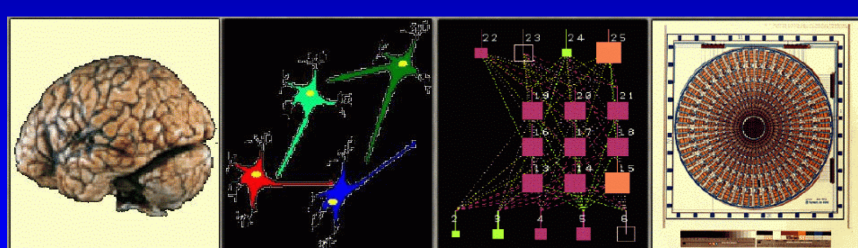
Introduction
COMP / ELEC / STAT 502

http://www.ece.rice.edu/~erzsebet/ANNcourse.html

What is an ANN?

For the neurobiologist:
mathematical model of the nervous system

For the computational scientist:
biologically inspired computing architecture



To mimic the smarts of brain-like information processing

An ANN is …

- Massively parallel

  A *very large number* of (usually) identical, densely interconnected processing units work parallel in time on one task.

- Very finely distributed

  The units perform *very low level (usually identical) function*. ANN's can be viewed as "ultimate RISC machines"; ultimate level in Parallel Distributed Processing, finest granularity.

Current computer clusters:	ANNs:
• *High level function of CPUs*	• *Low level function of units*
• *Low connectivity*	• *Dense connectivity*

- Learning machine

  Distribution of a task among the processing units is achieved by adaptation (learning), not by scheduling (programming).

  Learning from data (examples, experience); acquire knowledge, store, & apply to solve new problems.

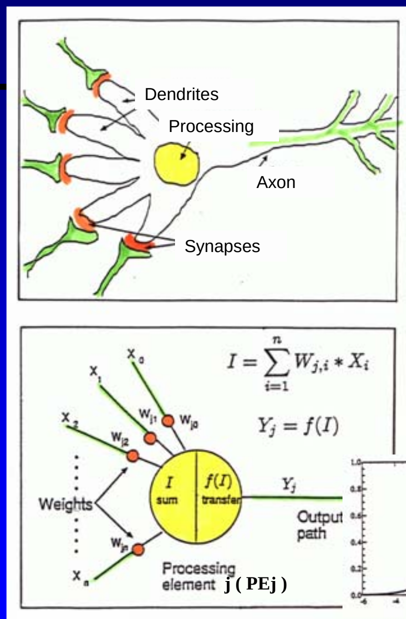
The subject of ANN research

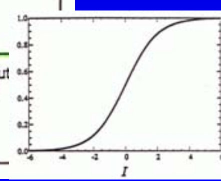
How does coordination / collaboration evolve "on its own" among locally, individually acting units, to solve tasks?

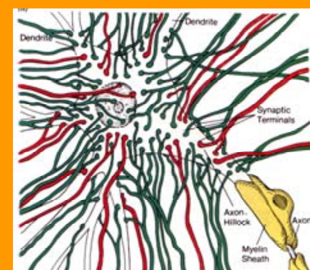
How can we control what kind of coordination / collaboration evolves?

What inspiration can be found from biology about PDP?	What can the computational side tell about biology?

The building block of ANNs
Artificial neuron model



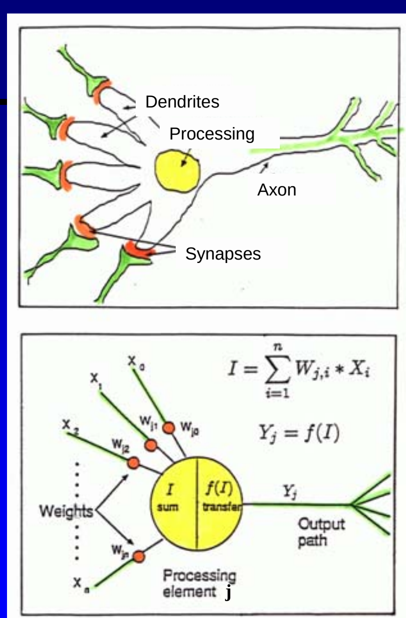




(From Wessels & Hopson: Biology)

More at "Neuroscience for Kids", under "Additional Links of Interest" at the course web site.

