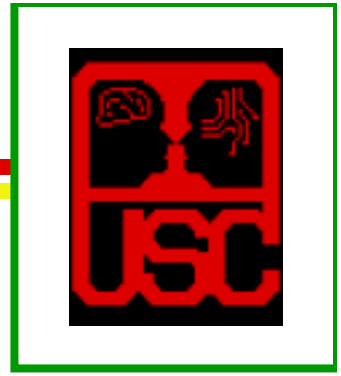


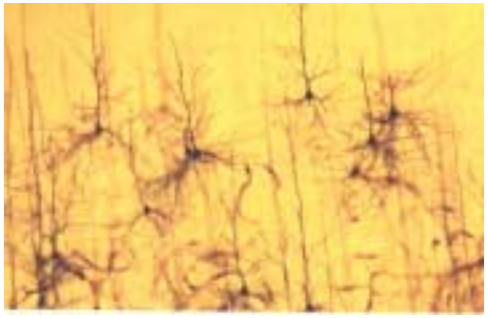
Lecture 17. Examples and Review

Reading Assignments:

None

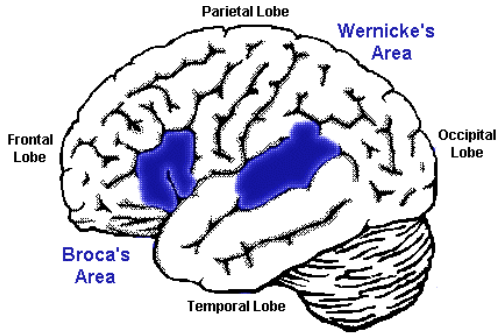
Syllabus Overview

Introduction

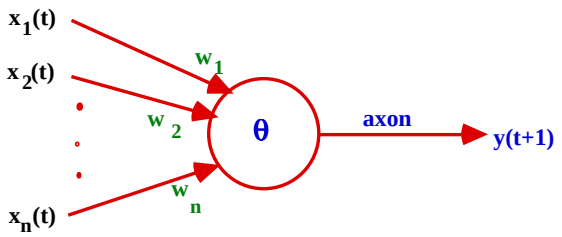


Overview

Charting the brain



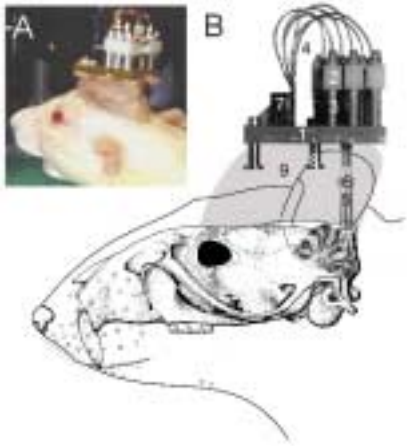
The Brain as a Network of Neurons



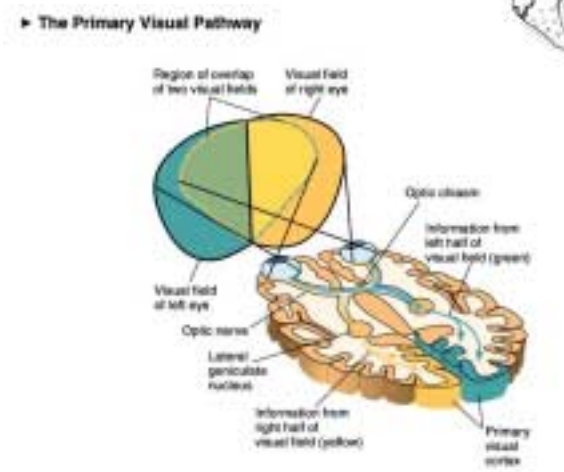
Syllabus Overview

Introduction (cont.)

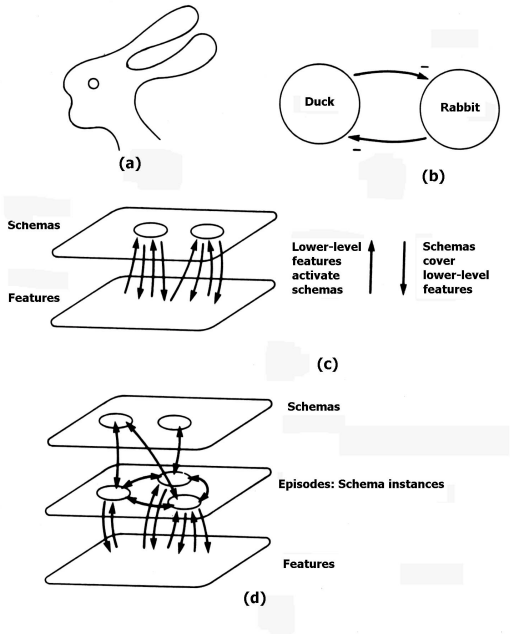
Experimental techniques



Introduction to Vision



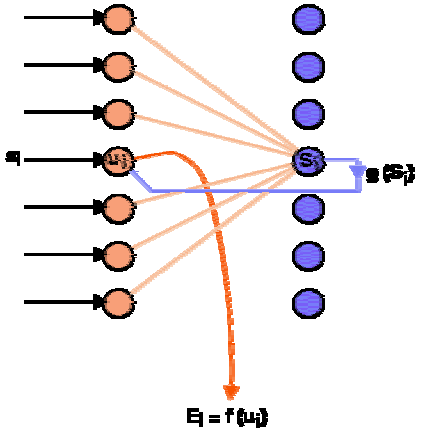
Schemas



Syllabus Overview

Basic Neural Modeling & Adaptive Networks

Didday Model of Winner-Take-All

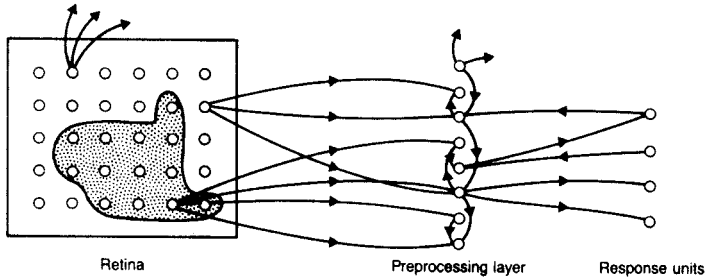


Hopfield networks

$$E = -\tfrac{1}{2} \sum_{ij} s_i s_j w_{ij} + \sum_i s_i \theta_i$$

Adaptive networks: Hebbian learning;

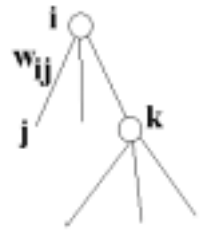
Perceptrons; landmark learning



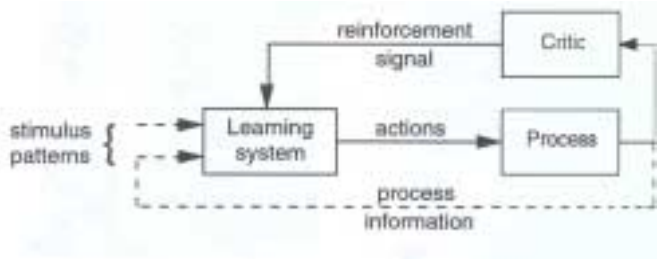
Syllabus Overview

Neural Modeling & Adaptive Networks (cont.)

