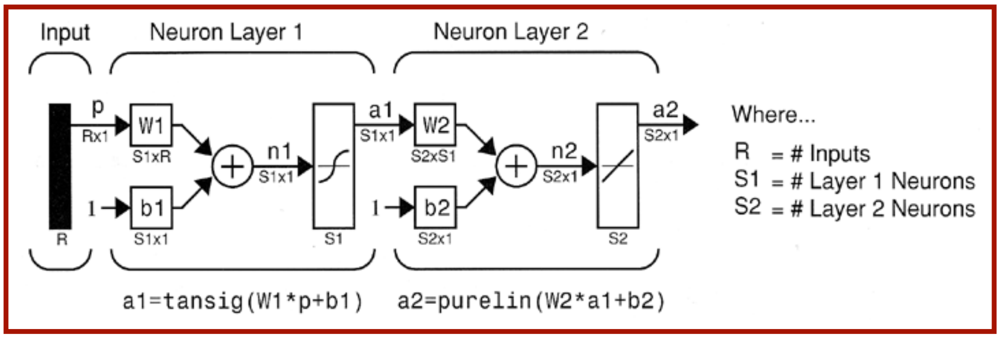


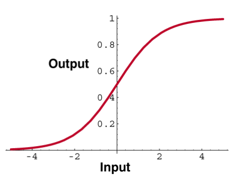




Single Sigmoid Layer is Sufficient …

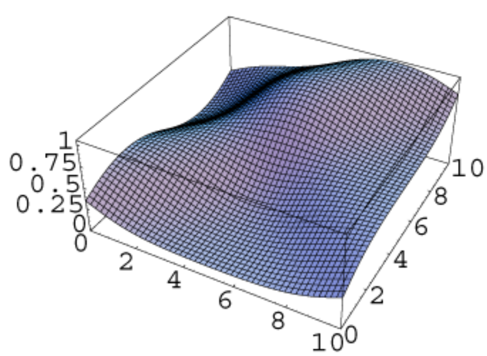
- Sigmoid network with single hidden layer can approximate any continuous function
- Therefore, additional sigmoid layers are unnecessary
- Typical sigmoid network contains
  - Single sigmoid hidden layer (nonlinear fit)
  - Single linear output layer (scaling)

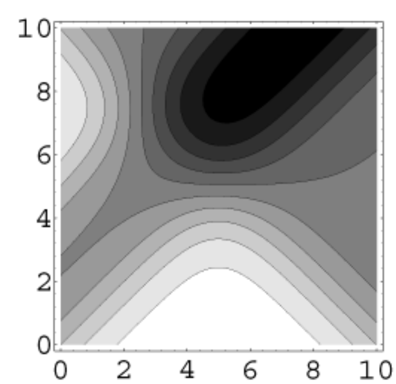




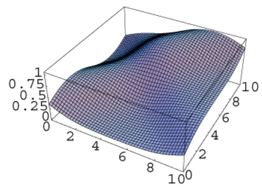
Typical Sigmoid Neural Network Output

Classification is not limited to linear discriminants



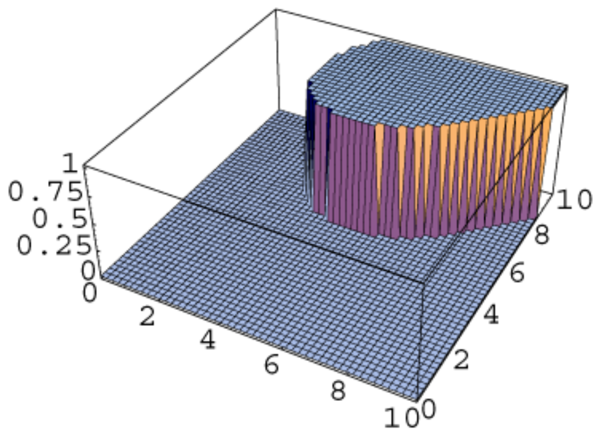


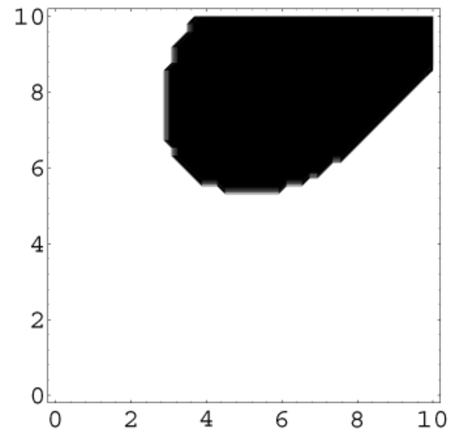
Sigmoid network can approximate a continuous nonlinear function to arbitrary accuracy with a single hidden layer



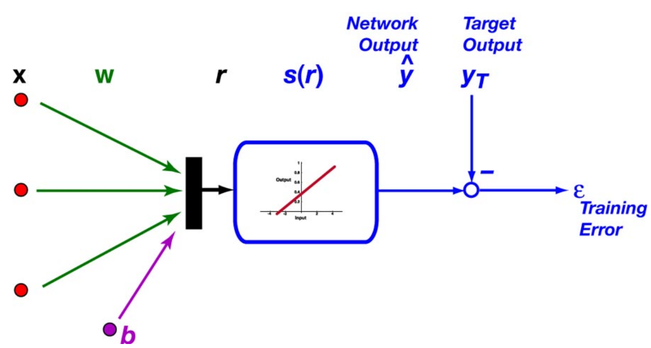
Thresholded Neural Network Output

Threshold gives "yes/no" output





Training Error and Cost for a Single Linear Neuron



$$\hat{y} = r = \hat{\mathbf{w}}^T \mathbf{x} + \hat{b}$$

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

$$\varepsilon = \hat{y} - y_T$$

$$J = \frac{1}{2}\varepsilon^2 = \frac{1}{2}\left(\hat{y} - y_T\right)^2 = \frac{1}{2}\left(\hat{y}^2 - 2\hat{y}\,y_T + {y_T}^2\right)$$

