



Computer Science @ RIT

B. Thomas Golisano College of Computing & Information Sciences

B. Thomas Golisano College of Computing & Information Sciences

RJUG: Java AI Technologies

“The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.”

-Edsger D. Dijkstra

Kurt Alfred Kluever (kurt@klover.com)

Department of Computer Science
Rochester Institute of Technology

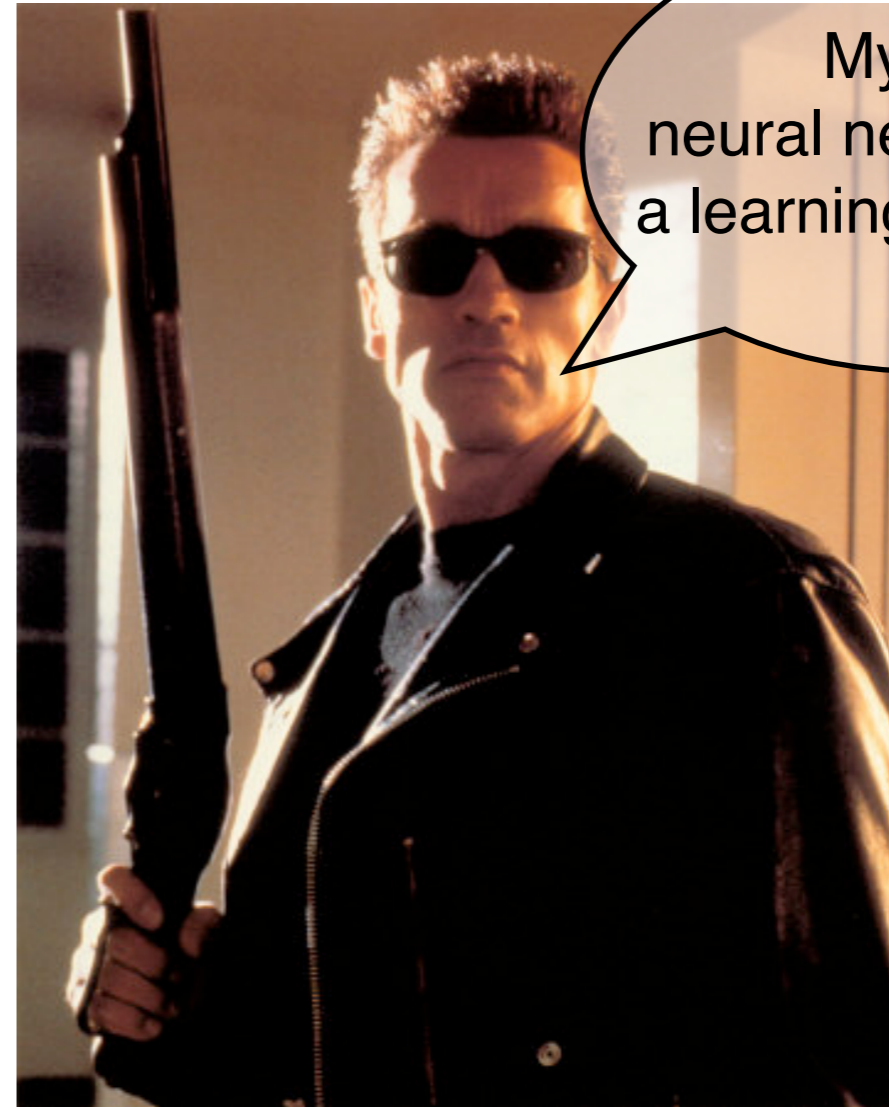
Natural Intelligence

- Infer
- Problem Solve
- Think abstractly
- Comprehend ideas
- Reason
- Plan
- Learn



Artificial Intelligence

- Create intelligent agents
 - Perceive environment
 - Gain knowledge
 - Make decisions
 - Reason
 - Plan
 - Learn



AI Tools

- Artificial Neural Networks (ANN)
 - Model biological neural networks
 - Example: JOONE
- Expert Systems (ES)
 - Rule based evaluation of inputs
 - Example: JESS
- Genetic Algorithms (GAs)
 - Class of evolutionary algorithm
 - Example: JGAP