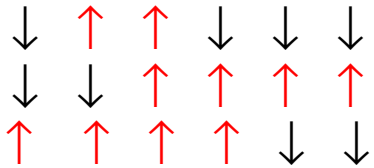
Adaptive networks: gradient descent and backpropagation



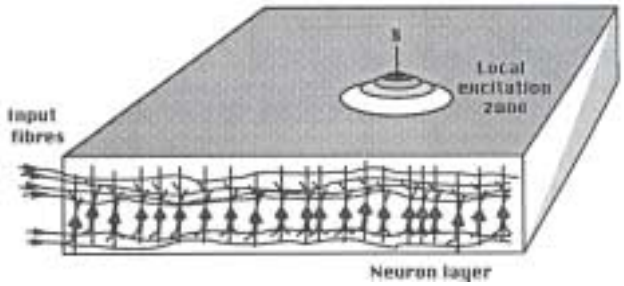
Reinforcement learning



Competition and cooperation



Visual plasticity; self-organizing feature maps; Kohonen maps

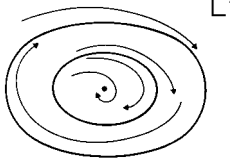


Syllabus Overview

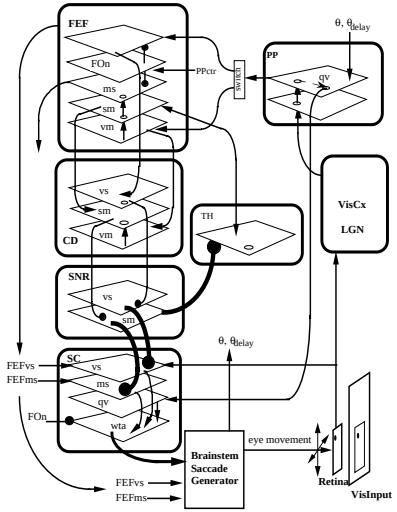
Examples of Large-scale Neural Modeling

System concepts

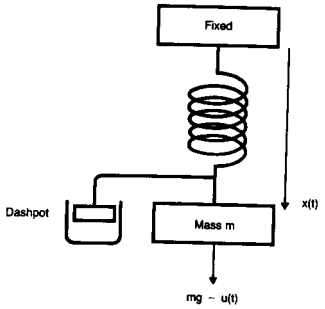
$$\dot{q}(t) = \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u(t)$$



Model of saccadic eye movements



Feedback and the spinal cord; mass-spring model of muscle



Neuron Models

We have studied the following types of neurons:

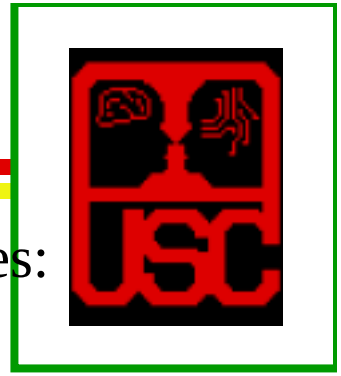
Biological: very complex, activity depends on many factors (including presynaptic activity, topography of dendritic tree, ion channel densities, concentrations of neurotransmitters and other ions, etc). Not fully understood.

McCulloch & Pitts: binary output as thresholded weighted sum of inputs. Highly non-linear model.

Continuous extension (used in Hopfield & Backprop networks): continuous output as sigmoid'ed weighted sum of inputs.

Leaky integrator: adds explicit time evolution (RC circuit behavior, plus possible threshold and spiking mechanism).

Network Architectures



We have studied these major categories of network architectures:

Layered, feedforward networks with synchronous update and no loops

Hopfield networks with asynchronous update and symmetric weights

Self-organizing feature maps in which some local connectivity pattern yields interesting emergent global behavior

Arbitrary biologically-inspired networks with loops, e.g., the winner-take-all

System Architectures

We have started looking at system architecture issues:

The NSL simulation environment and modular, hierarchical development of complex neural models

Discussion of black-box vs. fully-engineered approaches

Notion of schemas as intermediaries between neural patterns of activity and mental events

The Dominey-Arbib model of saccadic eye movements

… and we will focus on studying more examples of complex, biologically-inspired models in the second part of the course.

Learning and Adaptation

Finally, we have studied the following adaptation schemes:

Hebbian learning (strenghtening by co-activation) and Pavlovian conditioning

Perceptron learning rule (strengthening based on comparison between actual output and desired output)

Backpropagation (to extend the perceptron learning rule to hidden units subject to the credit assignment problem)

Reinforcement learning (or learning through monitoring one's own successes and failures, through a critic that may itself be adaptive)

Some Current Trends

In basic computational neuroscience, much current work goes into understanding the basic biophysics of computation. This typically involves much more detailed models and heavy simulations.

Issues of interest include:

- The computational role of specific dendritic tree structures
- Spike timing and synchronization
- Neuromodulation
- Coupling and properties of small recurrent networks
- Information-theoretic analysis of neurons and synapses
- Biochemical bases of learning
- … and many more.

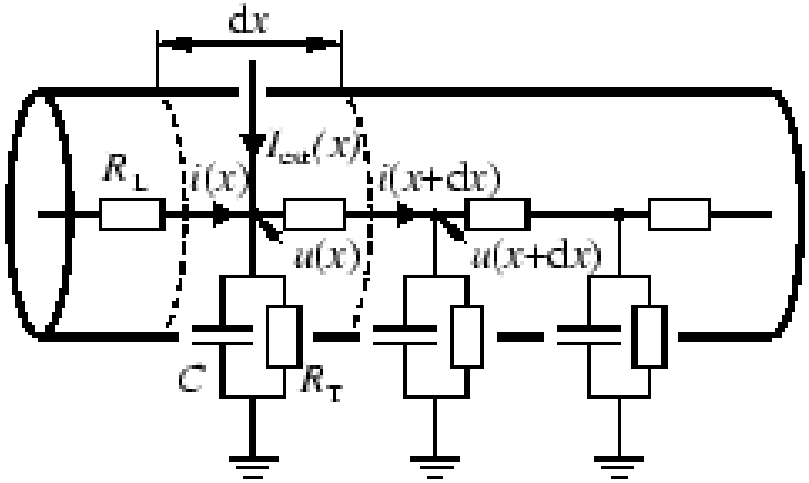
The Cable Equation

See

http://diwww.epfl.ch/~gerstner/SPNM/SPNM.html

For excellent additional material (some reproduced here).

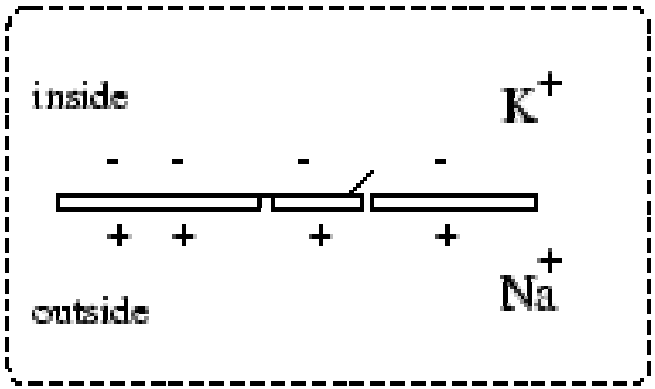
Just a piece of passive dendrite can yield complicated differential equations which have been extensively studied by electronicians in the context of the study of coaxial cables (TV antenna cable):

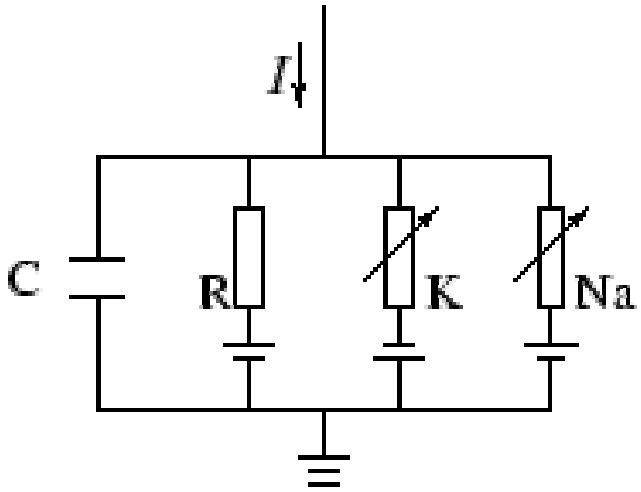


The Hodgkin-Huxley Model

Adding active ion channels yields a fairly realistic description of axons, dendrites and neurons.