Linear Neuron Gradient

$$\hat{y} = r = \mathbf{w}^T\mathbf{x} + b$$
$$\frac{d\hat{y}}{dr} = 1$$

$$\varepsilon = \hat{y} - y_T$$
$$J = \frac{1}{2}\varepsilon^2 = \frac{1}{2}\left(\hat{y} - y_T\right)^2 = \frac{1}{2}\left(\hat{y}^2 - 2\hat{y}\,y_T + y_T{}^2\right)$$

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

$$\mathbf{p} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ \ldots \\ p_{n+1} \end{bmatrix}$$

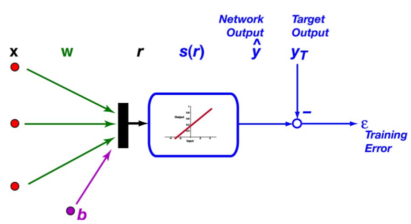
- Optimality condition $\dfrac{\partial J}{\partial \mathbf{p}} = \mathbf{0}$

- Gradient

$$\frac{\partial J}{\partial \mathbf{p}} = \left(\hat{y} - y_T\right)\frac{\partial y}{\partial \mathbf{p}} = \left(\hat{y} - y_T\right)\frac{\partial y}{\partial r}\frac{\partial r}{\partial \mathbf{p}}$$

where

$$\frac{\partial r}{\partial \mathbf{p}} = \begin{bmatrix} \dfrac{\partial r}{\partial p_1} & \dfrac{\partial r}{\partial p_2} & \ldots & \dfrac{\partial r}{\partial p_{n+1}} \end{bmatrix} = \frac{\partial\left(\mathbf{w}^T\mathbf{x} + b\right)}{\partial \mathbf{p}} = \begin{bmatrix} \mathbf{x}^T & 1 \end{bmatrix}$$



Steepest-Descent Learning for a Single Linear Neuron

Gradient

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

Steepest-descent algorithm

$\eta =$ learning rate
$k =$ iteration index(epoch)

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \eta\left(\frac{\partial J}{\partial \mathbf{p}}\right)_k^T = \mathbf{p}_k - \eta\left(\hat{y}_k - y_{T_k}\right)\begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_k - \eta\left[\left(\mathbf{w}_k{}^T\mathbf{x}_k + b_k\right) - y_{T_k}\right]\begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}$$

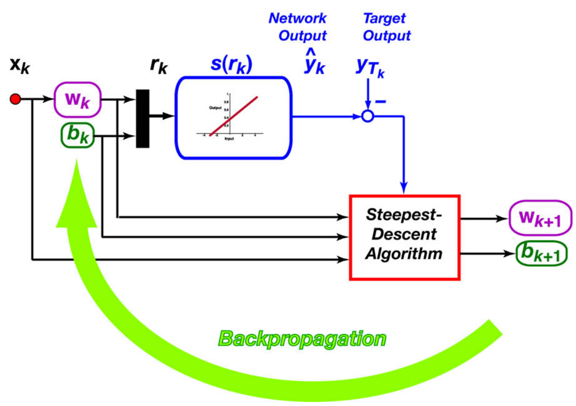
Backpropagation for a Single Linear Neuron

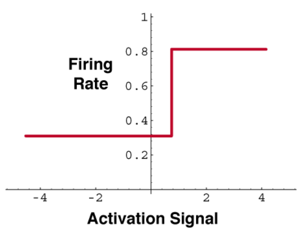
- **Training set** (n members)
  - Target outputs, $\mathbf{y}_T$ (1 x n)
  - Feature set, $\mathbf{X}$ (m x n)

$$\begin{bmatrix} \mathbf{y}_T \\ \mathbf{X} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{T_1} & \mathbf{y}_{T_2} & \cdots & \mathbf{y}_{T_n} \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix}$$

- **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 $\partial J/\partial p \sim 0$**

$$\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_k - \eta\left[\left(\mathbf{w}_k^T \mathbf{x}_k + b_k\right) - y_{T_k}\right]\begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}$$





Steepest-Descent Algorithm for a Single-Step Perceptron

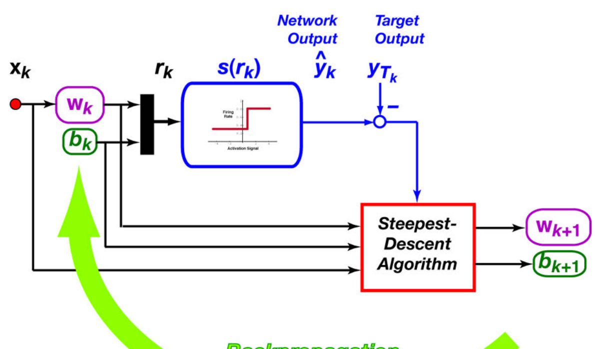
- **Neuron output is discontinuous**

$$y = s(r) = \begin{cases} 1, & r > 0 \\ 0, & r \le 0 \end{cases}$$

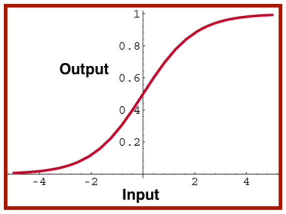
- **Binary target output**
  - y_T = 0 or 1, for classification

$$\left(\hat{y}_k - y_{T_k}\right) = \begin{cases} 1, & y_k = 1, \quad y_{T_k} = 0 \\ 0, & y_k = y_{T_k} \\ -1, & y_k = 0, \quad y_{T_k} = 1 \end{cases}$$

$$\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_k - \eta\left[\hat{y}_k - y_{T_k}\right]\begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} \mathbf{y}_T \\ \mathbf{X} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{T_1} & \mathbf{y}_{T_2} & \cdots & \mathbf{y}_{T_n} \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix}$$



