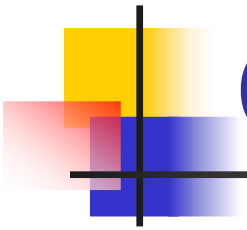
Probabilistic & Machine Learning Applications

Joel Coburn
Ilya Katsnelson
Brad Schumitsch
Jean Suh



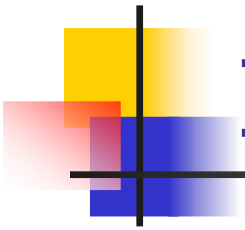
Outline

- Genetic algorithms
- Functionality of learning algorithms
- Characteristics of neural networks
- Available parallelism
- System bottlenecks
- Trade-off analysis

Genetic Algorithms (GE)

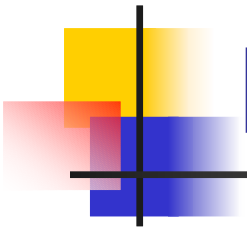
- A search procedure that optimizes to some objective
- Maintains a population of candidate solutions
- Employs operations inspired by genetics (crossover and mutation) to generate a new population from the previous one

```
T = 0;
Initialize and evaluate [p(t)];
While (not stop_condition) do {
     P'(t) = variation [p(t)];
     Evaluate [p'(t)];
     P(t+1) = select [p'(t),p(t)];
     T = t + 1;
}
```



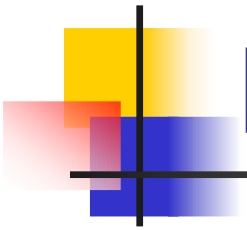
Implementation

- Massively parallel
  - Most iterations are parallel = data and thread parallel
  - Almost no communications between independent runs
  - Several versions can be executed in parallel
  - Different algorithm models map naturally to a specific HW architecture
- Performance depends on the target problem and implementation:
  - Host OS support
  - Cache use
  - Communication between nodes



Learning Algorithms

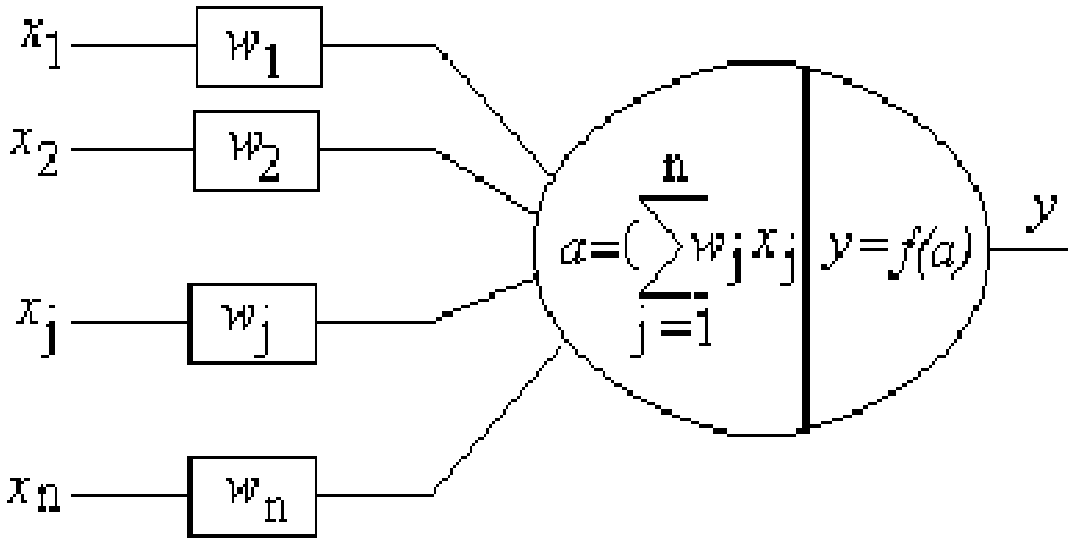
- The set of problems which are easily solved in nature but not by humans
- Requires massive parallelism
- Biologically-inspired technique based on learning – Artificial Neural Networks (ANN)