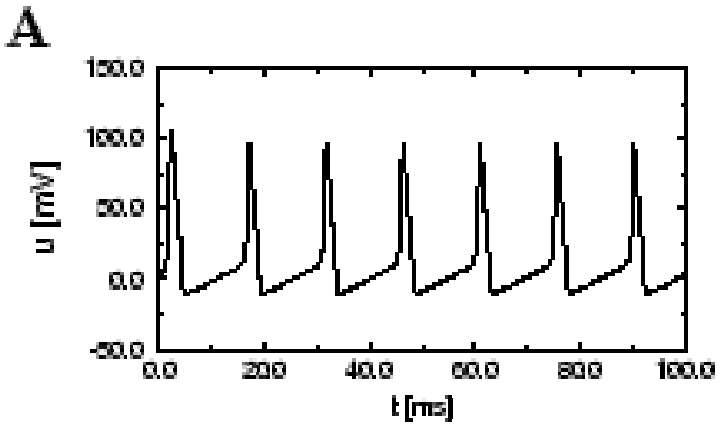
The Hodgkin-Huxley is an example of such fairly detailed model. It is an extension of the leaky integrator model, adding active ion channels. It is described by a set of coupled non-linear first-order differential equations. Simulating these equations yields fairly realistic time-dependent simulations.

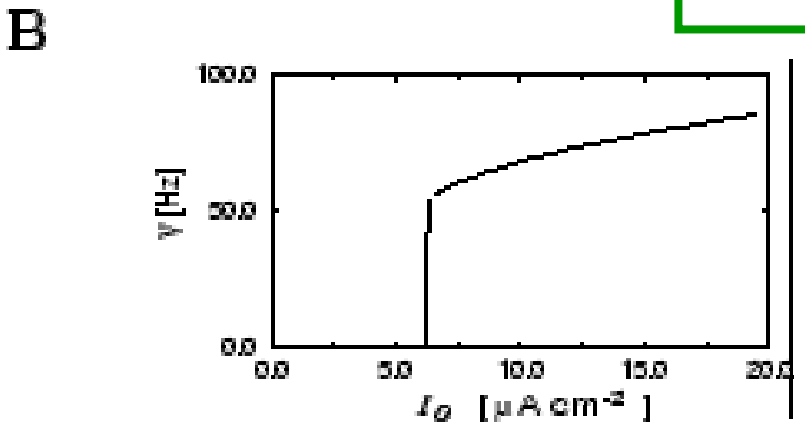


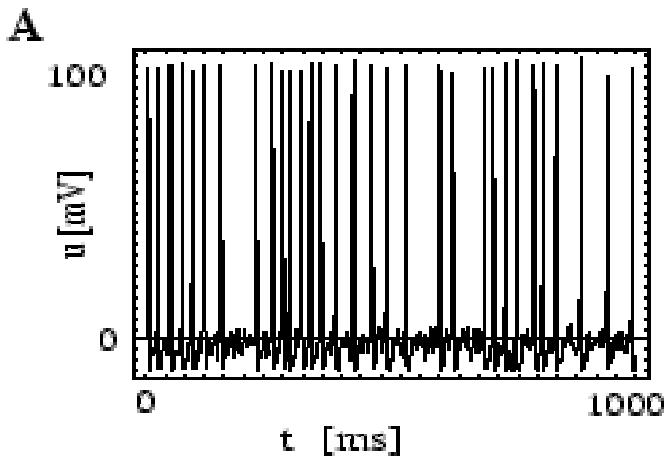


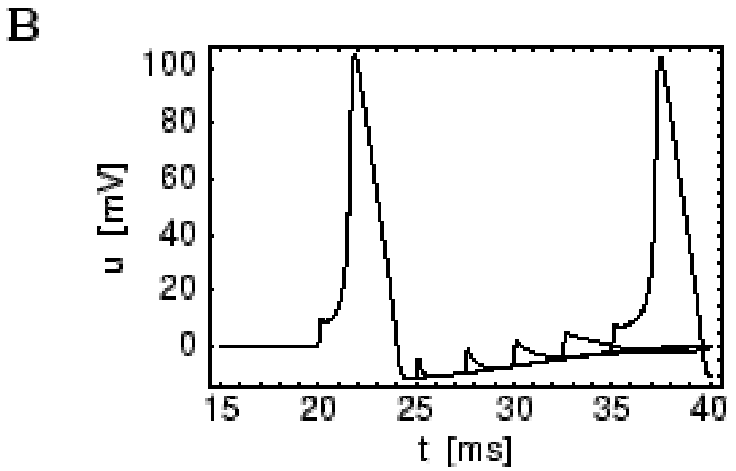
The Hodgkin-Huxley Model

Example spike trains obtained…





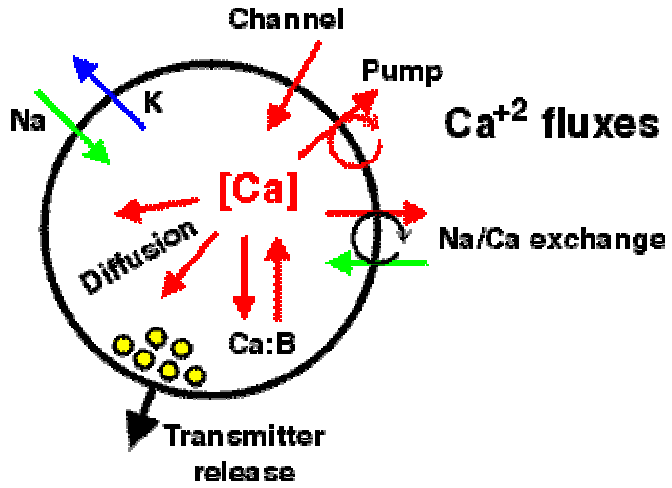


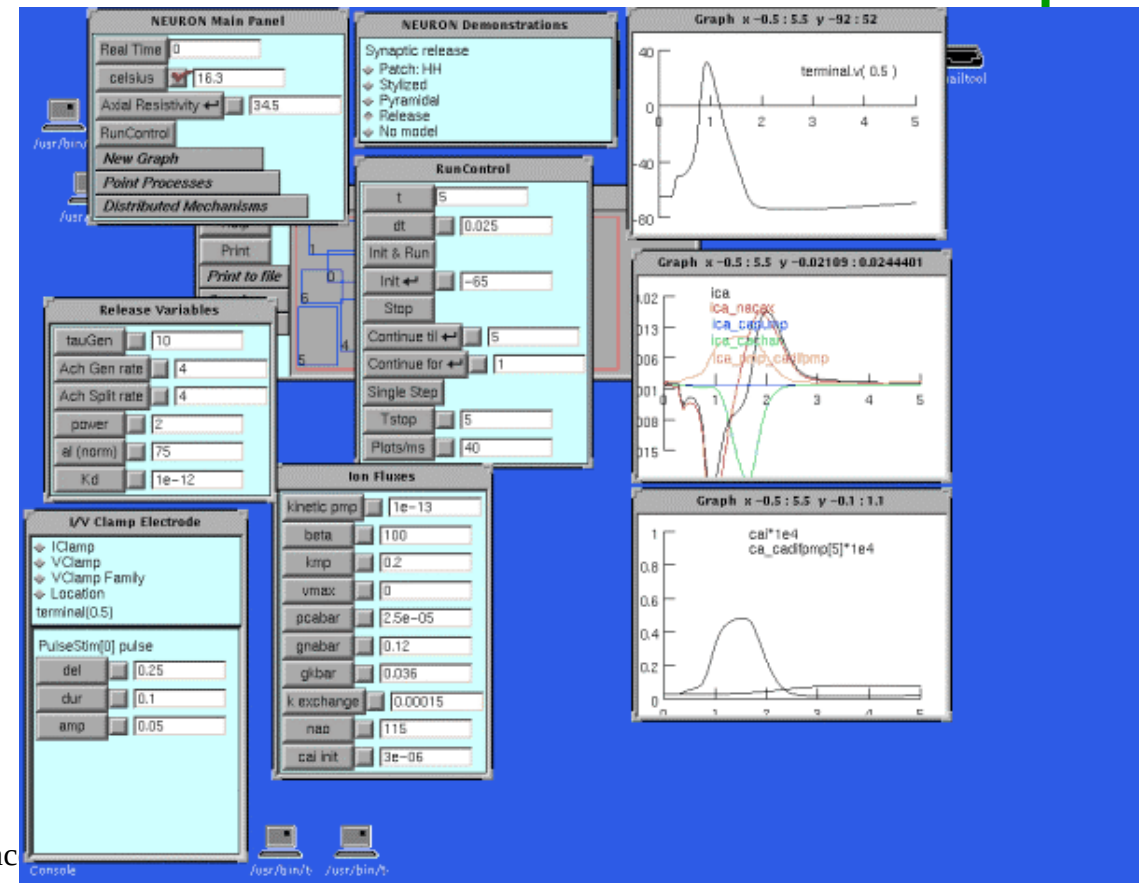


Detailed Neural Modeling

A simulator, called “Neuron” has been developed at Yale to simulate the Hodgkin-Huxley equations, as well as other membranes/channels/etc.