The building block of ANNs
Artificial neuron model



Processing cycle in nervous systems

(= receiving data through dendrites & reacting by adapting the synaptic strength and firing, then recovering)

~ 20 – 50 msec SLOW

Connectivity in real nervous systems

$10^3 – 10^4$ synapses per neuron;

~ 10^9 neurons in the human brain;

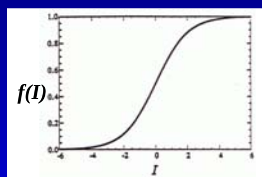
With 10^2 cycles / sec,

~ 10^{15} - 10^{16} ops/sec a LOT

Real intelligence is thought to be related to this kind of computational bandwidth. Computers are not yet there.

Consider: PE with f(I) = I is a correlator: max response is for w = x (the stored weight = memory, recognizes x). Each PE can recognize a different pattern ⇒combined effect is powerful

Transfer functions in Processing Elements



Desirable properties:

- Monoton non-decreasing
- Non-linear
- Bounded (asymptotically)
- Largest change is at intermediate range. Least change is at extremes.
- Derivative is easy to compute

Examples from MATLAB NN Toolbox

HW: Study and know transfer functions

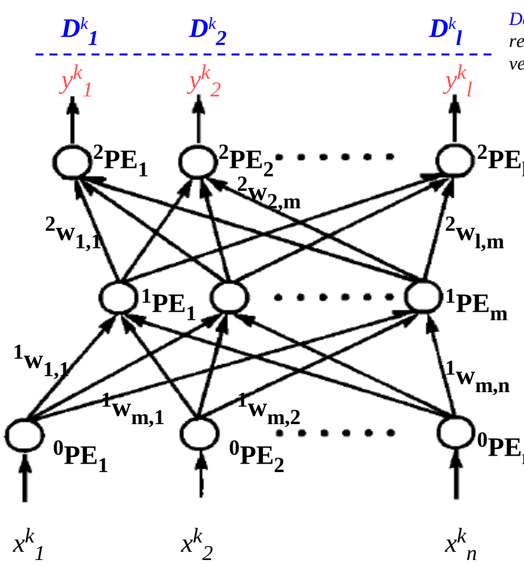
dhardlim	Hard limit function
Dhardlms	Symmetric hard limit (Heaviside)
Dlogsig	Log sigmoid (logistic, in figure) → $f(x) = \frac{1}{1+e^{-ax}}$
Dposlin	Positive linear
dpurelin	Linear
dradbas	Radial basis
dtansig	Hyperbolic tangent sigmoid → $f(x) = a\frac{e^{bx}-e^{-bx}}{e^{bx}+e^{-bx}}$

Schematics of an ANN

(a typical supervised example)

2-layer ANN

Many iterations …



Desired outputs: Target responses for k^{th} *input vector, from teacher*

Output vector: actual values computed by ANN

Learning: Adaptation of weights at every time step (*k*), based on (*y-D*) according to a *learning rule*, such that the output gets closer to the target. The learned ANN can predict for unseen patterns. The knowledge is stored in the weights.

k^{th} *input vector (training pattern)*

Schematics of an ANN, cont'd

Learning:

• *Adaptation of the weights* at every time step (k), in response to a training pattern $\boldsymbol{x}^k$, *according to a learning rule*, such that the output $\boldsymbol{y}^k$ gets closer to the target $\boldsymbol{D}^k$. Training is finished when all training patterns are recognized satisfactorily. The knowledge is stored in the weights.

• The goal is to learn to *generalize*, i.e., for the ANN to be able to predict / recognize unseen patterns (patterns that were not part of the training set). Generalization performance is measured on *test patterns*.