Neural Network Overview

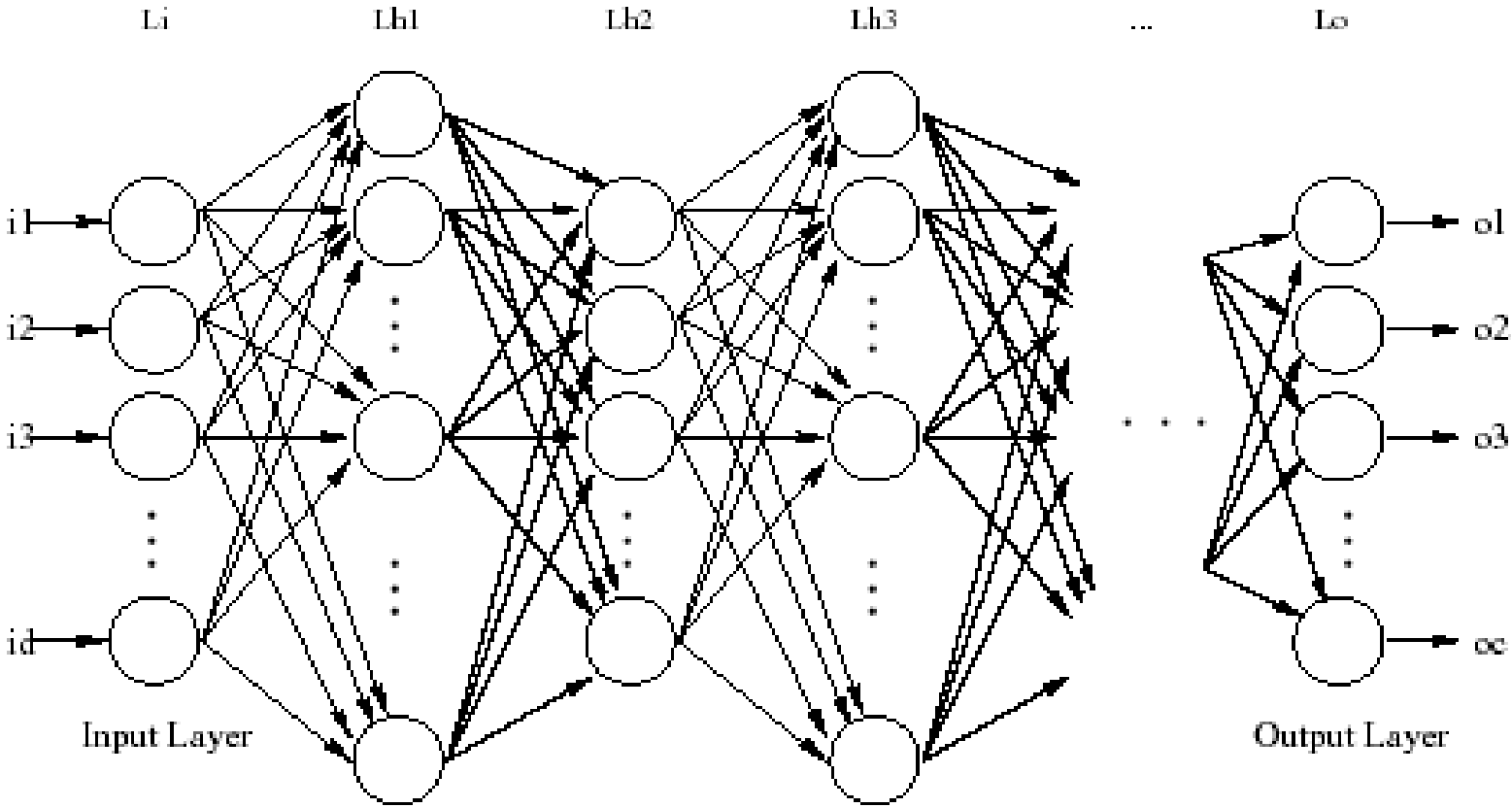
- Basic building block neuron or node
- Propagate output from weighted linear input combination
- Network characterization:
  - Feed-forward
  - Recurrent(feedback)