See http://www.neuron.yale.edu/

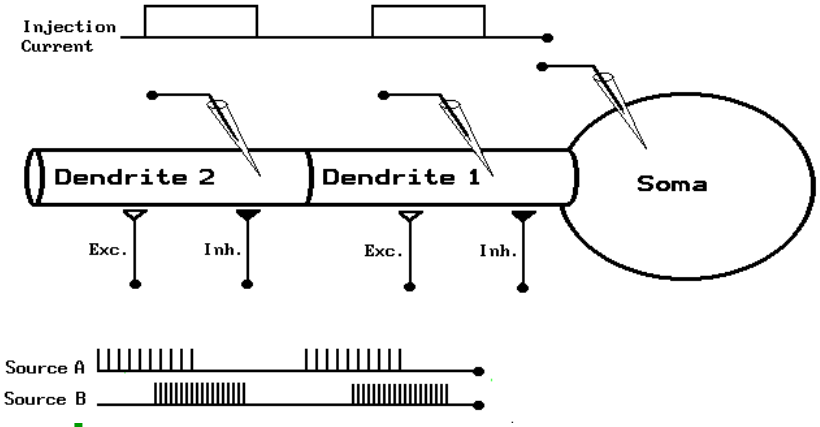


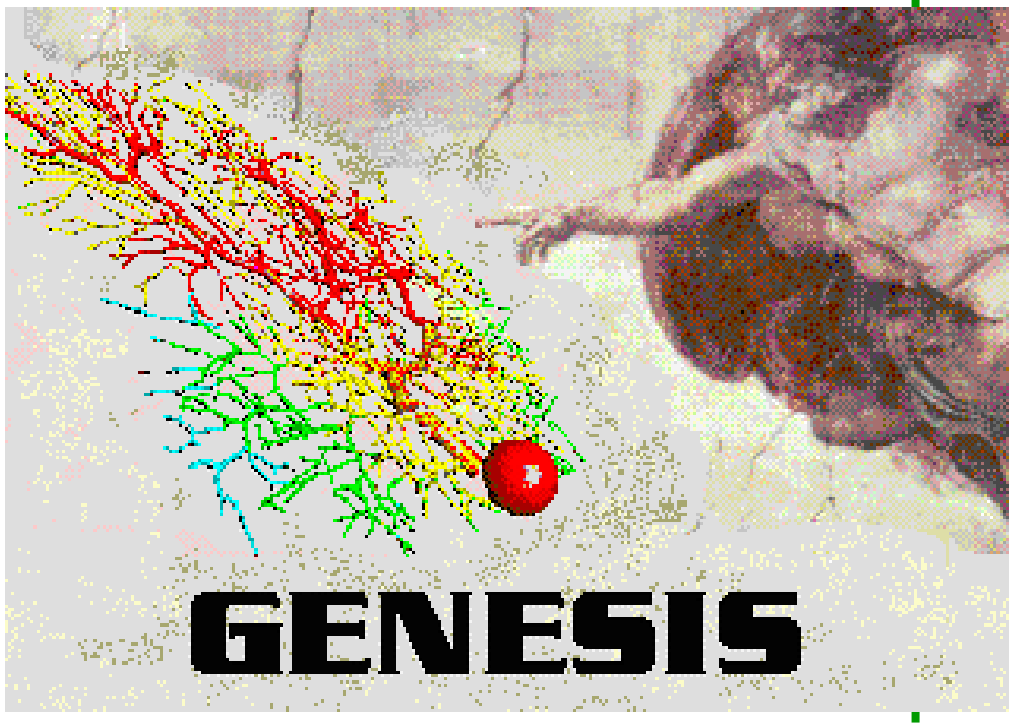


Detailed Neural Modeling

The Genesis model has been developed at Caltech to simulate large, complex dendritic structures, using compartmental modeling.

See http://www.genesis-sim.org/GENESIS/





Applications of Neural Networks

See http://www.neusciences.com/Technologies/nn_intro.htm

Applications: Classification

Business

- Credit rating and risk assessment
- Insurance risk evaluation
- Fraud detection
- Insider dealing detection
- Marketing analysis
- Mailshot profiling
- Signature verification
- Inventory control

Engineering

- Machinery defect diagnosis
- Signal processing
- Character recognition
- Process supervision
- Process fault analysis
- Speech recognition
- Machine vision
- Speech recognition
- Radar signal classification

Security

- Face recognition
- Speaker verification
- Fingerprint analysis

Medicine

- General diagnosis
- Detection of heart defects

Science

- Recognising genes
- Botanical classification
- Bacteria identification

Applications: Modelling

Business

- Prediction of share and commodity prices
- Prediction of economic indicators
- Insider dealing detection
- Marketing analysis
- Mailshot profiling
- Signature verification

Engineering

- Transducer linerisation
- Colour discrimination
- Robot control and navigation
- Process control
- Aircraft landing control
- Car active suspension control
- Printed Circuit auto routing
- Integrated circuit layout
- Image compression

Science

- Prediction of the performance of drugs from the molecular structure
- Weather prediction
- Sunspot prediction

