EECS 484 Computational Intelligence  
Fall 2009  
Mon/Wed 9:00-10:15 am; Glennan 716

Instructor: Wyatt Newman, professor, EECS Dept  
Office: Glennan 516  
Office phone: 368-6432  
Email: wsn@case.edu or wyatt.newman@case.edu

Office hours: Mon 10:30-11:30, Tue 9:30-10:30, Wed 1:00-2:00, Thurs 3:00-4:00

Text: Neural Networks (3rd Ed), Simon Haykin, Prentice Hall  
Addl Refs: additional readings, as posted on Blackboard

Content: Biologically inspired algorithms for computational intelligence, including: unsupervised learning (clustering); self-ordering maps; function approximation with neural networks; associative memory; solving planning problems with parallel distributed processing; genetic algorithms for optimization; real-time recurrent neural nets and back-propagation through time.

Goals/Outcomes: The goals of this course are to introduce a variety of computational paradigms concerning multivariate objects, adaptation in computation, and parallel distributed processing (PDP). A survey of techniques will be presented through lectures and readings and explored through problem sets in the form of programming labs. Students will acquire a grasp of the current strengths and weaknesses and future potential of biologically-inspired computing. They will master a variety of specific algorithms useful for: discovering patterns in data; recognizing and classifying approximate patterns; performing function approximation through PDP techniques; performing optimizations through genetic/evolutionary algorithms; encoding, recovering and correcting associations via content-addressable memory; and computing plans and designing temporal pattern generators via PDP.

Homework: The bulk of this material is learned through doing the problem sets. There will be 8 or 9 problem sets and a final project. Homework is to be submitted electronically via the Blackboard digital dropbox. The final project is due on the last day of classes before reading period, Friday, 12/4. For the problem sets, starter code will be provided. Students are encouraged to work collaboratively at the conceptual level, but each solution submitted should be an independent effort in the details. (E.g., you may discuss equations, but do not exchange code).
**Exams:**
There will be two in-class tests on:
- Test 1: Wednesday, 10/14
- Test 2: Wednesday, 12/2
There is a final project, but no final exam.
The exams will be open book, open notes, calculators allowed but no computers.

**Grading:**
Homework will count for 50% of the final grade. The lowest homework grade will be forgiven (assigned equal to the class average). The in-class exams will count for 20% each, and the final project will count for 10%.

**Lectures:**
Mondays/Wednesdays 9:00-10:30 in Glennan 716.

**Website:**
Problem sets and supplemental reading will be posted on Blackboard: https://blackboard.cwru.edu/

**Schedule of topics:**
There will be 8 or 9 assigned problem sets, associated with the topics below (approximately 2 weeks per topic). All problem sets will be posted on Blackboard.

**Introduction:** Scan Intro chapter of text. This is a dense overview of topics in neural networks (not all of which will be covered in this course).

**Topic 1: Clustering** Refer to readings posted on Blackboard (not covered in the textbook), plus text section 5.5. Clustering is a method of “unsupervised learning”, in which data patterns are organized by groups using some similarity metric. Subsequently, attributes found to be dominantly associated with specific groups are inferred to be true of new patterns that are classified within existing groups.

**Topic 2: Self-Organizing Maps.** Refer to readings posted on Blackboard. Also, text Chapter 9. By introducing some cross-talk influence between cluster centers, and introducing a slightly different distance metric, clustering can be mutated into a means to organize data geometrically. This may be useful in creating organization such as retinotopic maps in the brain.

**Topic 3: Competitive Neural Networks.** See on-line notes. Neural networks—an architecture of fine-grained distributed processing based on abstractions from neuroscience—can be constructed in a variety of connectivities. With feedback (recurrence), a network can be made to recognize a closest match of an input pattern to one of a collection of stored patterns. Such a network is “MaxNet”.

**Topic 4: Training Neural Networks by Back-Propagation.** See on-line notes. Read Ch 1 (single-layer perceptrons) and Ch 4 (multi-layer perceptrons). Focus on sections 4.1 through 4.6 (back-propagation algorithm). Neural networks will be introduced in the context of function approximation (input/output mappings). Single-layer perceptrons will be described as a means of computing linear discriminators (decision planes). Limitations of single-layer perceptrons are shown to be addressed with multi-layer, nonlinear networks.

**Topic 5: Radial Basis Function Networks.** See on-line reading and text Ch 5. Radial-basis function networks are a form of functional-link net. By preprocessing sensory input (or input data patterns) using nonlinear functions, an alternative feature space is created, which often allows for classification using single-layer perceptrons. This radial-basis version of functional link networks is easier to train than generic nonlinear neural nets, at the expense of more limited extrapolation capability.

**Topic 6: Content-Addressable Memory.** See on-line notes. Scan text Ch 13. Neural networks with recurrence (feedback) can store pattern representations implicitly in the synapse weights. A feature of such networks is that an incomplete or partially flawed stimulus (initial condition) can be presented to such a network, and the network can evolve to a state that fills in or corrects the erroneous data to match a complete memory. This mechanism is proposed as a means of biological memory and associative recall.

**Topic 7: Problem Solving Using Hopfield Networks.** See on-line notes and Ch 13. Surprisingly, recurrent networks can also be used to solve planning problems. This may help explain (in part) how planning may be computed in biological brains. Hopfield’s classic paper will be analyzed and explored in a problem set.

**Topic 8: Evolutionary Programming.** See on-line notes. (Not covered in text). Genetic algorithms and evolutionary programming methods are based on biological principals. Such algorithms can perform optimization, including tuning synaptic weights for performance of a neural network. This search technique is particularly useful when it is too difficult (or impossible) to compute gradients of a system’s performance function. Simple evolutionary rules nonetheless can discover remarkably clever solutions in complex systems.

**Topic 9: Training Dynamic Networks.** See on-line reading and Ch 15. Recurrent neural networks can be used not only for their convergence characteristics, but also for their ability to generate useful timing patterns (e.g., pattern generators for coordinated limb control). However, recurrent networks can be hard to train. “Back-propagation through time” will be explored in a problem set.

**Additional topics:** Additional material will be introduced in lectures, but not explored in problem sets. Topics to be chosen from: information-theoretic learning models, neurodynamic programming and reinforcement learning, connections to neuroscience (posted readings), and the chemistry and dynamics of emotions.