

NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models

Radek Grzeszczuk
Demetri Terzopoulos
Geoffrey Hinton

Presented by Joseph Carnahan
October 20, 2003

Physical Simulation

- Produces very realistic output
- Highly automated

One major downside:

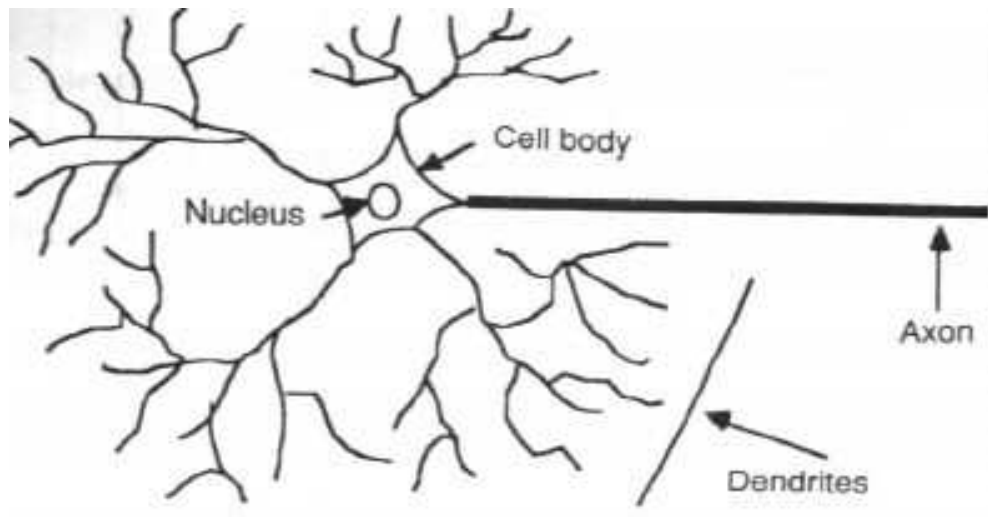
- Computationally expensive

The NeuroAnimator Approach

Replace simulator with an emulator, trained to behave the same as the simulator

- Faster than simulation
- AI offers “learning systems”
 - Artificial neural networks

Artificial Neuron

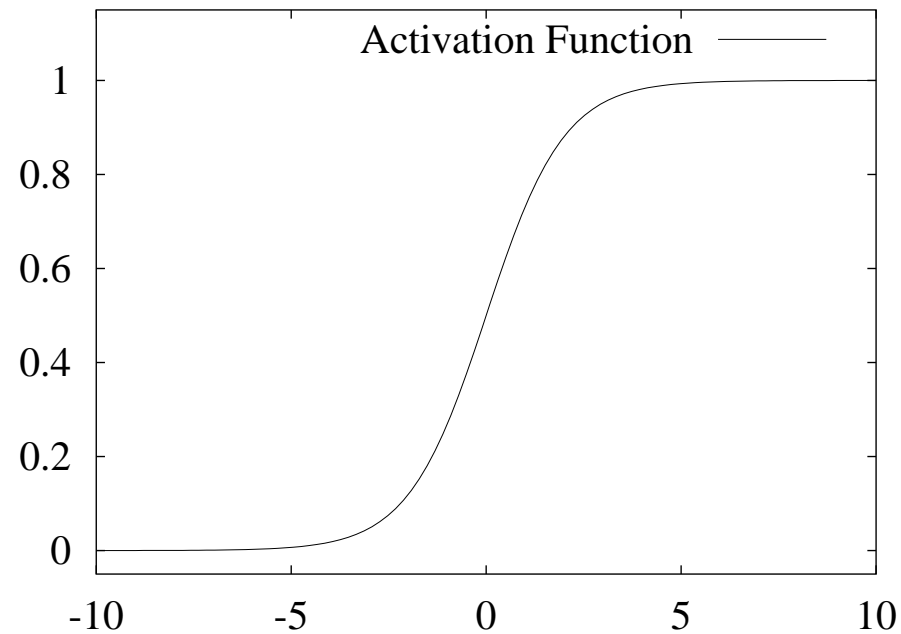


Neuron modeled as

- Set of weighted inputs (real numbers)
- Single output (another real number)
 - Linear combination of inputs (basis function)
 - Processed to determine if neuron fires (activation function)