Medicine

- . Medical imaging and image processing

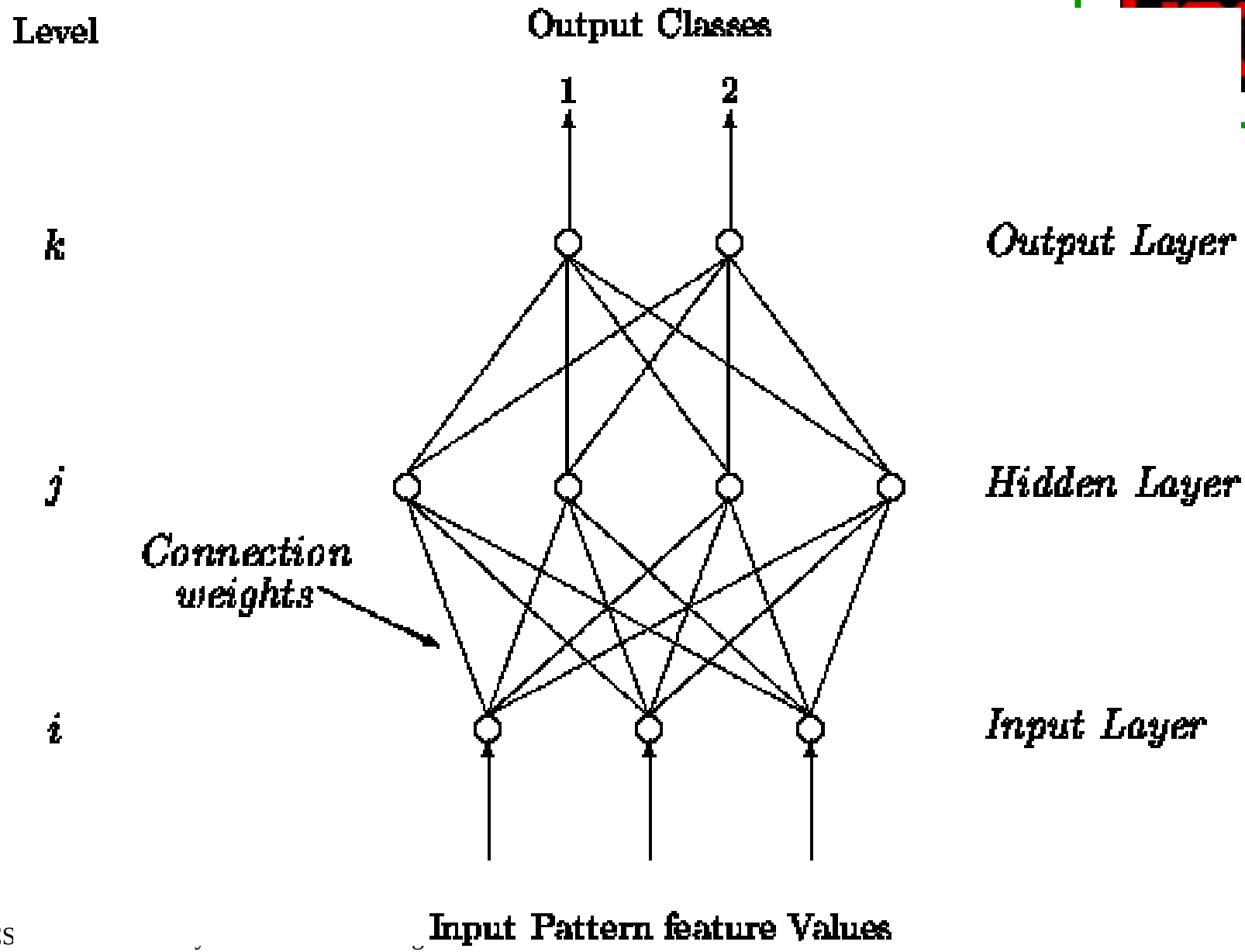
Applications: Forecasting

- Future sales
- Production Requirements
- Market Performance
- Economic Indicators
- Energy Requirements
- Time Based Variables

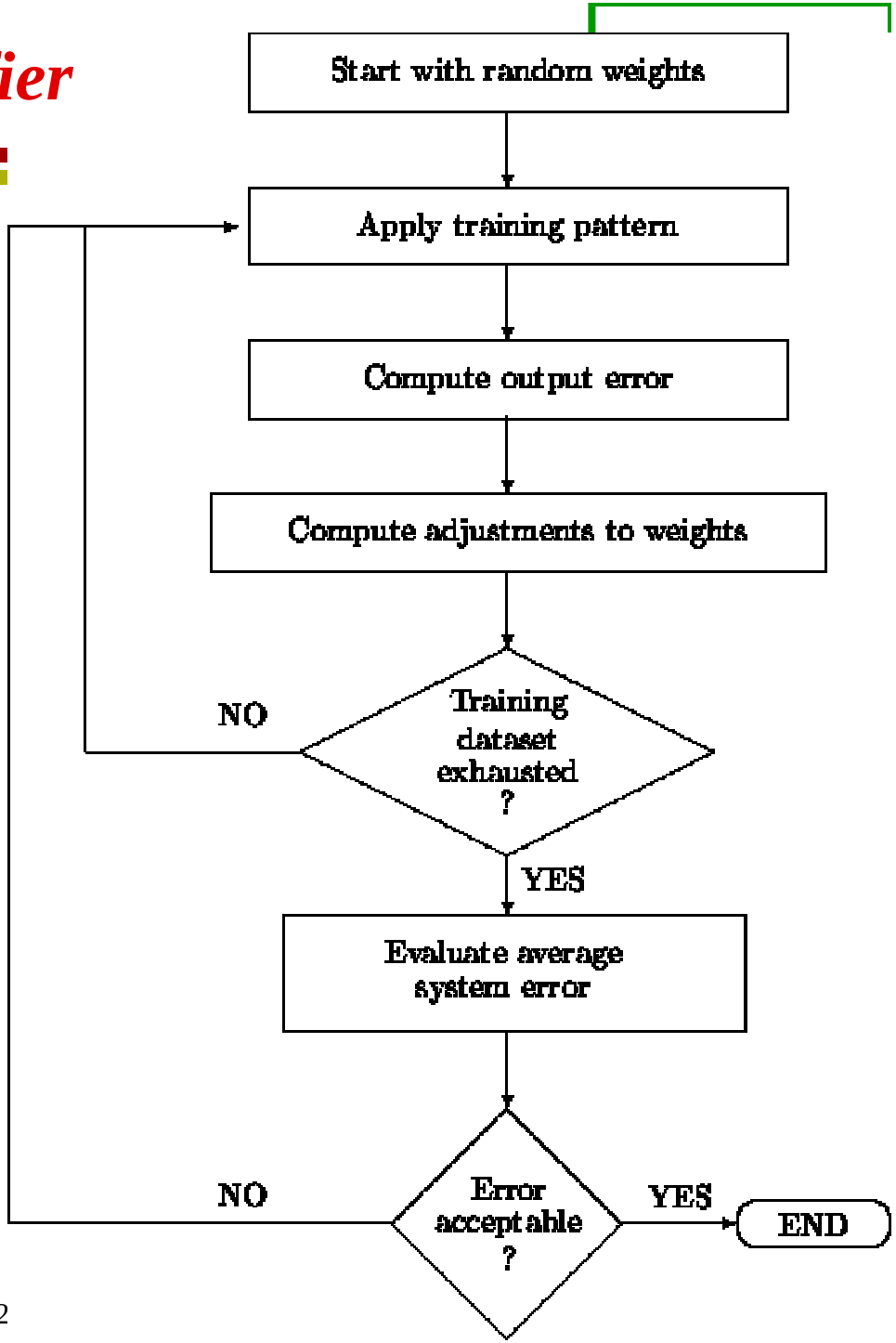
Applications: Novelty Detection

- Fault Monitoring
- Performance Monitoring
- Fraud Detection
- Detecting Rate Features
- Different Cases

Multi-layer Perceptron Classifier

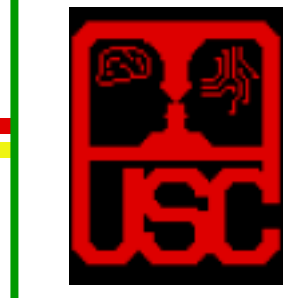


Multi-layer Perceptron Classifier



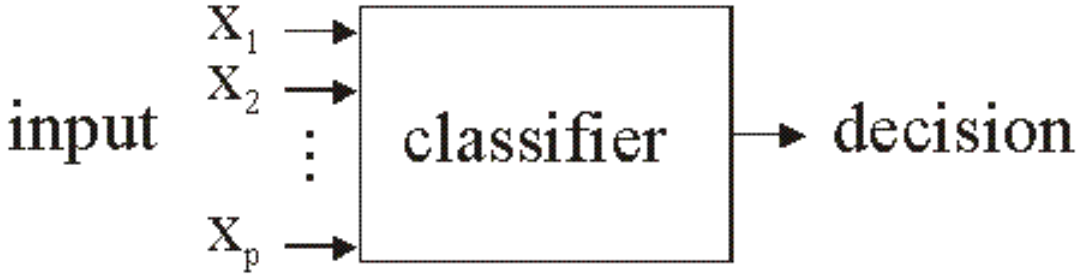
http://ams.egeo.sai.jrc.it/eurostat/Lot16-SUPCOM95/node7.html