Artificial Neural Networks

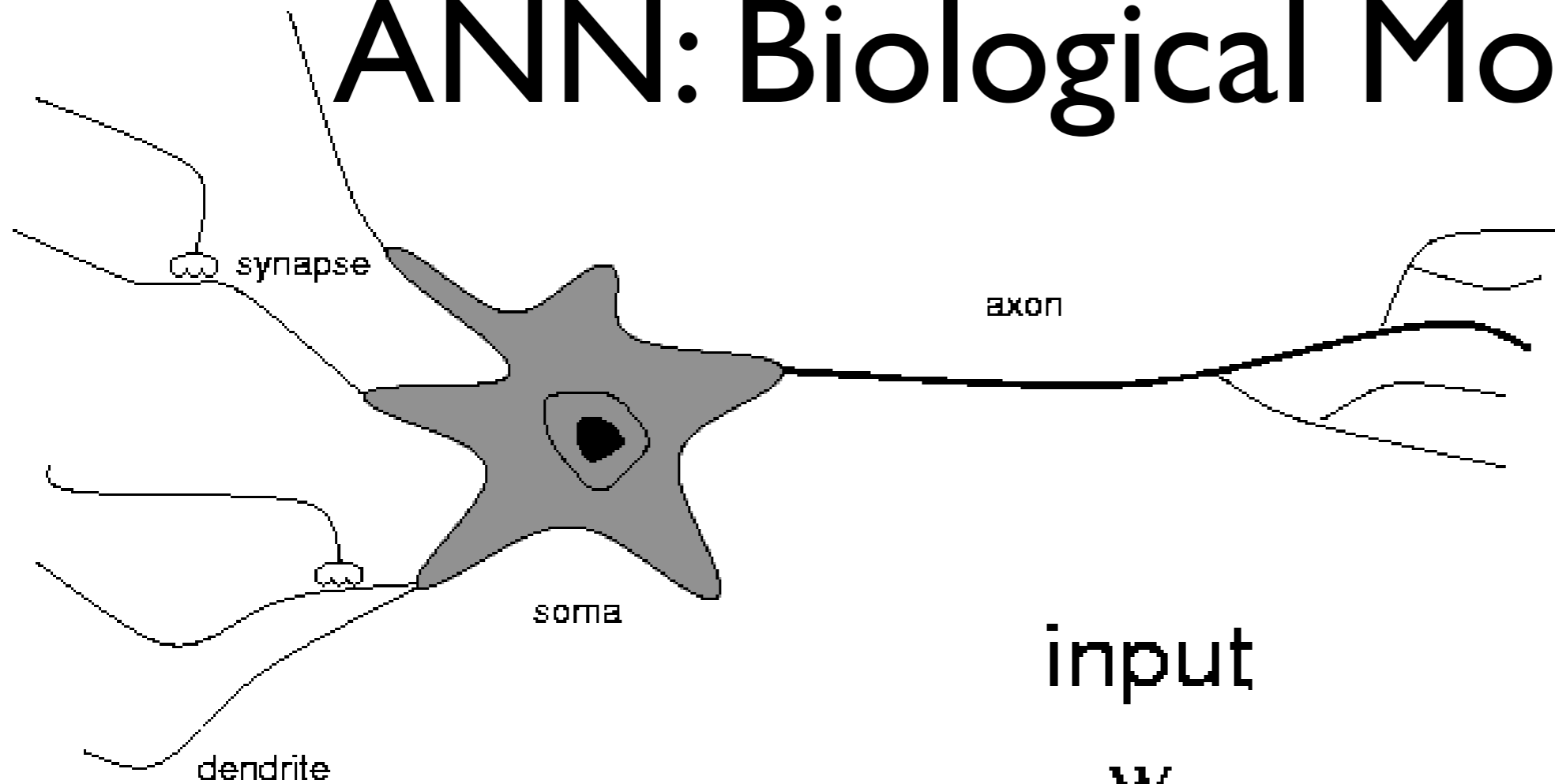
- Models the biological neural network (brain)
- Interconnected group of nodes (neurons)
 - Neurons are simple processing elements
 - Neurons perform some function to inputs
 - Inputs of neurons are weighted
 - Connections between neurons forms a network
- ***** NORMALIZE YOUR INPUTS *****
 - To values $[-1, 1]$ or $[0, 1]$: $(Val - Min)/(Max - Min)$