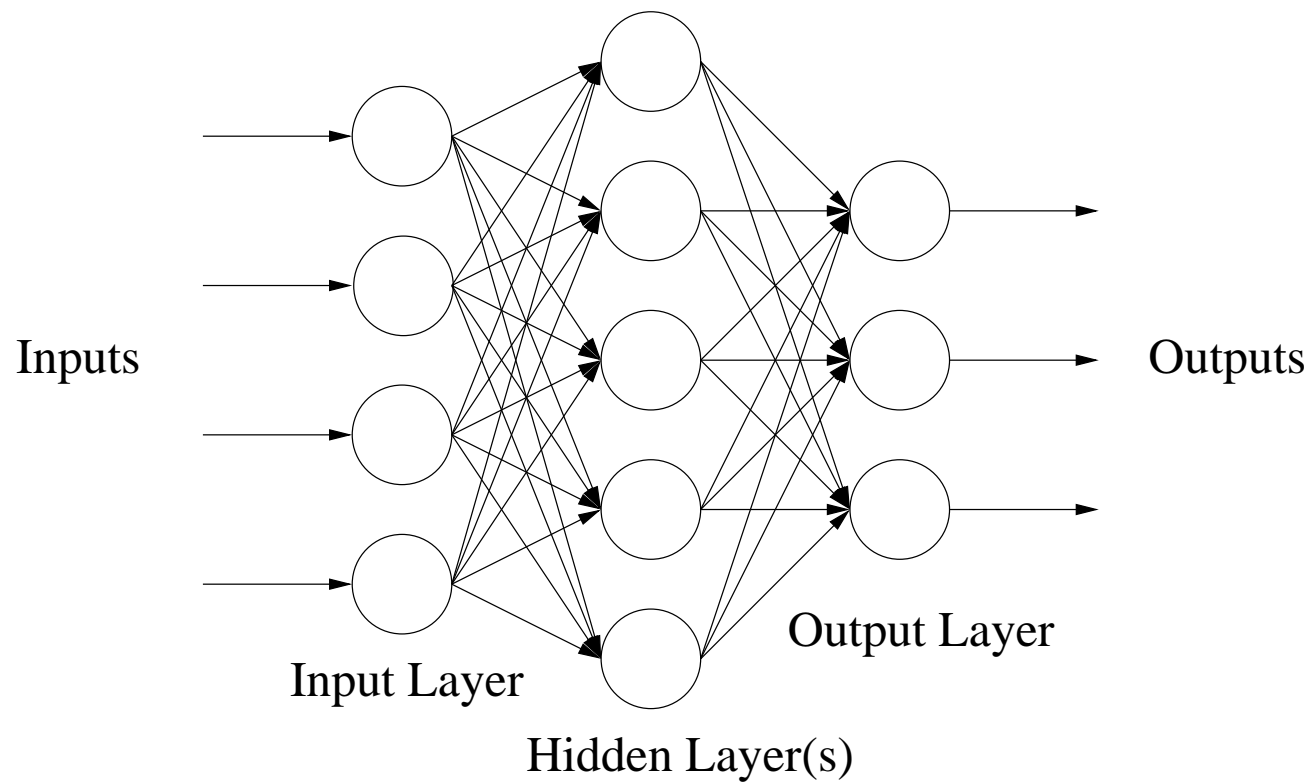
The Activation Function

- Reflects whether or not a neuron fires
- This algorithm uses logistic sigmoid function $\frac{1}{1+e^{-z}}$:



Artificial Neural Network

Neurons connected in layers:



Various Details

- Bias input of 1 added to hidden layers and output layer
- Input and output layers use linear activation functions, hidden layer uses sigmoidal function
- How many hidden layers and neurons to use?
 - Three layers (one hidden) is common
 - Avoid underfitting (not enough nodes)
 - Avoid overfitting (too many nodes, although that can be overcome with enough data)

Training the Neural Network

Given n representative samples \mathbf{x} of possible inputs and a function Φ that we would like to emulate, we want to minimize

$$E(\mathbf{w}) = \sum_{\tau=1}^n e(\mathbf{x}_{\tau}, \mathbf{w})$$

where

- \mathbf{w} is the vector of weights on inputs in the network
- e is the error between the current output of the neural network \mathbf{N} and the output of the target function Φ :

$$e(\mathbf{x}, \mathbf{w}) = \|\Phi(\mathbf{x}) - \mathbf{N}(\mathbf{x}, \mathbf{w})\|^2$$

Optimization Methods

- Used tools built into Xerion neural network simulator
- Basic idea: Gradient descent
 - Find direction of changes in weights that will reduce error as much as possible
 - Take a step (i.e. adjust weights) in that direction
 - Repeat until satisfied (i.e. error is less than ϵ)

More Advanced Methods

- Line search: Improves on this by varying step size
- Conjugate gradient search: Change in w at time t must be orthogonal to change at time $t - 1$
- Momentum method: Change in \mathbf{w} at time t is added to change at time $t - 1$ times a momentum factor δ

In this paper, conjugate gradient and momentum methods were used.

Plugging in the Neural Network

Problems to overcome:

- Range of inputs is very large, hard to capture with neural net
 - Hidden-layer neurons produce output between 0 and 1
 - Output layer neurons produce output which is linear function of hidden-layer neurons' output

- State of system includes lots of superfluous data
 - Many behaviors do not depend on position or orientation
 - Makes training more difficult
- Training a network with too many inputs becomes impractical under even the best conditions
- Errors could accumulate, since mis-computing the state of the system means providing incorrect inputs to the system at the next time step

Transforming Inputs and Outputs

Solution: Apply transformations to inputs and outputs

- NeuroAnimator computes *changes* in system state, rather than system state values themselves
- NeuroAnimator computes changes relative to object coordinates, rather than in world coordinates
- Input data is normalized before being given to NeuroAnimator, while outputs are “un-normalized”

Hierarchies of NeuroAnimators

To speed the learning process, model complex systems with multiple NeuroAnimators

Similar to using hierarchical breakdown of objects for any graphical object representation

“Regularizing” to Avoid Errors

For models where error accumulates over time, the authors suggest applying a few time steps of physical simulation between every (large) emulation time step

Note: Authors took yet another shot at Euler integration here...

NeuroAnimators and Control

Control problem in animation:

- Would like to know what forces to apply to meet a desired goal
- Optimization problem (again)
- Combination of physical interactions is not differentiable, hard to optimize
- In NeuroAnimator, it *is* possible to determine partial derivative
- Use this to find optimal controller values

NeuroAnimator Advantages

- Considerable computational speedup over physical simulation
- Quantifiably close results to physical simulation
- Supports controller synthesis

NeuroAnimator Disadvantages

- Must have existing physical simulation for training
- Must be trained on inputs that will be representative of the inputs used for the actual animation
- Physical constraints are not enforced
 - SUV sliding sideways

Questions?