Artificial Neuron

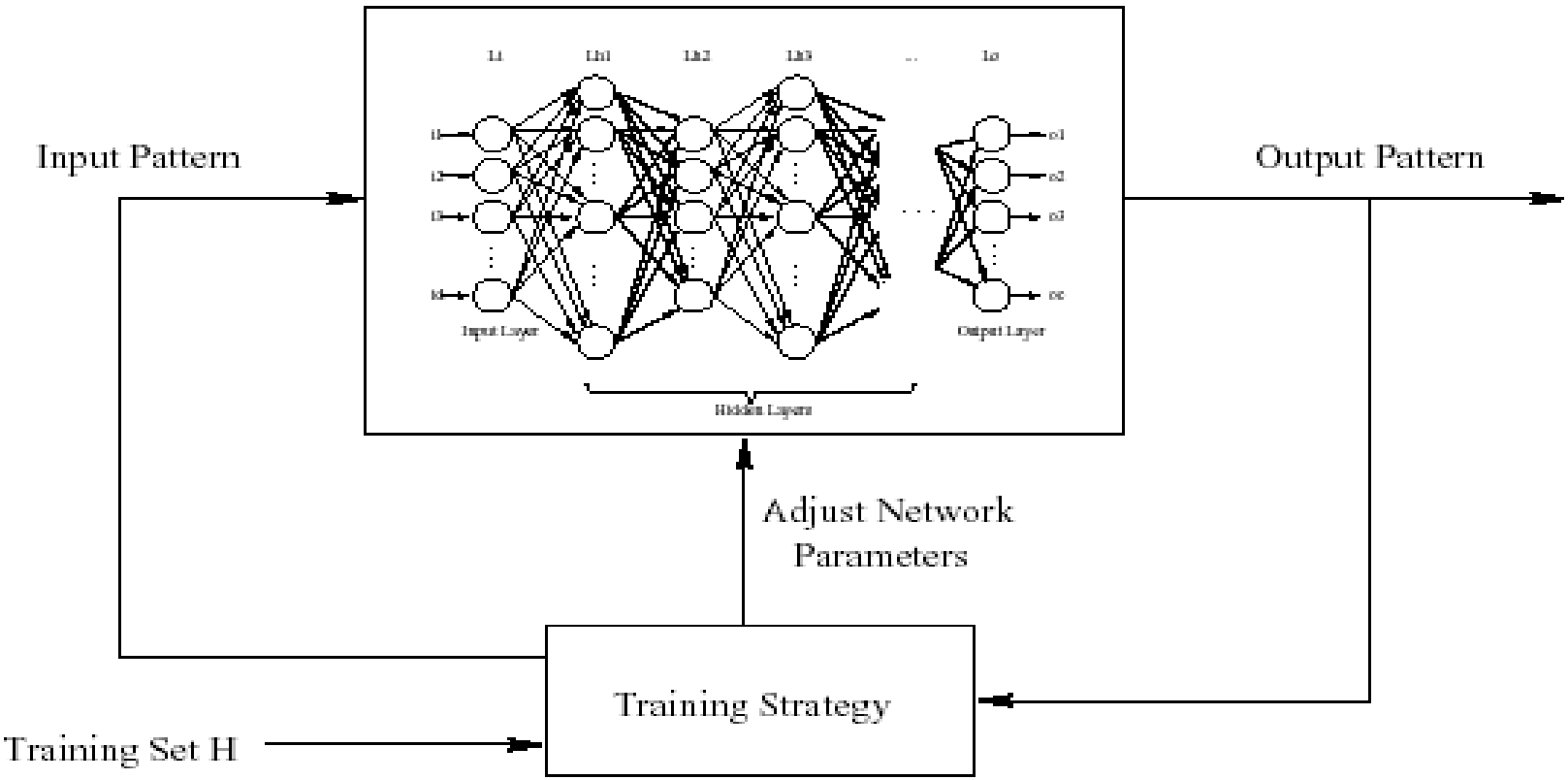


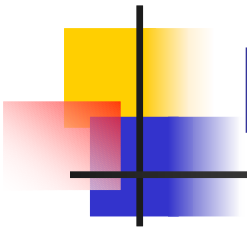
Activation function is weighted summation of the inputs
Can represent in vector notation as $a=w^T x$

Neural Network



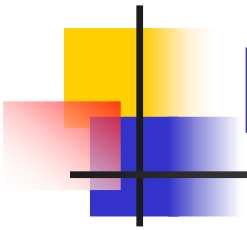
Training Algorithm





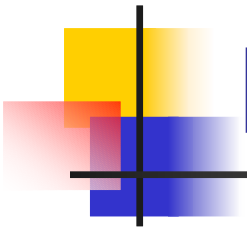
Behavior During Execution

- Learning Phase
  - Iterate through multiple data sets
  - Feedback loop to provide correction term for weights
- Processing Phase
  - Normal mapping through the network



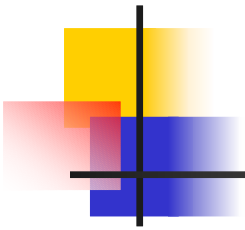
Parallelism

- Significant parallelism available
- Conceptually, a processor per neuron (TLP)
- 3 non-orthogonal ways to exploit it



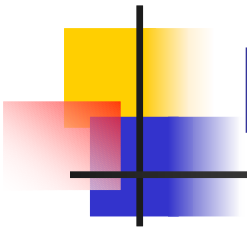
Node Parallelism

- One processing element per node
- Private storage for weights of each node
- Serial data transfer between nodes
- Local storage for output data