Major types of machine learning:

Supervised Unsupervised Reinforcement

The learning rule of an ANN

$$F : \boldsymbol{W}^{old} \longrightarrow \boldsymbol{W}^{new} \qquad \boldsymbol{W}^{old} = \boldsymbol{W}(t), \quad \boldsymbol{W}^{new} = \boldsymbol{W}(t+1), \quad t\text{: time step}$$

$$F(\boldsymbol{W}^{old}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{D}, \{\text{learning parameters}\}) = \boldsymbol{W}^{new}$$

Learning is data driven, the final 'brain' is determined by the training data

$\boldsymbol{W}$: *weight matrix*
$\boldsymbol{x}$: *input vector at t*
$\boldsymbol{y}$: *output vector at t*
$\boldsymbol{D}$: *target vector*

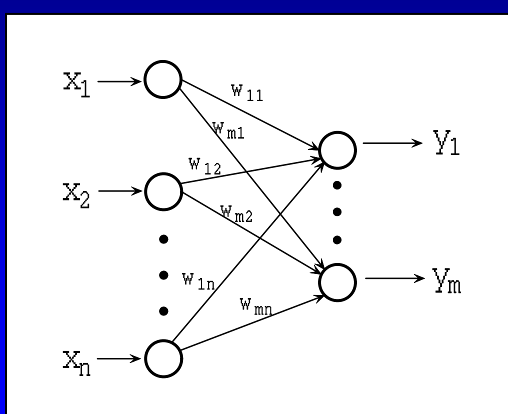
At every time step t, a different training pattern $\boldsymbol{x} = \boldsymbol{x}^k$ is chosen randomly, for input. $k=1, \ldots, P$, P is the number of training patterns.

The learning rule of a given ANN paradigm may not have all of the above arguments. (For example, $\boldsymbol{D}$ is not used for unsupervised learning.)

ANN paradigms are defined by

- connection topology and data flow direction
  - fully or partially connected
  - feed forward or feed back (e.g., recurrent nets)
  - numerous combinations
- transfer function
  - non-linear transfer function makes ANNs powerful
- learning rule

Example: Simple Perceptron or ?



Topology:
- Fully connected, 1-layer
- Feed forward

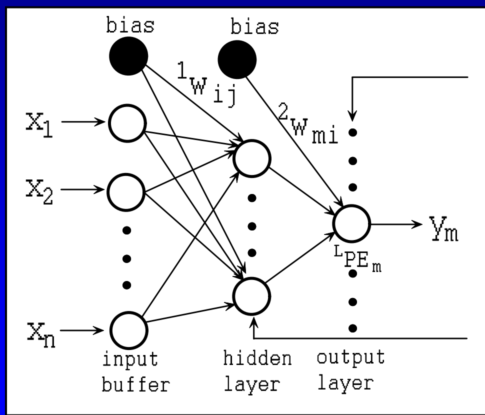
This could be a
- Simple perceptron
- Assocative memory
- Principal Components network

Depending on the choice of

Transfer function:
- {Sigmoid | hyperbolic tangent | linear | linear}

Learning rule:
- {Delta | Hebb/supervised | Hebb/unsupervised}

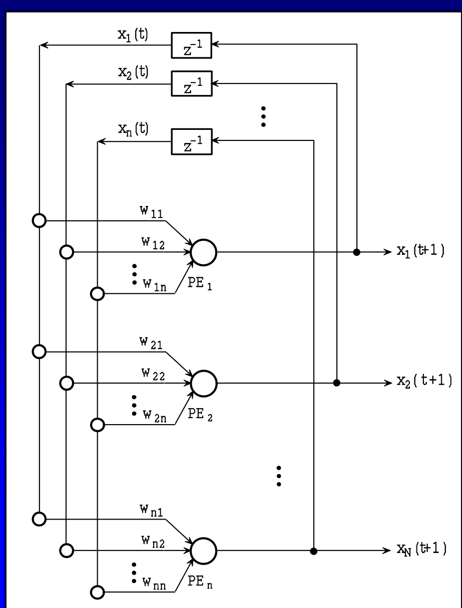
Example of Multilayer Perceptron (Back Propagation Network)



Topology:
- Fully connected
- Feed forward, 2-layer (2 layers of weights !!)