Classifiers

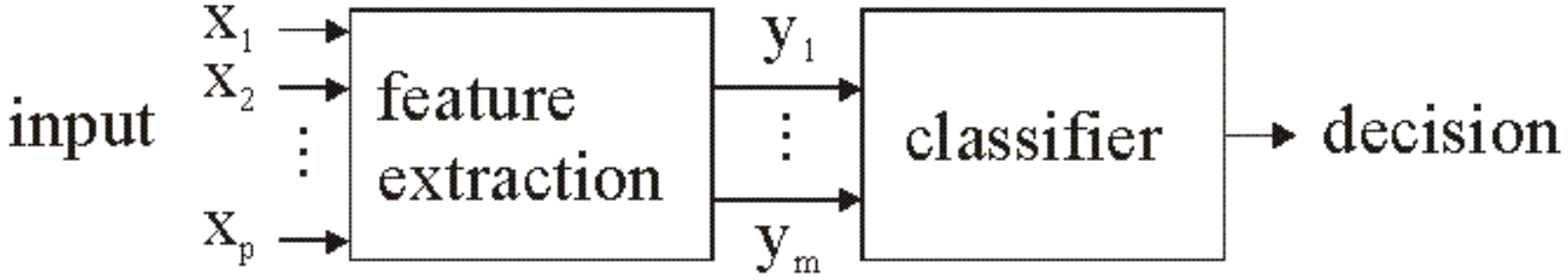


http://www.electronicsletters.com/papers/2001/0020/paper.asp

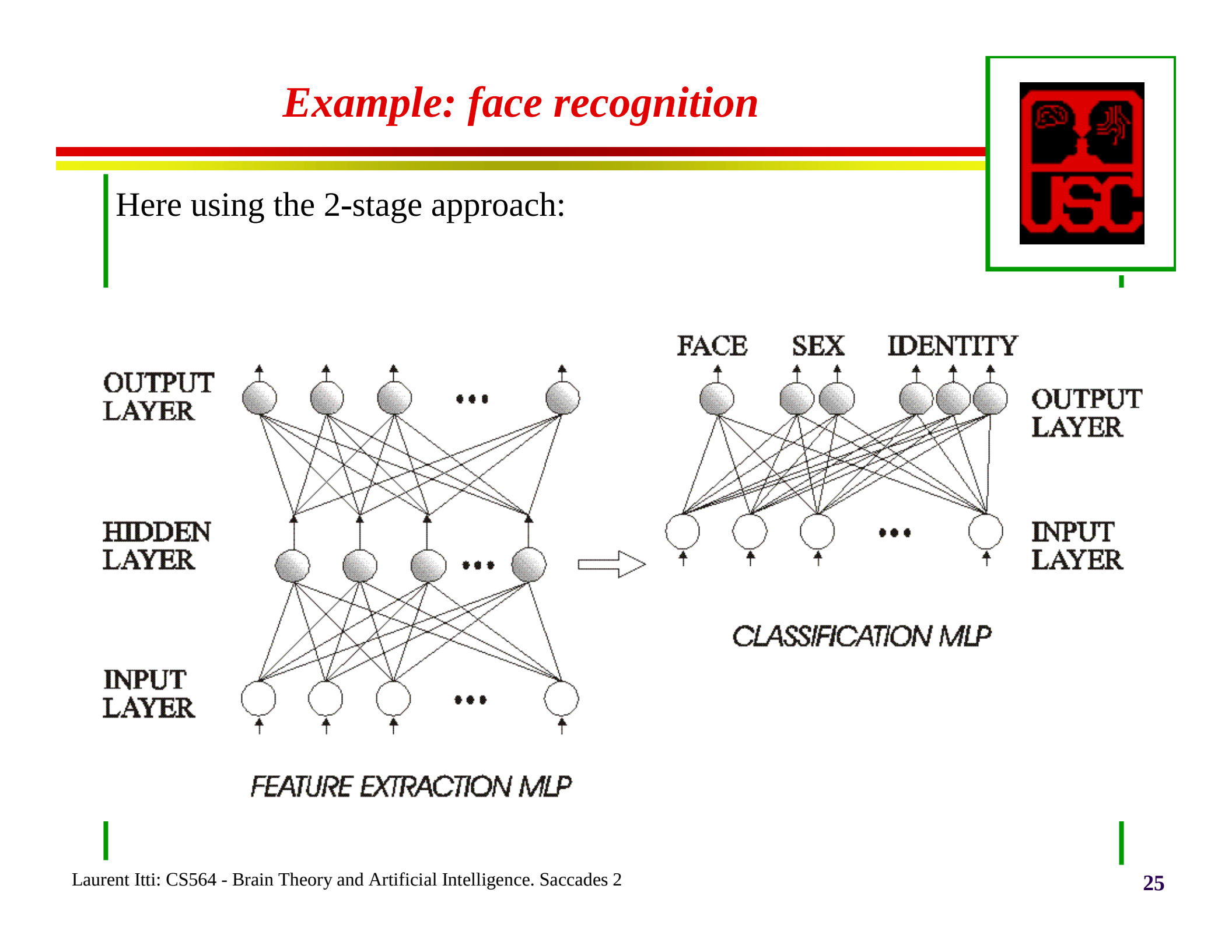
1-stage approach



2-stage approach



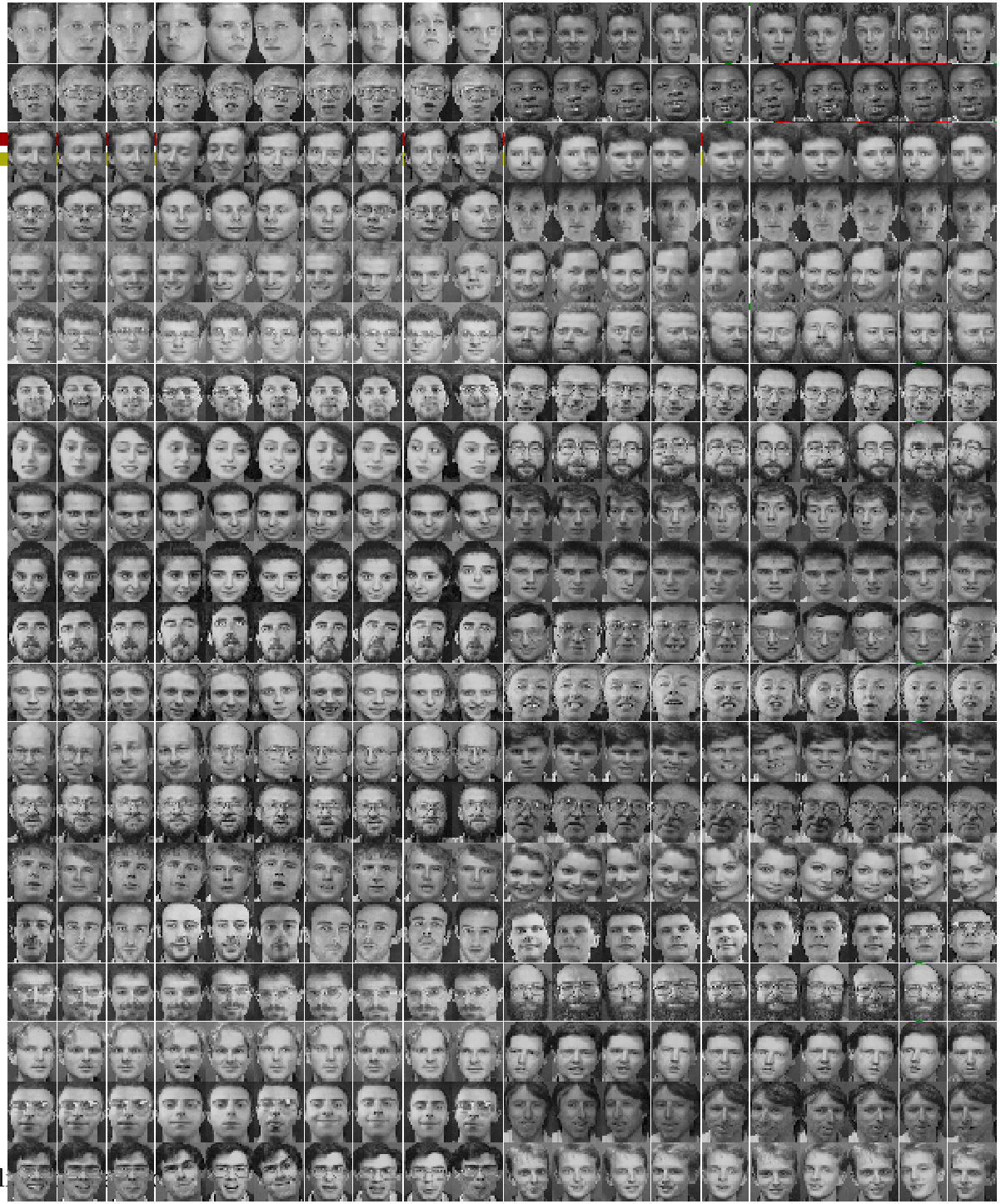
Example: face recognition



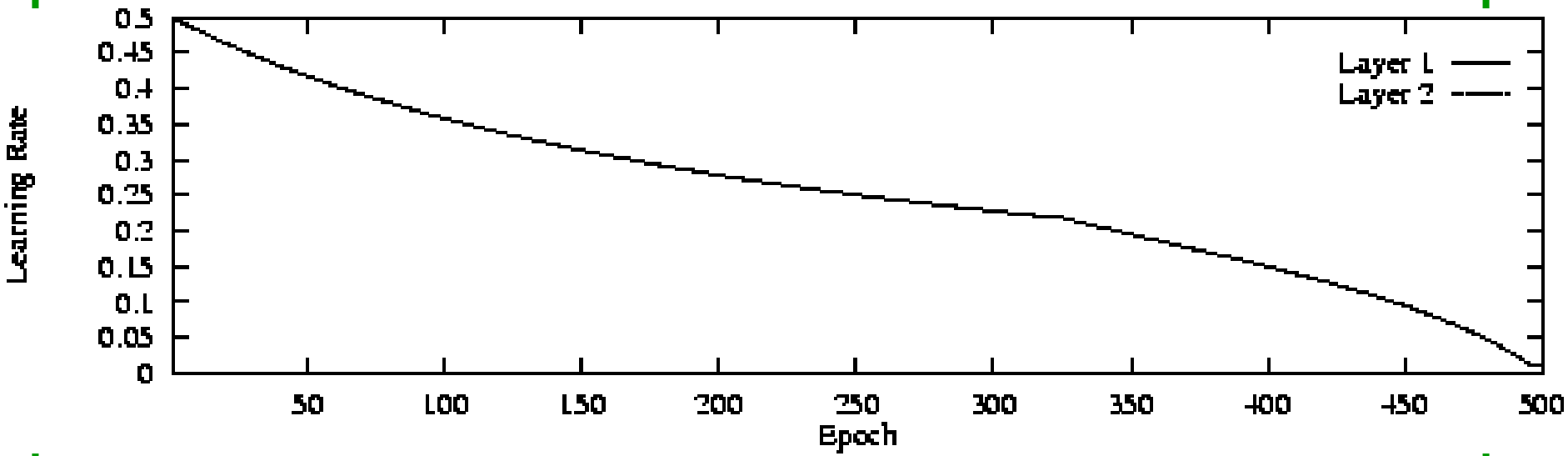
Here using the 2-stage approach:

Training

http://www.neci.nec.com/homepages/lawrence/papers/face-tr96/latex.html

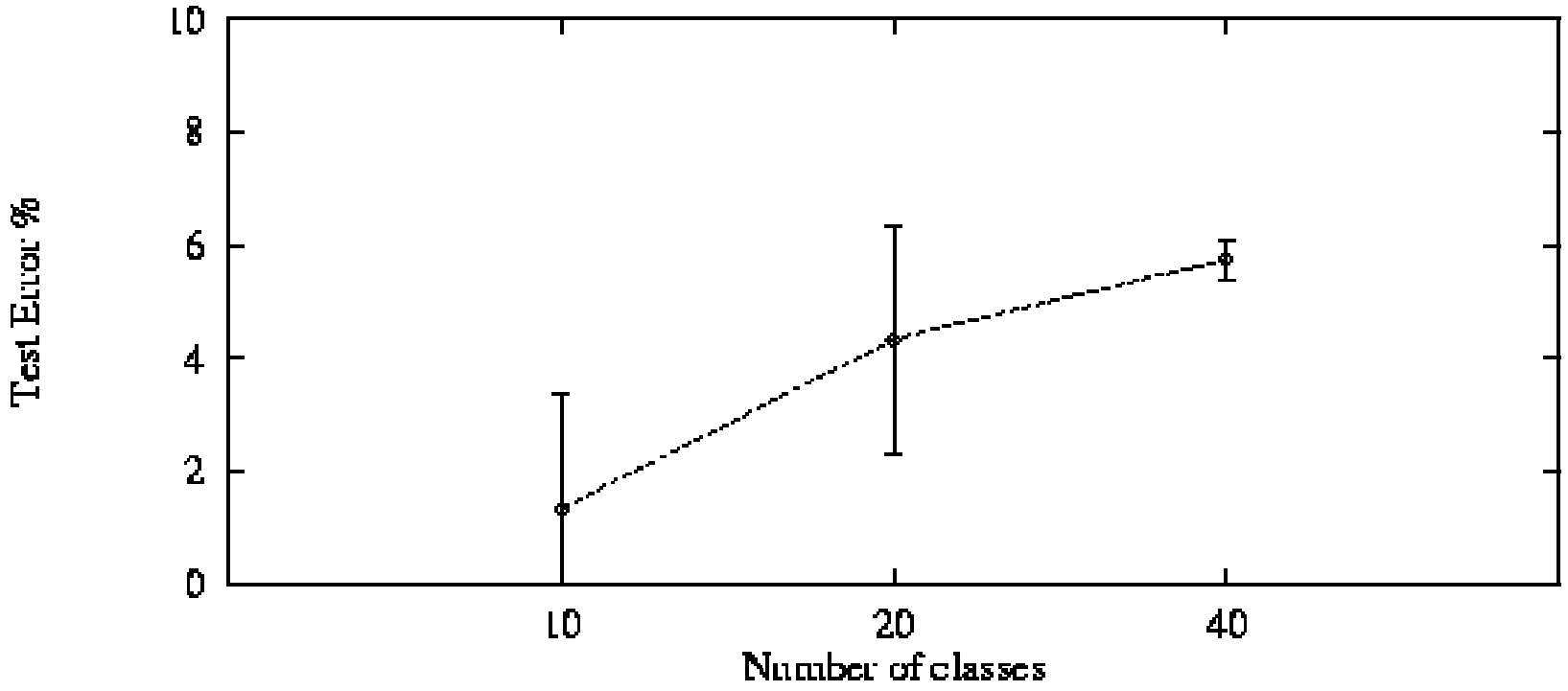


Learning rate

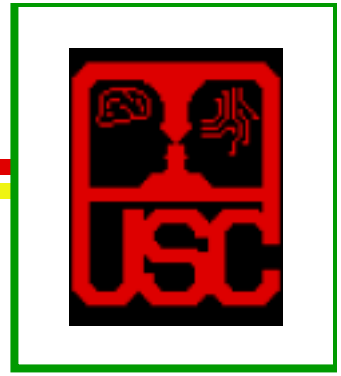


Testing / Evaluation

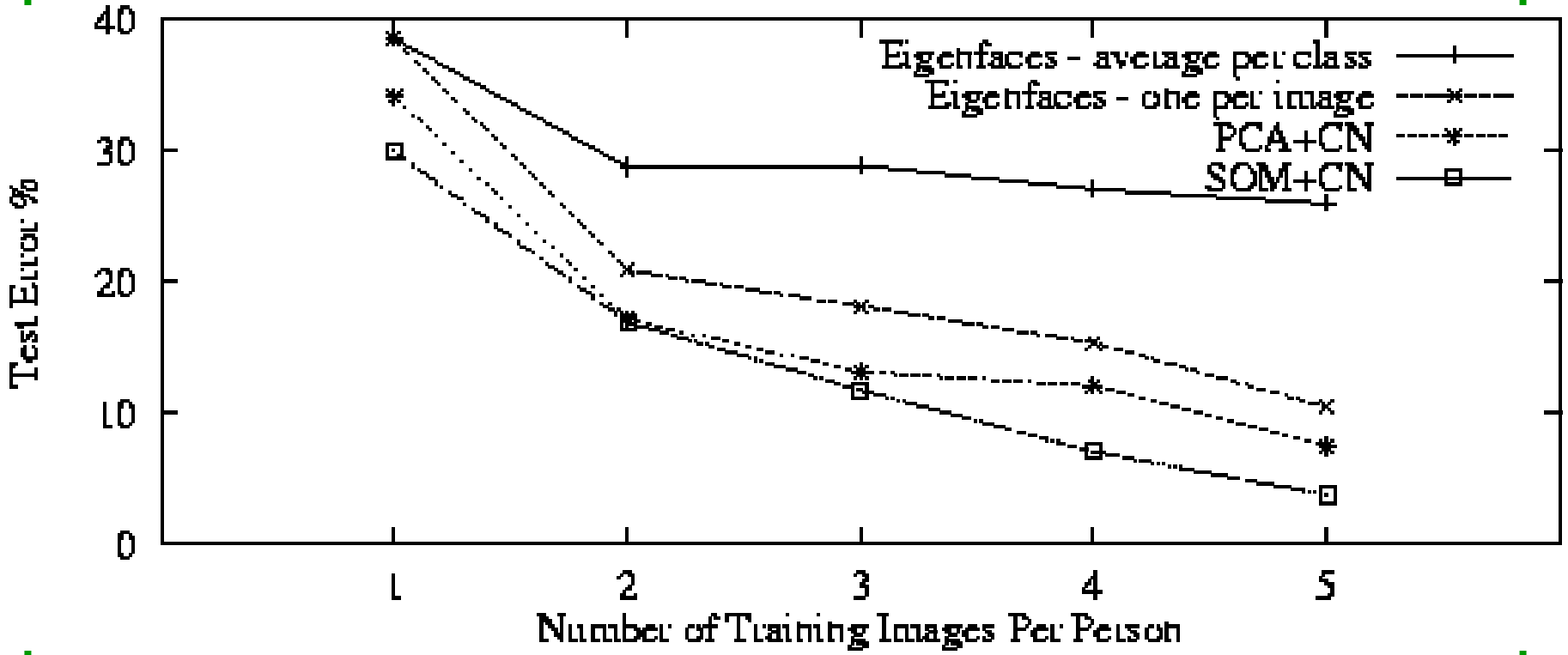
Look at performance as a function of network complexity



Testing / Evaluation



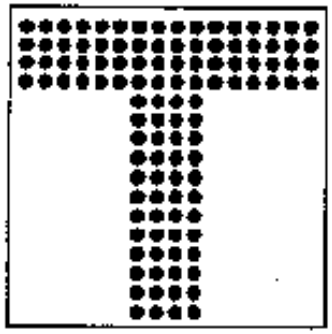
Comparison with other known techniques



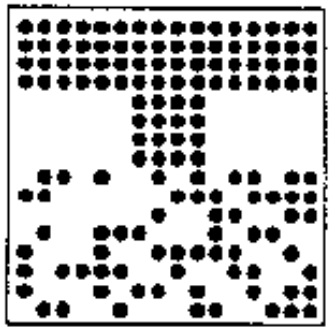
Associative Memories

http://www.shef.ac.uk/psychology/gurney/notes/l5/l5.html

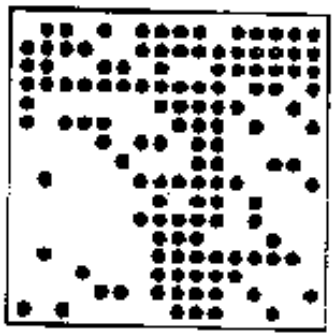
Idea: store:



Original 'T'



half of image corrupted by noise



20% corrupted by noise (whole image)

So that we can recover it if presented with corrupted data such as:

Associative Memories

How can we set the weights such as to store multiple Patterns?

Use Hebbian learning!

Result:

$$W_{ij} = 1/N \quad \underset{\text{training patterns } u}{\text{Sum}} \quad p_i^u \, p_j^u$$

See HKP chapter 2.

Associative memory with Hopfield nets

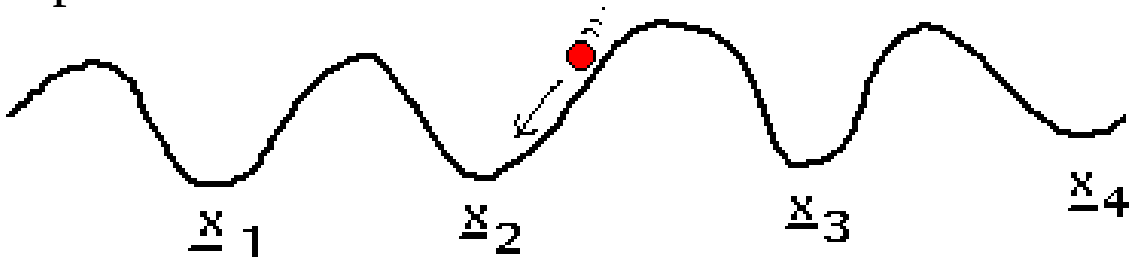
Setup a Hopfield net such that local minima correspond to the stored patterns.

Issues:

-because of weight symmetry, anti-patterns (binary reverse) are stored as well as the original patterns (also spurious local minima are created when many patterns are stored)

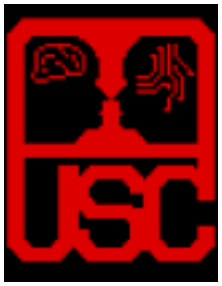
-if one tries to store more than about **0.14*(number of neurons)** patterns, the network exhibits unstable behavior

- works well only if patterns are uncorrelated



$\{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4 \ldots\}$ are the 'memories' stored

Capabilities and Limitations of Layered Networks

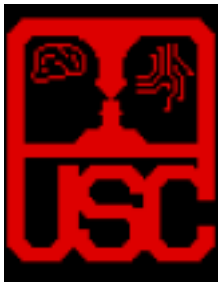


Issues:

- what can given networks do?
- What can they learn to do?
- How many layers required for given task?