ANN: The Good + the Bad

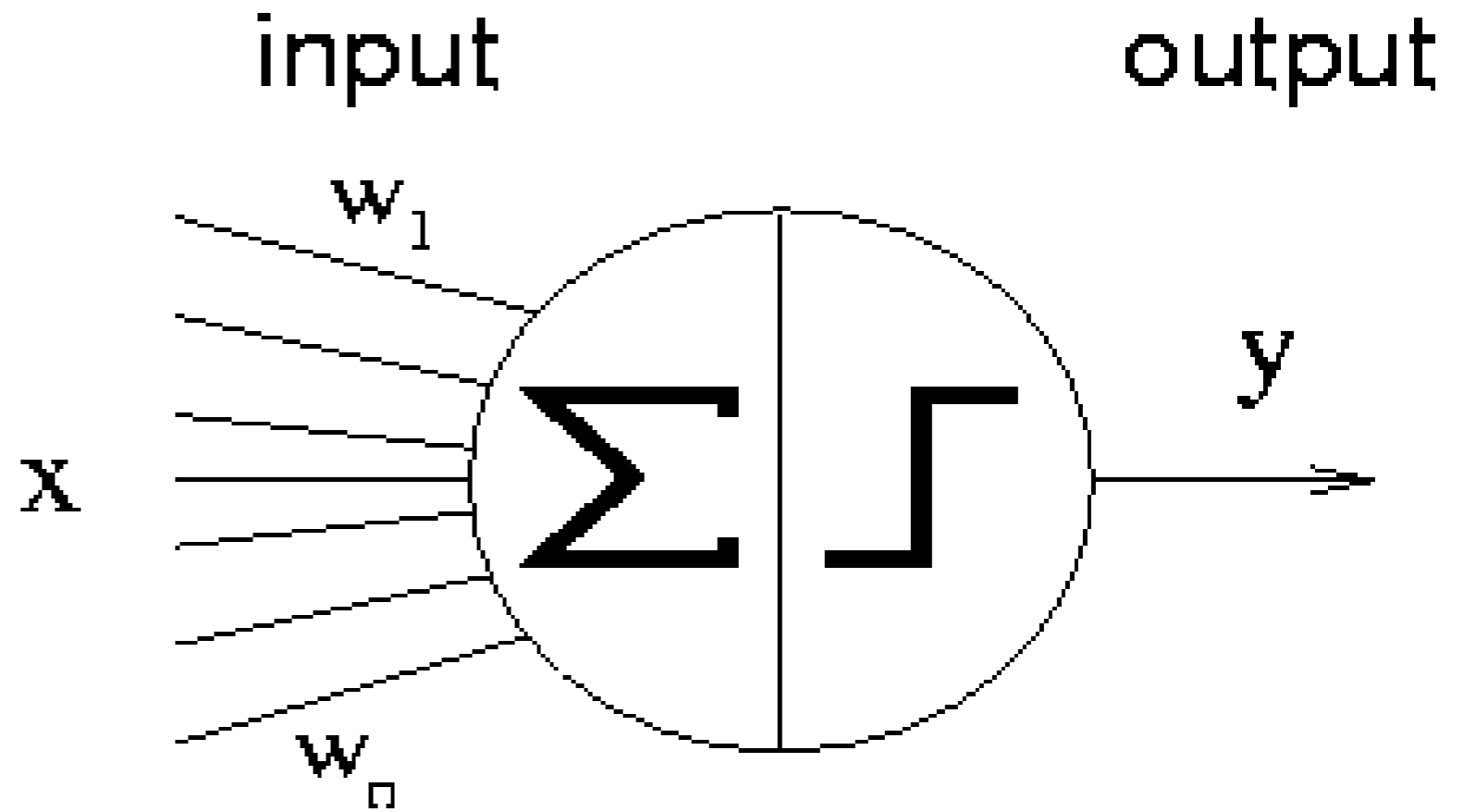
- Advantages:
 - Uncertainty and imprecision tolerance
 - Adaptability, learning ability
 - Knowledge discovery
 - Maintainability
- Disadvantages
 - “Black box”, explanation ability
 - Knowledge representation