Training Variables for a Single Sigmoid Neuron

Input-output characteristic and 1st derivative

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

$$\frac{dy}{dr} = \frac{ds(r)}{dr} = \frac{e^{-r}}{\left(1+e^{-r}\right)^2} = e^{-r}s^2(r)$$

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

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

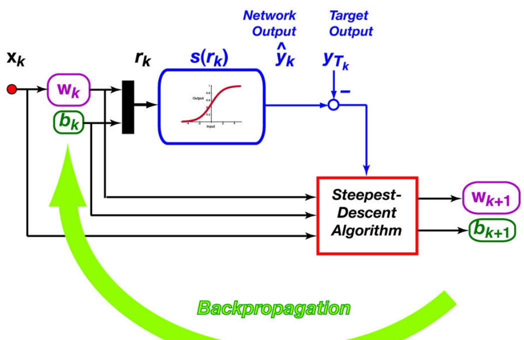
Training error and quadratic error cost

$$\varepsilon = \hat{y} - y_T$$

$$J = \frac{1}{2}\varepsilon^2 = \frac{1}{2}\left(\hat{y}-y_T\right)^2 = \frac{1}{2}\left(\hat{y}^2 - 2\hat{y}y_T + y_T^{\,2}\right)$$

Control parameter

$$\mathbf{p} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ \ldots \\ p_{n+1} \end{bmatrix}$$



Training a Single Sigmoid Neuron

$$\frac{\partial J}{\partial \mathbf{p}} = \left(\hat{y}-y_T\right)\frac{\partial y}{\partial \mathbf{p}} = \left(\hat{y}-y_T\right)\frac{\partial \hat{y}}{\partial r}\frac{\partial r}{\partial \mathbf{p}}$$

where

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

$$\frac{d\hat{y}}{dr} = \left(1-\hat{y}\right)\hat{y}$$

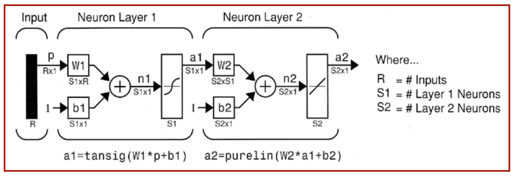
$$\frac{\partial r}{\partial \mathbf{p}} = \begin{bmatrix} \mathbf{x}^T & 1 \end{bmatrix}$$

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

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \eta\left(\frac{\partial J}{\partial \mathbf{p}}\right)_k^T$$

or

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



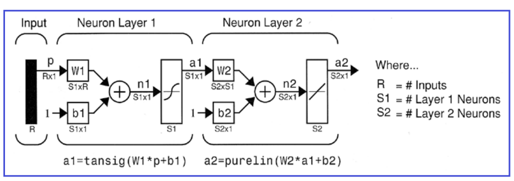
Training a Sigmoid Network

Two parameter vectors for 2-layer network

$$\mathbf{p}_{1,2} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}_{1,2} = \begin{bmatrix} p_1 \\ p_2 \\ \ldots \\ p_{n+1} \end{bmatrix}_{1,2}$$

Output vector

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{u}_2 \\ &= \mathbf{s}_2(\mathbf{r}_2) = \mathbf{s}_2(\mathbf{W}_2\mathbf{u}_1 + \mathbf{b}_2) \\ &= \mathbf{s}_2\left[\mathbf{W}_2\ \mathbf{s}_1(\mathbf{r}_1) + \mathbf{b}_2\right] \\ &= \mathbf{s}_2\left[\mathbf{W}_2\ \mathbf{s}_1(\mathbf{W}_1\mathbf{u}_0 + \mathbf{b}_1) + \mathbf{b}_2\right] \\ &= \mathbf{s}_2\left[\mathbf{W}_2\ \mathbf{s}_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2\right] \end{aligned}$$



Training a Sigmoid Network

$$\mathbf{p}_{1,2_k} = \mathbf{p}_{1,2_k} - \eta\left(\frac{\partial J}{\partial \mathbf{p}_{1,2}}\right)_k^T$$

where

$$\frac{\partial J}{\partial \mathbf{p}_{1,2}} = (\hat{\mathbf{y}} - \mathbf{y}_T)\frac{\partial \mathbf{y}}{\partial \mathbf{p}_{1,2}} = (\hat{\mathbf{y}} - \mathbf{y}_T)\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{r}_{1,2}}\frac{\partial \mathbf{r}_{1,2}}{\partial \mathbf{p}_{1,2}}$$

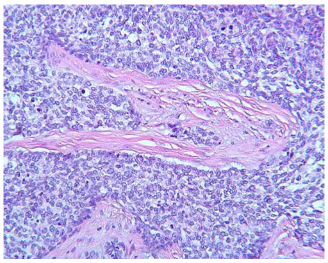
where

$$\mathbf{r}_{1,2} = \mathbf{W}_{1,2}\mathbf{u}_{0,1} + \mathbf{b}_{1,2}$$

$$\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{r}_2} = \mathbf{I}; \quad \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{r}_1} = \begin{bmatrix} (1-\hat{y}_1)\hat{y}_1 & 0 & \ldots & 0 \\ 0 & (1-\hat{y}_2)\hat{y}_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & 0 \\ 0 & 0 & \ldots & (1-\hat{y}_n)\hat{y}_n \end{bmatrix}$$