- How many units per layer?
- When will a network generalize?
- What do we mean by generalize?
- …

See HKP chapter 6.4.

Capabilities and Limitations of Layered Networks



What about boolean functions?

Single-layer perceptrons are very limited:

- XOR problem
- connectivity problem
- etc.

But what about multilayer perceptrons?

We saw (midterm) that we can represent them with a network with just one hidden layer.

Capabilities and Limitations of Layered Networks

To approximate a set of functions of the inputs by
A layered network with continuous-valued units and
Sigmoidal activation function…

Cybenko, 1988: … at most two hidden layers are necessary, with arbitrary accuracy attainable by adding more hidden units.

Cybenko, 1989: one hidden layer is enough to approximate any continuous function.

Intuition of proof: decompose function to be approximated into a sum of localized "bumps." The bumps can be constructed with two hidden layers.

Similar in spirit to Fourier decomposition. Bumps = radial basis functions.

Optimal Network Architectures

How can we determine the number of hidden units?

- genetic algorithms: evaluate variations of the network, using a metric that combines its performance and its complexity. Then apply various mutations to the network (change number of hidden units) until the best one is found.

- Pruning and weight decay:
  - apply weight decay (remember reinforcement learning) during training
  - eliminate connections with weight below threshold
  - re-train

- How about eliminating units? For example, eliminate units with total synaptic input weight smaller than threshold.

For further information

See HKP:

Hertz, Krogh & Palmer: Introduction to the theory of neural computation (Addison Wesley)

In particular, the end of chapters 2 and 6.