CN2 1: Introduction

Paul Gribble

http://gribblelab.org

Sep 10, 2012
Class meets Mondays, 2:00pm - 3:30pm and Thursdays, 11:30am - 1:00pm, in NSC 245A

Contact me with any questions or to set up a meeting:
  ▶ paul@gribblelab.org
  ▶ Office: NSC 228 Lab: NSC 245G

Materials will be posted on the course website:
http://www.gribblelab.org/compneuro/
### Grading

<table>
<thead>
<tr>
<th>Component</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments</td>
<td>50%</td>
</tr>
<tr>
<td>Written Report</td>
<td>20%</td>
</tr>
<tr>
<td>Oral Presentation</td>
<td>20%</td>
</tr>
<tr>
<td>Participation</td>
<td>10%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>
Topics

- Modelling Dynamical Systems
- Modelling Action Potentials
- Musculoskeletal Models
- Perceptrons
- Multi-Layer Networks
- Recurrent Networks
- Stochastic Networks
- Unsupervised Learning
- Computational Motor Control
Useful Background Knowledge

- Calculus
- Linear Algebra
- Basic Probability Theory
- Differential Equations
- Computer programming

**No Fear!**

Let me know if you need a refresher on any topics, I can provide resources to help you catch up.
Readings

- Selected readings from a number of books
- Selected original research articles
We will be using **Python**: http://www.python.org/

- also the add-on library **SciPy** http://www.scipy.org/
- free
- open-source
- available for all operating systems
- widely used
What is Computational Neuroscience?

CN borrows methods and ways of thinking from

- mathematics
- physics
- computer science
- statistics
- machine learning
- psychology
- physiology
- neuroscience
Levels of Abstraction

- CNS
- System
- Maps
- Networks
- Neurons
- Synapses
- Molecules

1 m
10 cm
1 cm
1 mm
100 µm
1 µm
1 Å
Integration

- Psychology
- Neurophysiology
- Neurobiology

Experimental facts

Computational neuroscience

- Quantitative models (mathematical)
- Nonlinear dynamics
- Information theory

Experimental predictions

Applications

Psychology

Neurophysiology

Neurobiology

Refinement feedback
What is a Model?

- abstraction
  - e.g. architectural model
- concrete implementation
  - express as equations
  - simulated on a computer
  - generate quantitative predictions
- test (not confirm) hypothesis
Types of Models

Descriptive Models

phenomenological functional
description characterization of
data curve-fitting

\[ d(M) = b + k \cos \theta_{CM} \]
Types of Models

Explanatory Models

explain complex functions in terms of interactions of subordinate (simplified) components

Hodgkin-Huxley Model

\[ R_L \quad R_K \quad R_{Na} \]

\[ C \]

\[ E_L \quad E_K \quad E_{Na} \]

\[ I_{ext} \]

\[ \text{Membrane potential } V \]

\[ \text{Time [ms]} \]

\[ \text{External current } I \]

\[ \text{Firing frequency [Hz]} \]

A. B.
Serial vs Parallel Processing

- **Computers**
  - small # of very powerful CPUs
  - CPUs execute complex binary instructions
  - Gflops (billions of operations per sec)

- **Nervous Systems**
  - Many simple processing units
  - millions (billions?) of simple integrate-and-fire units
  - out of rich connectivity emerges complex computation
Parallel Distributed Processing

- artificial neural networks
- connectionist models
- interactions between units enables processing abilities not present in single units
- emergent properties
- simple rules lead to complex behaviour
- “knowledge” emerges that is not explicitly specified in system
- networks learn from experience
Neural Networks: Properties

- Generalization
- Nonlinearity
- Parallelism
- Gradedness
- Contextual Processing
- Adaptivity
- Graceful Degradation
Neural Networks: Applications

- **Aerospace**: aircraft autopilot systems
- **Banking**: OCR (e.g. cheques); credit evaluation; credit card fraud
- **Financial**: real estate appraisal; currency/equity price prediction
- **Military**: sonar object classification; image identification; who knows what else
- **Industrial**: process control; equipment failure prediction
- **Medical**: diagnosis; screening
Computing with Neurons

McCulloch & Pitts (1943)

- simple binary units
- proved: with sufficient # of small all-or-none units, and synaptic connections set to appropriate weights, a network can compute **any computable function**
Perceptron

Rosenblatt (1960)

- single-layer networks, given a **simple local learning rule** are guaranteed to converge on appropriate weights
- perceptrons compute mappings from one space to another
- many to one mappings
Multi-Layer Networks

- single-layer perceptrons are limited to problems that are *linearly separable*
- multi-layer networks given enough units and appropriate weights can compute any mapping
- learning rule called *backpropagation* allows weights to adapt
Multi-Layer Networks

Each character is a 12 x 13 matrix.
Multi-Layer Networks

12 x 13 = 156
input neurons

26 neurons
intermediate layer

26 neurons
output layer

total # synaptic weights
= (156*26)+26 + (26*26)+26
= 4,784
Unsupervised Learning

- **Supervised Learning** requires a training set of input-output examples and a detailed *teaching signal*.
- **Unsupervised Learning** approaches allow for learning about properties of the world by exposure only to inputs (exemplars).

**Hopfield Networks**

- Outputs fully connected to inputs.
- Auto-associative memory.
- Network updates dynamically over time, converges on equilibrium states.
Hopfield Network

train a network on four 5x5 patterns

each 5x5 pattern is represented by a vector of length 25

there are $25 \times 25 = 625$ weights
Hopfield Network

noisy input

ITERATION 1

ITERATION 2

ITERATION 3

ITERATION 4

ITERATION 5

ITERATION 6
Hopfield Network

partial input

ITERATION 1

ITERATION 2

ITERATION 3

ITERATION 4

ITERATION 5

ITERATION 6
Sensory-Motor Control

- in order to understand what neural control signals to muscles look like, it’s necessary to have a detailed account of the neuromuscular plant

- imagine you were studying the control inputs to a mechanical system containing springs, but you didn’t know that springs exert force in response to mechanical deformation

- what would you conclude about the control signals?
Musculoskeletal & Neuromuscular Models

For each muscle, we model the dependence of force on muscle length and active neural input. The muscle forces are generated in the following way. Each muscle, during the muscle active neural input, generates graded force development over time, passive elastic stiffness, and length and velocity dependent afferent input.

ARM MODEL

The arm model is a variant of that described by Zajac (1989), with the passive elastic stiffness of muscle (see Fig. 1). The muscle lengths and velocities associated with positive shortening, the graded development of force over time, and passive muscle stiffness. For each muscle we also include modeled neural activation and contraction dynamics and passive muscle stiffness. The muscle lengths and velocities associated with positive shortening, the graded development of force over time, and passive muscle stiffness. For each muscle we also include modeled neural activation and contraction dynamics and passive muscle stiffness.

cental command

length

rate of change of length

length reflex delay

force generating mechanism

graded force development

force velocity relationship

passive stiffness

force

FIG 1. Schematic representation of the muscle model. Model includes the dependence of force on muscle length and active neural input.