ANN: Biological Model



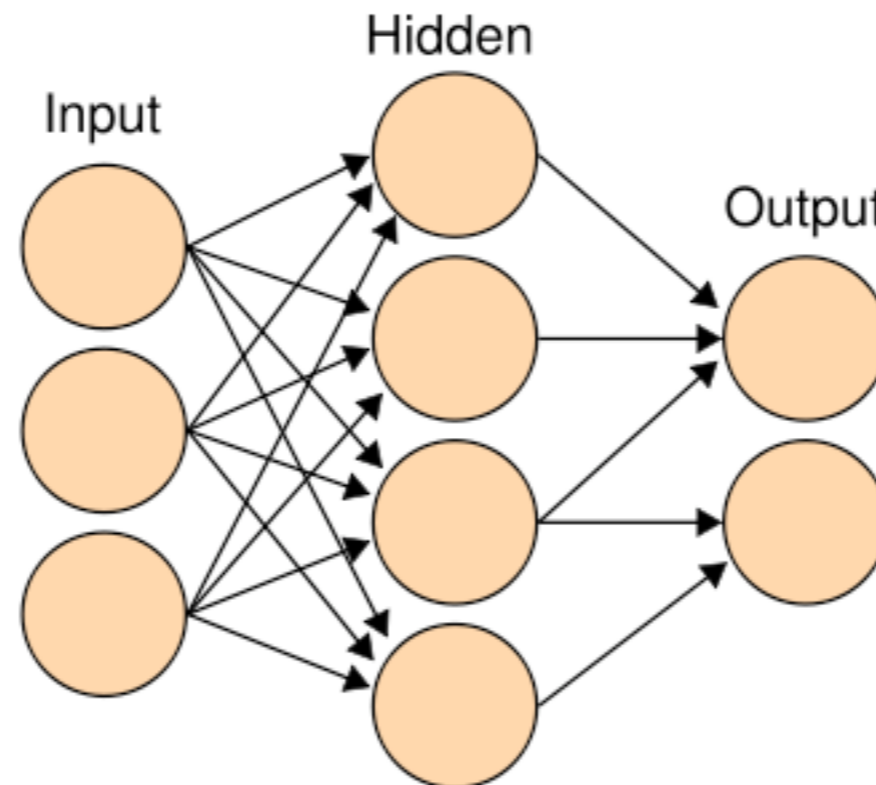
Soma \Leftrightarrow Neuron
Dendrite \Leftrightarrow Input
Axon \Leftrightarrow Output
Synapse \Leftrightarrow Weight



ANN: Computational Model

$$f(x) = K \left(\sum_i w_i g_i(x) \right)$$

- $f(x)$ is a composition of other functions $g(x)$
- K is some function (hyperbolic tangent, etc)
- w is the weight being applied to the input



ANN Example:



Java Object Oriented Neural Engine by Paolo Marrone

- JOONE: Java Object Oriented Neural Engine
- Component based framework with linkable components
- Supervised learning
 - Feed forward, recursive, time delay, standard back prop, resilient back prop
- Unsupervised learning
 - Kohonens SOMs, PCA
- Or mix some of the above

JOONE: GUI

The screenshot shows the Joone Neural Net Editor interface. At the top, the title bar reads "JoonEdit - Joone Neural Net Editor - C:\Programmi\Joone\samples\editor\scripting\InputConnector\ValidationSampl...". The menu bar includes "File", "Edit", "Align", "Attributes", "Tools", "Window", and "Help". Below the menu is a toolbar with various icons for editing and simulation.

The main workspace contains a neural network diagram. A text box at the top center states: "The validation phase of this net is controlled by a script. To see the code, open the Macro Editor and look at the netStarted and cycleTerminated events". The diagram shows a flow from "Input Connector InputTraining" and "Input Connector InputValidation" to a "Learning Switch Input Data" block. This block feeds into a "Linear Input 13" block, which connects to a "Sigmoid Hidden 10" block. The hidden layer connects to a "Sigmoid Output 1" block. This output block feeds into a "Teacher Teacher" block, which in turn feeds into a "Chart RMSE" block. A green box highlights a "Normalizer InputPlugin 1" block connected to an "Excel Input InputLayer 1" block, which also feeds into the "Learning Switch Input Data" block.

In the bottom right corner, there is a window titled "RMSE" showing a line graph. The Y-axis represents the Root Mean Square Error (RMSE) from 0.0 to 0.2. The X-axis represents the number of cycles from 0 to 2000. The graph shows a blue curve that starts at approximately 0.2 and rapidly decreases, leveling off around 0.02 after about 1000 cycles. A text label on the graph indicates "Current cycle is 2000".

