Intro. to Computational Modeling

Todd Gureckis
What are computers good at?

- Doing what a programmer tells them
- Symbolic, Rule-based behavior (i.e., LISP)
What are (traditional) computers bad at?

- Dealing with noisy data
- Dealing with an unknown environment
- Massive parallel computations (**changing)
- Fault tolerance (**also changing)
- Adapting to new circumstances
- Learning from experience
Artificial Neural Networks

- Variety of names/guises: Network networks, connectionism, PDP, etc..

- Information processing systems inspired (however loosely) by biological nervous system

- Key features:
  - Large number of highly interconnected processing elements
  - Each unit operates according to its own rules
  - Interactions between neurons encoded in patterns of connectivity
  - Learning involves modifying these connections through experience
Neural inspiration

- A simplified neuron has many inputs (via dendrites) and one more output “channel” called the axon.
- The output can be either excited (i.e., firing) or not excited (i.e., resting, or deactivated).
- Incoming signals influence the degree to which any individual unit will be in either the resting or active state.
The McCulloch-Pitts model

\[ z = \sum_{i=1}^{n} w_i x_i; \quad y = H(z) \]

Mathematical version
Artificial neurons

- Nonlinear generalization of the McCulloch-Pitts neuron:
- if $z$ is the neuron's internal activation, then:

$$H(z) = \frac{1}{1 + e^{-z}}$$
What can simple things these do for us?

Adaptive interaction between individual neurons
Power: collective behavior of interconnected neurons

The hidden layer learns to recode (or to provide a representation of) the inputs: associative mapping

not unlike the properties of GoL, etc... but more directly relevant for psychologists/cog neuro people
What can simple things these do for us?

- Key features/properties of PDP that made/make them interesting:
  - Content addressable memory
  - Parallel activation/mutual constraints
  - Fault tolerance (graceful degradation in noise/trauma)
  - Distributed processing
  - Pattern completion
  - Spontaneous stimulus generalization
  - Emphasize role of learning
Interactive Activation Model
McClelland & Rumelhart (1981)
Interactive Activation Model (IAM)

- **Cascading activation**
  - Feature-level processing not complete before higher-levels start
  - Top-down and bottom-up is not viscously circular
  - Contrast to standard information processing approaches which treat perception/cognition as separate “modules”

- **Architecture**
  - Features, letter, and word level units
  - Activation between levels, inhibition within levels
  - Lateral inhibition (good for creating discrete edges, categories, decisions)

- **Processing: flow of activation/inhibition along links**
Phenomena explained by IAT

- Word superiority effect (WSE)
- Pseudo-word effect
- Rich-get-richer effect
- Gang effect
Word Superiority Effect

Subjects are more likely to choose the correct letter when it is in the context of a word than when it is isolated.

Context improves sensitivity, not just bias.
Input “alphabet”

ABC IEF GHJ
JKLMNOPQR
STU VWXYZ

010101000101010100101
Details

\[ n_i(t) = \sum_j \alpha_{ij} e_j(t) - \sum_k \gamma_{ik} i_k(t) \]

n= net input, \( \alpha \)=weight of excitatory input, e =activation of incoming excitatory node. 
\( \gamma \)=weight of inhibitory input, i = activation of incoming inhibitory node.

Net input to node is based on consistent and inconsistent inputs.
Details

\[ E_i(t) = n_i(t)(M - a_i(t)) \text{if } n_i(t) > 0 \]

\[ E_i(t) = n_i(t)(a_i(t) - m) \text{if } n_i(t) < 0 \]

- \( E_i(t) \) = effect on node \( i \), \( M \) = maximum activation possible, \( a_i \) = activation of node \( i \).
- Effect on node is based on input, but has a ceiling at \( M \). The closer the current activity of the node is to \( M \), the less the effect of positive input will be.

- \( m \) = minimum activation possible.
- If input is negative, then the floor is at \( m \). If current activity is already at floor, then input has no effect.
Details - How to make a response

\[ a_i(t + \Delta t) = a_i(t) - \theta_i(a_i(t) - r_i) + E_i(t) \]

\( \theta \) = rate of decay, \( r \) = resting level of unit
New activity is based on old activity, and decay to a resting level, and the effect of the input to the node.

\[ \bar{a}_i(t) = \int_{-\infty}^{t} a_i e^{-(t-x)r} \, dx \]

\( \bar{a} \) = running average of activation, decay of old information.
A cumulative average across time of a unit's activity will be its strength.
More recent activity levels matter more than older activity levels, and the decay rate of old information is based on \( r \).
\[ S_i(t) = e^{\mu a_i(t)} \]

\( S_i \) = response strength of unit \( i \), \( \mu \)=steepness of function relating activation to response.

Exponential functions serve to emphasize differences between larger quantities, which is important because activations are capped at 1. The difference between .8 and .9 should be greater than between .7 and .8.
Details

\[ p(R_i, t) = \frac{s_i(t)}{\sum_{j \in L} s_j(t)} \]

Response proportional to final strength
letter level activations

activation

output values

probability

E in READ

E alone

time

G89.3405 Spr. 2009
Pattern Completion

Figure 5. A hypothetical set of features that might be extracted on a trial in an experiment on word perception.
the "rich get richer" effect

Figure 11. The rich-get-richer effect. (Activation functions for the nodes for have, gave, and save under presentation of MAVE.)
Figure 12. The gang effect. (Activation functions for move, male, and save under presentation of MAVE.)
Pattern completion in non-perceptual domain

- Rumelhart, et al. (1986) developed these types of systems for schema-based processing as well using variant of IAC based on constraint satisfaction
- Classroom schema: blackboards, chalk, “fills in” new information about chairs, other objects via mutual activation and inhibition
Learning!

- A key feature of these approaches is not only the "emergent" processing characteristics of the models but the fact that the weights/interconnections between things can be learned.

- Many types of learning:
  - Unsupervised learning (Hebbian Learning, self-organizing maps)
  - Error-driven learning (backprop, reinforcement learning, etc...)

G89.3405 Spr. 2009
Kohonen Self-organizing map (S.O.M.)

- **Goals:**
  - No “teacher”, try to abstract the structure of the environment through observation
  - Dimensionality reduction (higher dimensional input space mapped into lower dimensional “feature space”)
  - Relation to the topologically ordered representations in the cortex

- **Architecture:** Input nodes connected to every neuron
  - Winning neuron is the one who’s weights are most “similar” to the input
  - “Winner-take-all” learning rule
Kohonen S.O.M.

- Similarity/distance function in weight space (should look familiar)
- Each neuron computes:
  \[ \text{act}_i = \sqrt{\sum_j (w_{ij} - i_{nj})^2} \]
- Winner has lowest value of \( \text{act}_i \) (i.e., it is closest to the input in weight space)
Kohonen S.O.M.

- Connection weights are then updating according to:
  \[ \Delta w_{ij} = \eta (in_i - w_{ij}) NWT(\alpha, i) \]

- Where NWT is a “neighborhood function” in map-space. Common functions are the mexican hat distribution (differences of two gaussians), or a pure gaussian
Kohonen S.O.M. neighborhood function

\[ NW T(\alpha, i) = e^{-\left( \frac{d(\alpha, i)}{s} \right)^2} \]
Kohonen S.O.M.

- Key properties:
  - Approximates input space
  - Topologically ordered inputs
  - Density matching (distribution of units within the maps can/may match the distribution of the input)
Localist versus Distributed Representations