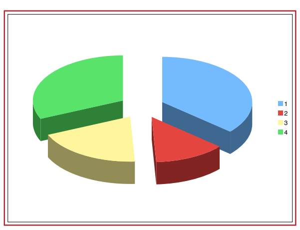
$$\frac{\partial \mathbf{r}_1}{\partial \mathbf{p}_1} = \begin{bmatrix} \mathbf{x}^T & 1 \end{bmatrix}; \quad \frac{\partial \mathbf{r}_2}{\partial \mathbf{p}_2} = \begin{bmatrix} \mathbf{u}_1^T & 1 \end{bmatrix}$$

Small, Round Blue-Cell Tumor Classification Example



Desmoplastic small, round blue-cell tumors

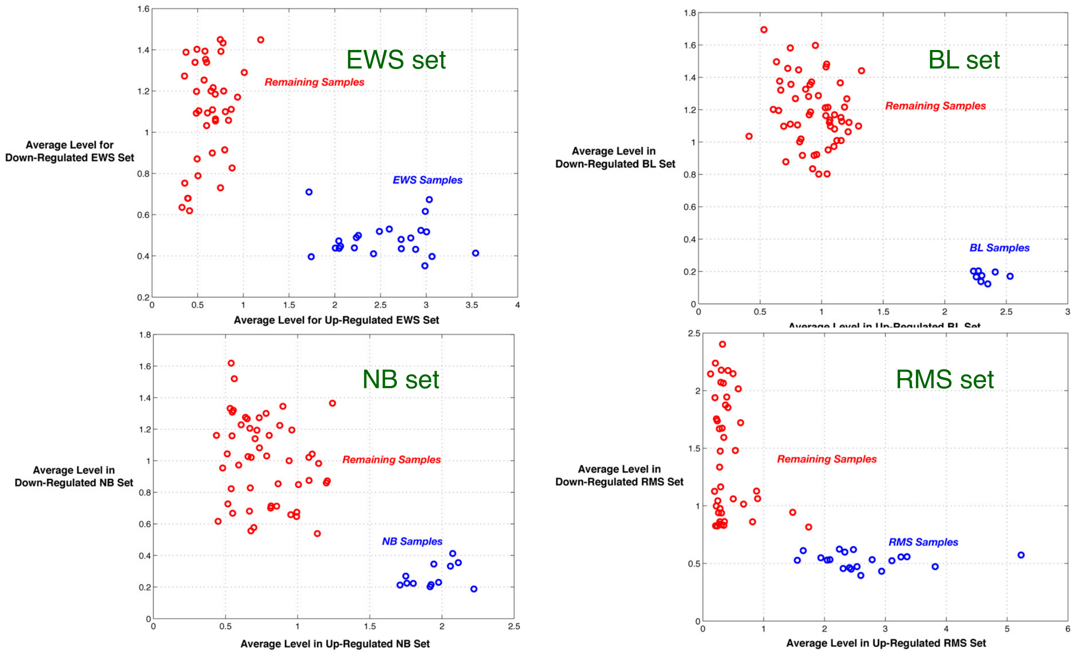
- Childhood cancers, including
  - Ewing's sarcoma (EWS)
  - Burkitt's Lymphoma (BL)
  - Neuroblastoma (NB)
  - Rhabdomyosarcoma (RMS)
- cDNA microarray analysis presented by **J. Khan, *et al.*, *Nature Medicine*, 2001, 673-679.**
  - 96 transcripts chosen from 2,308 probes for training
  - 63 EWS, BL, NB, and RMS training samples
- **Source of data for my analysis**



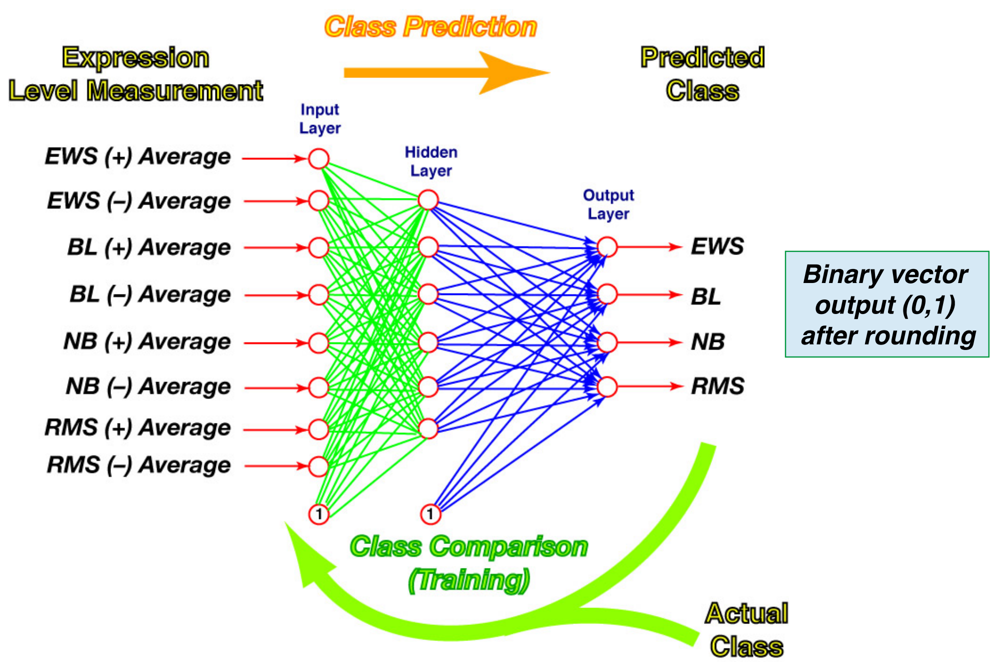
Overview of Present SRBCT Analysis

- **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



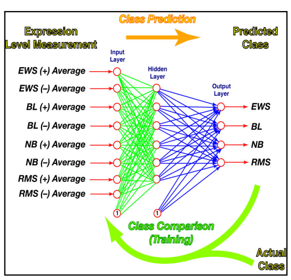
SRBCT Neural Network



Neural Network Training Set

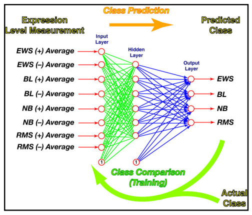
Each input row is an ensemble average for a transcript set, normalized in (–1,+1)