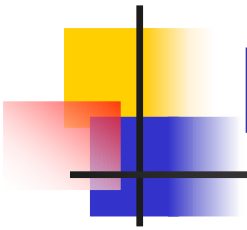
Layer Parallelism

- One processing element per layer
- Central storage for all weights of a layer (shared memory)
- A buffer to store input and output values
- Serial or parallel data transfer between layers (depends on # of multipliers)



Link Parallelism

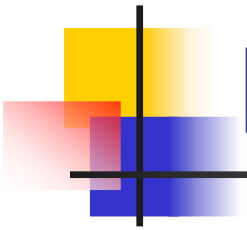
- One processing element per connections
- Each connection of neuron is calculated in parallel
- # of multipliers = # of connections
- Central storage for all weights of a layer
- Parallel data transfer between layers



Data Set and Working Set Size

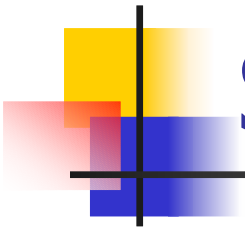
- Most applications use only 10s of inputs
  - Larger networks are rarely used because of the unacceptable learning-time required
- Working set size is a function of the number of neurons in the system
  - Each neuron operates on several data elements at a time

Arithmetic Operations and Memory Access

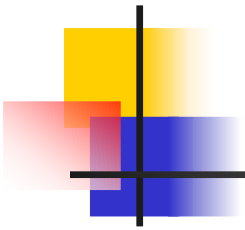


- For the learning or processing phases, the data set will have to be fetched from memory
  - Either one large parallel operation (too many ports) or slower serial access
- Inside each PN:
  - Read from SRAM to find weight for each input – can be direct-mapped to fit data set
  - Ratio of arithmetic/memory operations for each PN is about 2

Bottlenecks on Current Systems

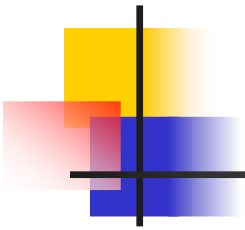


- **Communication bandwidth**
  - Many interconnections between processing elements
- Cost
  - Approximate to the number of processors required
- Complex programming interfaces
- Power consumption
- Large area



Performance Evaluation

- ANN performance is measured by two metrics:
  - Processing Speed: Multiply and accumulate operations performed in unit time = MCPS (Millions of Connections Per Second)
  - Learning Speed: Rate of weight updates = MCUPS (Millions of Connection Updates Per Second)
- These metrics ignore learning convergence



References (1)

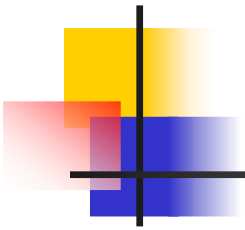
- Aybay, I., Cetinkaya, S., Halici, U., 1996, "Classification of Neural Network Hardware", *Neural Network World*, IDG Co.,Vol 6 No 1.
- Burr, J., 1993, "Digital Neurochip Design," Chapter 8 of *Parallel Digital Implementations of Neural Networks*, H. W. Przytula and V. K. Prasanna, eds., Englewood Cliffs: Prentice Hall.
- Burr, J., 1991, "Energy, Capacity, and Technology Scaling in Digital VLSI Neural Networks", *IEEE International Conference on Computer Design.*

References (2)

- Cornu, T., Ienne, P., 1994, "Performance of digital neuro-computers.", *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*.
- Heemskerk, J.N.H, 1995, "Overview of Neural Hardware.", later published in his Ph.D. thesis.
- Ienne, P., 1993, "Architectures for Neuro-Computers: Review and Performance Evaluation.", *EPFL Technical Report 93/21*.

References (3)

- Ienne, P., Kuhn, G., 1995, "Digital Systems for Neural Networks.", *Digital Signal Processing Technology*, SPIE Optical engineering.
- Jahnke, A., Klar, H., Schoenauer, T., 1998, "Digital Neurohardware: Principles and Perspectives.", *Neural Networks in Applications*, Institute of Microelectronics, Technical University of Berlin.



References (4)

- Whitley D, Rana S, Dzubera J, et al. "Evaluating evolutionary algorithms." *Artificial Intelligence*. 85 (1-2): 245-276 Aug 1996.
- Alba E, Nebro AJ, Troya JM. "Heterogeneous computing and parallel genetic algorithms." J*ournal of Parallel and Distributed Computing*. 62 (9): 1362-1385 Sep 2002.
- "Solutions to parallel and distributed computing problems : lessons from biological sciences." Edited by Albert Y. Zomaya, Fikret Ercal, Stephan Olariu. New York : John Wiley, c2001.