Transfer function:
- Sigmoid, or hyperbolic tangent

Learning rule:
- Error Back Propagation



Example: Recurrent Network (Hopfield Net)

Topology:
- Feed back, no layers

Transfer function:
- Identity

Learning rule:
- Hopfield

Some standard ANN paradigms

(often found in packages)

- Adaline and Madaline (old, historical)
- Adaptive Resonance Theory (ARTxx)
- Associative Memory
- Back-Propagation (BP)
- Bi-Directional Associative Memory (BAM)
- Boltzmann Machine
- Cascade Correlation Network
- Hopfield Network
- Learning Vector Quantization (LVQxx)
- Neocognitron
- Perceptron (simple)
- Probabilistic Neural Network (PNN)
- Radial Basis Function Network (RBF net)
- Recirculation Network
- Self-Organizing Map (SOM)

Types of tasks ANN's are good for

ANNs perform well on types of tasks on which humans traditionally perform well and serial computers do not.

These are tasks which are easier to solve by learning from examples than by defining rules, because

- the *rules are not (well) known*
- the *rules are too complicated* or impractical to formulate

Wide range of applications: pattern recognition, classification, clustering (structure discovery), data compression, feature extraction, optimization, image restoration, speech analysis, forcasting, time series analysis, real time process control, system identification, …

Many different ANN paradigms (both supervised and unsupervised) have been devised for different types of tasks.

ANN computation has sound mathematical foundations

some

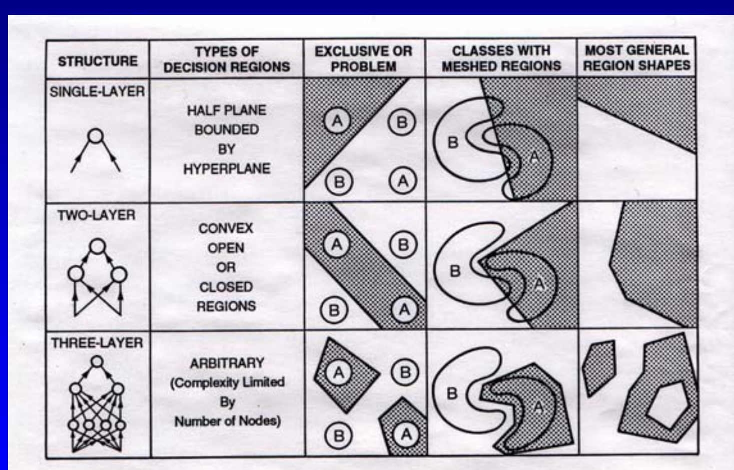
Theoretically proven fact:
A 3-layer ANN with non-linear transfer function is capable of learning any functional mapping.

Hecht-Nielsen, R. (1989) ``Theory of the Backpropagation Neural Networks", IJCN on Neural Networks, Vol. 1, Washington, DC, pp 593-605

Hornik, K., (1991) ``Approximation Capabilities of Multilayer Feedforward Networks", Neural Networks, Vol. 4, pp. 251-257

Kreinovich, V. Ya. (1991) ``Arbitrary Nonlinearity Is Sufficient to Represent All Functions by Neural Networks: A Theorem", Neural Networks, Vol. 4, pp. 381-383

Approximation capabilities of ANNs



Source: R. P. Lippman, "An Introduction to Computing with Neural Nets" IEEE ASSP Magazine, April 1987, p. 4

Major journals / publications

Neural Networks
Journal of the Int'l Neural Network Society; European NN Society; and Japanese NN Society

IEEE Transactions on Neural Networks
Journal of the IEEE Computational Intelligence Society (formerly NN Society)

Neurocomputing (Elsevier)

Neural Processing Letters

Biological Cybernetics

Springer series "Lecture Notes in Computer Science" and "Lecture Notes in Artificial Intelligence"

Many ANN-related articles are submitted to Computational Intelligence (CI) AI, and Soft Computing type journals and conferences.

Some tools

MATLAB NN Toolbox

NeuralWorks Professional Plus, by NeuralWare, expensive, proprietary (in my research environment)

Free software

- Stuttgart Neural Network Simulator (SNNS) -- at your own risk.
- LVQ PAK, SOM PAK
- SOM Toolbox for MATLAB
- FastICA Package for MATLAB
- more

Download these from "Free Software" link, under "Additional links of interest" at the course web site. Scroll down to Freeware and Shareware.

For this course, you will have to code most NNs that we use.