Identifier	Sample 1	Sample 2	Sample 3	...	Sample 62	Sample 63
Target Output	EWS	EWS	EWS	...	RMS	RMS
Transcript Training Set	EWS(+)Average	EWS(+)Average	EWS(+)Average	...	EWS(+)Average	EWS(+)Average
	EWS(–)Average	EWS(–)Average	EWS(–)Average	...	EWS(–)Average	EWS(–)Average
	BL(+)Average	BL(+)Average	BL(+)Average	...	BL(+)Average	BL(+)Average
	BL(–)Average	BL(–)Average	BL(–)Average	...	BL(–)Average	BL(–)Average
	NB(+)Average	NB(+)Average	NB(+)Average	...	NB(+)Average	NB(+)Average
	NB(–)Average	NB(–)Average	NB(–)Average	...	NB(–)Average	NB(–)Average
	RMS(+)Average	RMS(+)Average	RMS(+)Average	...	RMS(+)Average	RMS(+)Average
	RMS(–)Average	RMS(–)Average	RMS(–)Average	...	RMS(–)Average	RMS(–)Average



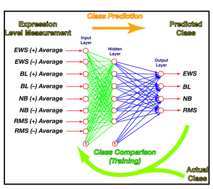
SRBCT Neural Network Training

- **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

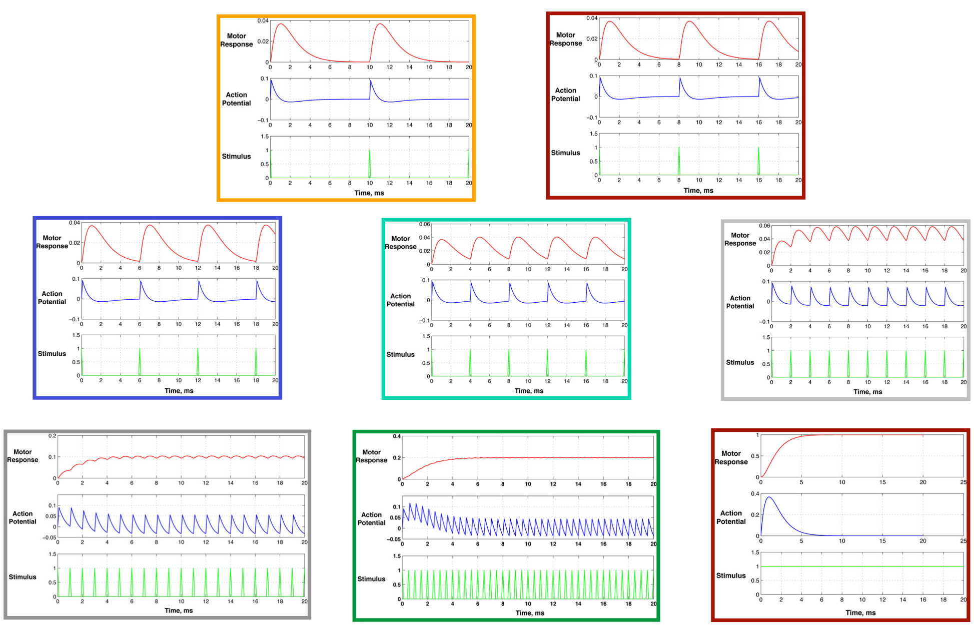
- *t* test identified strong features for classification in this data set
- Neural networks easily classified the four data types
- Caveat: Small, round blue-cell tumors occur in different tissue types
  - Ewing's sarcoma: Bone tissue
  - Burkitt's Lymphoma: Lymph nodes
  - Neuroblastoma: Nerve tissue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