JOONE: Core Engine

- Allows precise control over building of NN
- Define the input layers, hidden layers, output layers, number of neurons, type of functions, etc.
- LOTS of code required to hook up layers to one another; even more to train + test!

```
LinearLayer input = new LinearLayer();
SigmoidLayer hidden = new SigmoidLayer();
SigmoidLayer output = new SigmoidLayer();
input.setRows(2);
hidden.setRows(3);
output.setRows(1);
FullSynapse synapse_IH = new FullSynapse();
FullSynapse synapse_HO = new FullSynapse();
input.addOutputSynapse(synapse_IH);
hidden.addInputSynapse(synapse_IH);
hidden.addOutputSynapse(synapse_HO);
output.addInputSynapse(synapse_HO);
```

JOONE: Joone Tools

- Create, train, and interrogate a simple feed forward network in 3 lines of code!
- Supports creation of three types of networks:
 - Supervised: Standard Feed Forward (acyclic network where info flows in 1 direction only)
 - Supervised: Time Delay (allows for temporal learning of events)
 - Unsupervised: SOM (maps points in input space to points in output space while preserving dimensionality and topology)

JOONE: Layers

- Choice of layer depends on problem
 - Linear layer: function approximation
 - Ex: $f(4) = 6$; $f(6) = 12$; $f(5) = ?$
 - Sigmoid/Logistic layer: binary classification
 - Ex: cat or dog
 - Softmax layer: 1 of C classification
 - Ex: 'a', 'b', 'c', 'd', etc.

JOONE: Create Helper

- `JooneTools.create_standard`
(`networkArchitecture`, `layerType`)
 - Create a MLP network with the architecture specified in the first parameter and a output layer specified by the second parameter
 - `create_standard(new int[]{2, 2, 1}, 2)`
 - 2 inputs, 2 hidden, 1 output; sigmoid/logistic layers
- `JooneTools.create_timeDelay(int[], int, int)`
- `JooneTools.create_unsupervised(int[], int)`

JOONE: Train Helper

- `JooneTools.train(nnet, inputs, outputs, epochs, rmse, interval, notification, async)`
 - `nnet` = NeuralNetwork created by `.create()`
 - `inputs` = Your input training data (`int[]`)
 - `outputs` = Your output training data (`int[]`)
 - `epochs` = Number of cycles to train until
 - `rmse` = Stop training if this RMSE is met
 - `interval` = How often you want updates
 - `notification` = How to receive updates
 - `async` = Whether to train asynchronously

JOONE: Interrogate Helper

- `JooneTools.interrogate(nnet, inputs)`
 - `nnet` = Your trained neural network
 - `inputs` = Your input testing data
- Returns an output array containing the results of the interrogation
- Careful, output will not be exact!
 - Your turn to classify the resulting doubles

JOONE: Test Helper

- `JooneTools.test(nnet, inputs, outputs)`
 - `nnet` = Your trained network
 - `inputs` = Your input testing data
 - `outputs` = Your output testing data
- Allows you to calculate the RMSE of the trained network given input and output data
 - Typically you don't have the output data for real world applications, so this is for testing purposes only

JOONE Examples + Demo

- Solve software development effort estimation
 - Uses a large historical db to estimate times
- Handwritten digit recognition
 - Uses chain codes as the feature vectors
- JOONEGAP: NN Evolutionary Environment
 - Uses JGAP GAs to evolve JOONE ANNs
- Demo time: XOR and RJUG Attendance!

Expert Systems

- Set of fixed rules (supplied by experts) define how the program works
- Declarative rules
- “Reasoning”
- Think: bunch of if..then statements
- Provides excellent explanation abilities and knowledge representation (rule chaining)
- Poor learning abilities (read: none)

ES Example:



the Rule Engine for the Java™ Platform

- Jess: Java Expert System Shell
 - Rule engine utilizing an enhanced version of the Rete algorithm (Forgy 1982) which performs many-to-many matching
 - Continuously applies a set of rules to a collection of facts through pattern matching
 - Allows for backward chaining, working memory queries, and interaction with Java
 - Commercial product :(Free for academics :)
 - Rules, facts, functions in CLIPS (LISP syntax)

Genetic Algorithms

- Technique used to find solutions to optimization or search problems
 - Initialize to a randomly generated population
 - Evaluate the fitness of every individual
 - Select the most fit individuals and modify:
 - Recombine (crossover): combine two parents
 - Natural Selection or Roulette Wheel
 - Randomly mutate
 - Terminate after fit enough or n generations

Questions?