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

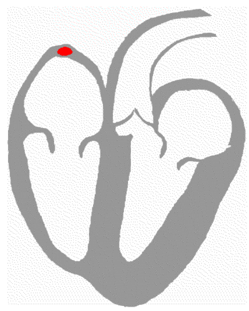
Next Time:
Neural Networks - 2

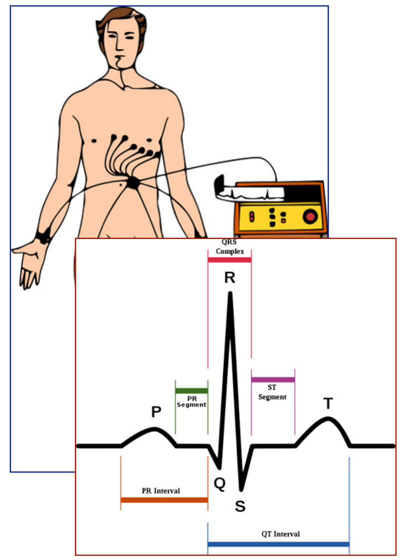
Supplementary Material

Impulse, Pulse-Train, and Step Response of a LTI 2nd-Order Neural Model

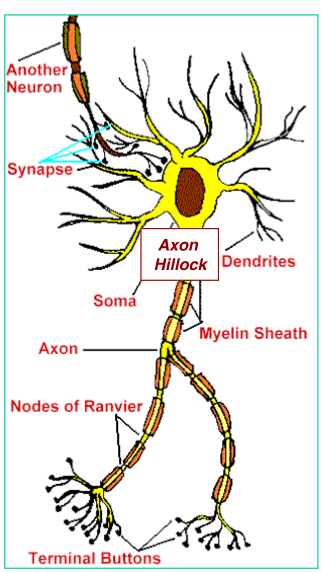


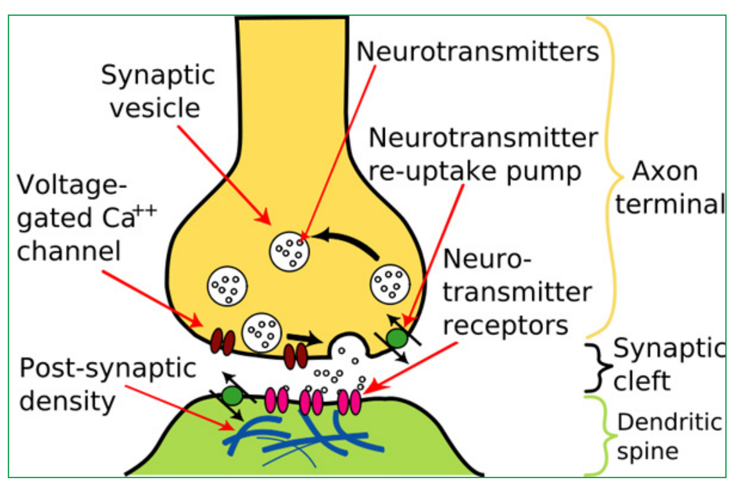
Cardiac Pacemaker and EKG Signals





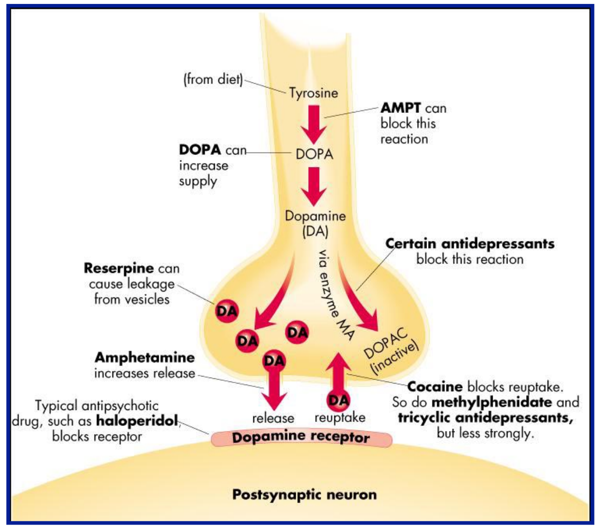
Electrochemical Signaling at Axon Hillock and Synapse



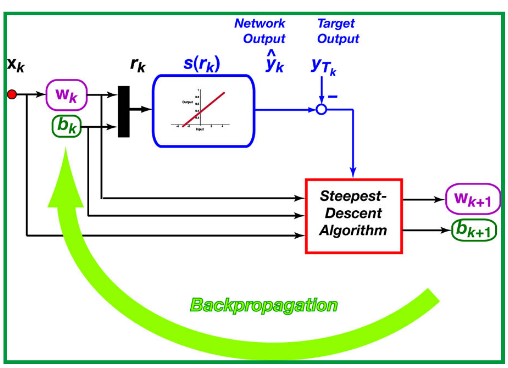


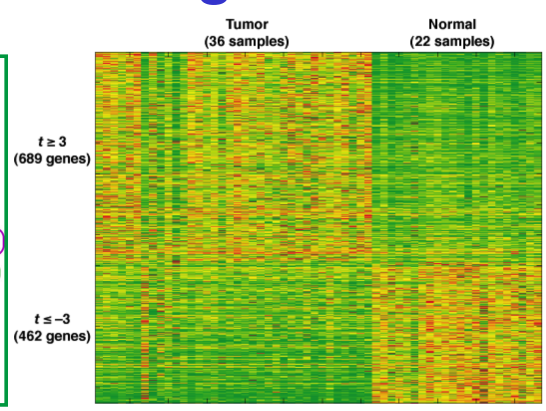
Synaptic Strength Can Be Increased or Decreased by Externalities

- **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



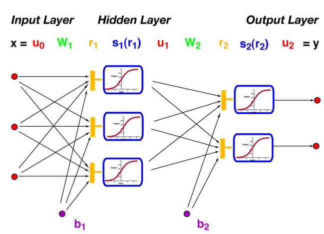


$$\begin{bmatrix} \textit{Identifier} \\ \mathbf{y}_T \\ \mathbf{X} \end{bmatrix} = \begin{bmatrix} \textit{Sample } 1 & \textit{Sample } 2 & \textit{Sample } 3 & \ldots & \textit{Sample } n-1 & \textit{Sample } n \\ \textit{Tumor} & \textit{Tumor} & \textit{Tumor} & \ldots & \textit{Normal} & \textit{Normal} \\ \textit{Gene A Level} & \textit{Gene A Level} & \textit{Gene A Level} & \ldots & \textit{Gene A Level} & \textit{Gene A Level} \\ \textit{Gene B Level} & \textit{Gene B Level} & \textit{Gene B Level} & \ldots & \textit{Gene B Level} & \textit{Gene B Level} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \textit{Gene m Level} & \textit{Gene m Level} & \textit{Gene m Level} & \ldots & \textit{Gene m Level} & \textit{Gene m Level} \end{bmatrix}$$

Microarray Training Data

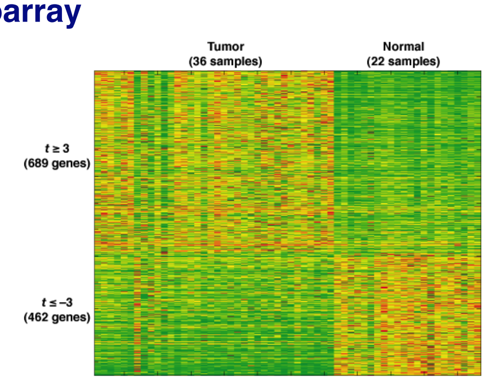
- **First row: Target classification**
- **2^{nd}-5^{th} rows: Up-regulated genes**
- **6^{th}-10^{th} rows: Down-regulated genes**

```
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   74   35  149   97   21  ...
                14  -51   -3  -81   57   -4   16   28  -73   -4   45  -28   -9  -13  ...
                25   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];
```



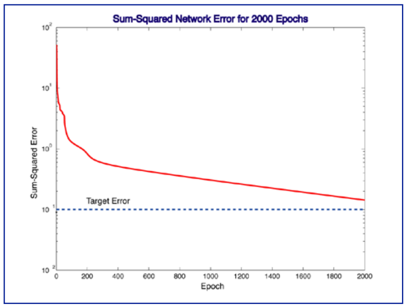
Neural Network Classification Example

- ~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

Binary network output (0,1) after rounding

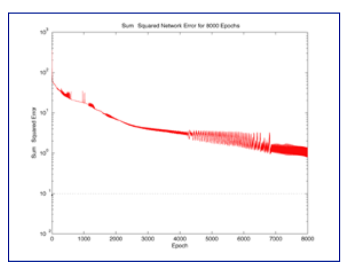
Zero classification errors

Classification =
Columns 1 through 13
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
Columns 27 through 39
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
Columns 53 through 62
0 0 0 0 0 0 0 0 0 0

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

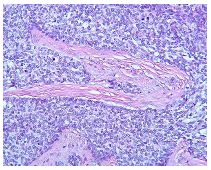
Scalar network output with varying magnitude



Target =
[2 1 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 1 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

Classification =
Columns 1 through 13
2 1 3 3 3 3 3 3 3 3 3 3 3
Columns 14 through 26
3 3 3 3 3 3 3 4 4 $\boxed{5}$ 4 4 4
Columns 27 through 39
4 4 4 5 5 5 5 5 5 5 5 1 0
Columns 40 through 52
0 0 0 0 0 0 0 0 0 0 0 0 0
Columns 53 through 60
0 0 0 0 0 0 0 0

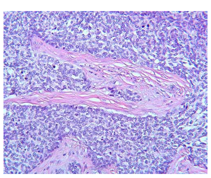


Ranking by EWS *t* Values (Top and Bottom 12)

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

Image ID	Sort by EWS t Value Transcript Description	EWS t Value	BL t Value	NB t Value	RMS t Value
770394	Fc fragment of IgG, receptor, transporter, alpha	12.04	-6.67	-6.17	-4.79
1435862	antigen identified by monoclonal antibodies 12E7, F21 and O13	9.09	-6.75	-5.01	-4.03
377461	caveolin 1, caveolae protein, 22kD	8.82	-5.97	-4.91	-4.78
814260	follicular lymphoma variant translocation 1	8.17	-4.31	-4.70	-5.48
491565	Cbp/p300-interacting transactivator, with Glu/Asp-rich carboxy-terminal domain	7.60	-5.82	-2.62	-3.68
841641	cyclin D1 (PRAD1: parathyroid adenomatosis 1)	6.84	-9.93	0.56	-4.30
1471841	ATPase, Na+/K+ transporting, alpha 1 polypeptide	6.65	-3.56	-2.72	-4.69
866702	protein tyrosine phosphatase, non-receptor type 13	6.54	-4.99	-4.07	-4.84
713922	glutathione S-transferase M1	6.17	-5.61	-5.16	-1.97
308497	KIAA0467 protein	5.99	-6.69	-6.63	-1.11
770868	NGFI-A binding protein 2 (ERG1 binding protein 2)	5.93	-6.74	-3.88	-1.21
345232	lymphotoxin alpha (TNF superfamily, member 1)	5.61	-8.05	-2.49	-1.19
786084	chromobox homolog 1 (Drosophila HP1 beta)	-5.04	-1.05	9.65	-0.62
796258	sarcoglycan, alpha (50kD dystrophin-associated glycoprotein)	-5.04	-3.31	-3.86	6.83
431397		-5.04	2.64	2.19	0.64
825411	N-acetylglucosamine receptor 1 (thyroid)	-5.06	-1.45	5.79	0.76
859359	quinone oxidoreductase homolog	-5.23	-7.27	0.78	5.40
75254	cysteine and glycine-rich protein 2 (LIM domain only, smooth muscle)	-5.30	-4.11	2.20	3.68
448386		-5.38	-0.42	3.76	0.14
68950	cyclin E1	-5.80	0.03	-1.58	5.10
774502	protein tyrosine phosphatase, non-receptor type 12	-5.80	-5.56	3.76	3.66
842820	inducible poly(A)-binding protein	-6.14	0.60	0.66	3.80
214572	ESTs	-6.39	-0.08	-0.22	4.56
295985	ESTs	-9.26	-0.13	3.24	2.95

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



Comparison of Present SRBCT Set with Khan Top 10

Image ID	Gene Description	EWS Student t Value	BL Student t Value	NB Student t Value	RMS Student t Value	Most Significant t Value	Khan Gene Class
296448	insulin-like growth factor 2 (somatomedin A)	-4.789	-5.226	-1.185	5.998	RMS	RMS
207274	Human DNA for insulin-like growth factor II (IGF-2); exon 7 and additional ORF	-4.377	-5.424	-1.639	5.708	RMS	RMS
841641	cyclin D1 (PRAD1: parathyroid adenomatosis 1)	6.841	-9.932	0.565	-4.300	BL (-)	EWS/NB
365826	growth arrest-specific 1	3.551	-8.438	-6.995	1.583	BL (-)	EWS/RMS
486787	calponin 3, acidic	-4.335	-6.354	2.446	2.605	BL (-)	RMS/NB
770394	Fc fragment of IgG, receptor, transporter, alpha	12.037	-6.673	-6.173	-4.792	EWS	EWS
244618	ESTs	-4.174	-4.822	-3.484	5.986	RMS	RMS
233721	insulin-like growth factor binding protein 2 (36kD)	0.058	-7.487	-1.599	2.184	BL (-)	Not BL
43733	glycogenin 2	4.715	-4.576	-3.834	-3.524	EWS	EWS
295985	ESTs	-9.260	-0.133	3.237	2.948	EWS (-)	Not EWS

- Red: both sets
- Black: Khan set only

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

```
'Leave-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)

```
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)

```
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;ar
```

Check calling sequence of *newff*

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

```
%           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);-
```

MATLAB Program for Neural Network Analysis with Leave-One-Out Validation - Training(3)

```
%       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;
```

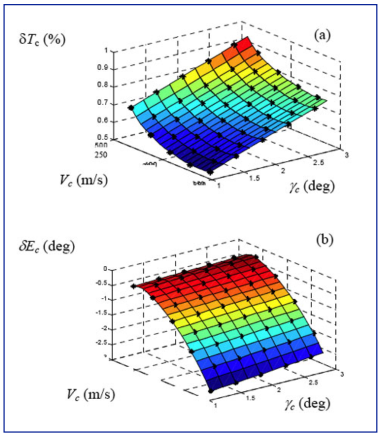
MATLAB Program for Neural Network Analysis with Leave-One-Out Validation - Training(4)

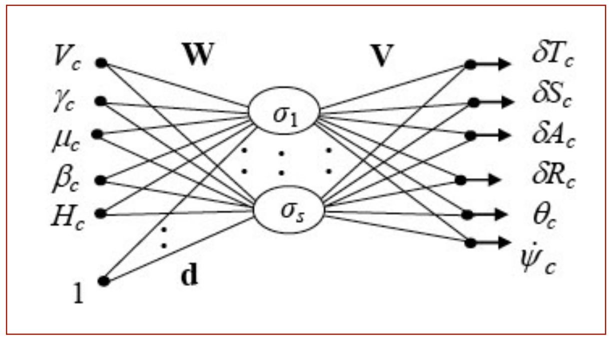
```
        if (~isequal(NovelError,zeros(LengthNO,1)))
            MisClass    =   MisClass + 1;
            iSamLog     =   [iSamLog iSam];
            iRepLog     =   [iRepLog i];
            ErrorLog    =   [ErrorLog NovelError];
            OutputLog   =   [OutputLog NovelOutput];
        end
 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

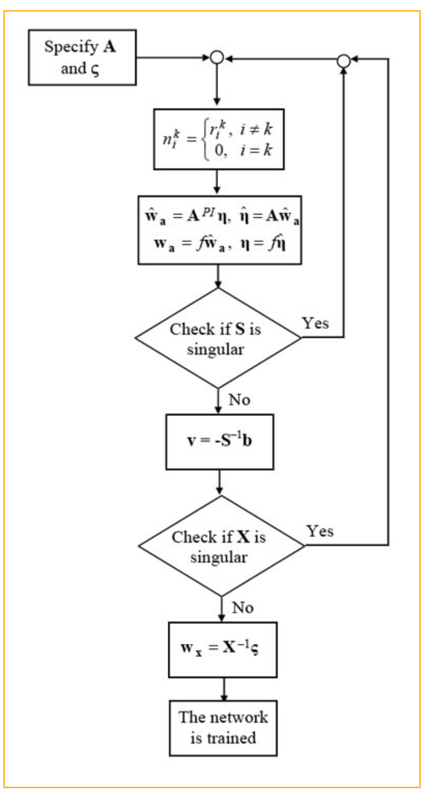
- Smooth functions define equilibrium control settings at many operating points
- Neural network required to fit the functions

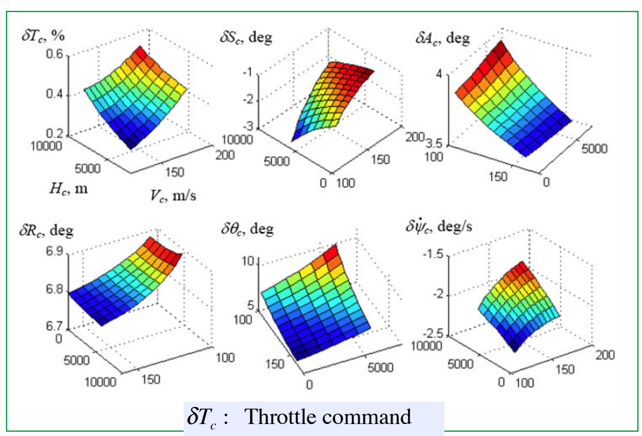




Ferrari and Stengel

Algorithm for Network Training





δT_c : Throttle command
δS_c : Spoiler command
δA_c : Aileron command
δR_c : Rudder command
$\delta \theta_c$: Pitch angle command
$\delta \psi_c$: Yaw rate command

Results for Network Training

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

Algorithm:	Time (Scaled):	Flops:	Lines of code (MATLAB®):	Epochs:	Final error:
Algebraic	1	2×10^5	8	1	0
Levenberg-Marquardt	50	5×10^7	150	6	10^{-26}
Resilient Backprop.	150	1×10^7	100	150	0.006