Visual Object Recognition
Computational Models and Neurophysiological Mechanisms
Neurobiology 130/230. Harvard College/GSAS 78454

Web site:  [http://tinyurl.com/visionclass](http://tinyurl.com/visionclass)
- Class notes, Class slides, Readings Assignments

Location:  Biolabs 2062

Time:  Mondays 03:00 – 05:00

Lectures:
Faculty:  Gabriel Kreiman and invited guests
TA:  Emma Giles

Contact information:
Gabriel Kreiman  Emma Giles
gabriel.kreiman@tch.harvard.edu  emmagiles@g.harvard.edu
617-919-2530
Office Hours: After Class. Mondays 5pm, or by appointment
Class 1 [09/10/2018]. Introduction to pattern recognition [Kreiman]
Class 2 [09/17/2018]. Why is vision difficult? Natural image statistics. The retina. [Kreiman]
Class 3 [09/24/2018]. Lesions and neurological studies [Kreiman].
Class 4 [10/01/2018]. Psychophysics of visual object recognition [Sarit Szpiro]

October 8: University Holiday

Class 5 [10/15/2018]. Primary visual cortex [Hartmann]
Class 6 [10/22/2018]. Adventures into terra incognita [Frederico Azevedo]
Class 7 [10/29/2018]. High-level visual cognition [Diego Mendoza-Haliday]
Class 8 [11/05/2018]. Correlation and causality. Electrical stimulation in visual cortex [Kreiman]

Class 12 [12/03/2018]. The operating system for vision. [Xavier Boix]

Class 1. Introduction to pattern recognition [Kreiman]
Class 3. Lesion and neurological studies of visual deficits in animals and humans. [Kreiman]
Class 4. Psychophysics of visual object recognition [Jiye Kim]
October 9: University Holiday
Class 5. Introduction to the thalamus and primary visual cortex [Camille Gomez-Laberge]
Class 6. Adventures into terra incognita. Neurophysiology beyond V1 [Frederico Azevedo]
Class 7. First steps into inferior temporal cortex [Carlos Ponce]
Class 8. From the highest echelons of visual processing to cognition [Leyla Isik]
Class 9. Correlation and causality. Electrical stimulation in visual cortex [Kreiman].

Class 11. Computer vision. Towards artificial intelligence systems for cognition [Bill Lotter]
Class 12. Vision and Language. [Andrei Barbu]

FINAL EXAM
1. Why build computational models?
2. Single neuron models
3. Network models
4. Algorithms and methods for data analysis
Why bother with computational models?

- Quantitative models force us to think about and formalize hypotheses and assumptions

- Models can integrate and summarize observations across experiments, resolutions and laboratories

- A good model can lead to (non-intuitive) experimental predictions

- A quantitative model, implemented through simulations, can be useful from an engineering viewpoint (e.g. face recognition)

- A model can point to important missing data, critical information and decisive experiments
What is a model, anyway?

Which hand was the person using?
What is the shape/color/material of the object?
What day of the week is it?
What type of surface is it?
What is the temperature/humidity?
What is the force exerted by the person?
What is the weight of the object?
What is the force of gravity on this object?
Where is the force exerted?
What is the person wearing?
How much contact is there between the object and the surface?

\[ F = m \ a \]
A model for orientation tuning in simple cells

A feed-forward model for orientation selectivity in V1
(by no means the only model)

Text-fig. 19. Possible scheme for explaining the organization of simple receptive fields. A large number of lateral geniculate cells, of which four are illustrated in the upper right in the figure, have receptive fields with ‘on’ centres arranged along a straight line on the retina. All of these project upon a single cortical cell, and the

Text-fig. 20. Possible scheme for explaining the organization of complex receptive fields. A number of cells with simple fields, of which three are shown schematically, are imagined to project to a single cortical cell of higher order. Each projecting neuron has a receptive field arranged as shown to the left: an excitatory region to the left and an inhibitory region to the right of a vertical straight-line boundary. The boundaries of the fields are staggered within an area outlined by the interrupted lines. Any vertical-edge stimulus falling across this rectangle, regardless of its position, will excite some simple-field cells, leading to excitation of the higher-order cell.

Hubel and Wiesel. J. Physiology (1962)
1. Why build computational models?
2. Single neuron models
3. Network models
4. Algorithms and methods for data analysis
A nested family of single neuron models

- Filter operations
- Integrate-and-fire circuit
- Hodgkin-Huxley units
- Multi-compartmental models
- Spines, channels

- Biological accuracy
- Lack of analytical solutions
- Computational complexity
Geometrically accurate models vs. spherical cows with point masses

A central question in Theoretical Neuroscience:
What is the “right” level of abstraction?
The leaky integrate-and-fire model

- Lapicque 1907
- Below threshold, the voltage is governed by:
  \[ C \frac{dV(t)}{dt} = -\frac{V(t)}{R} + I(t) \]
- A spike is fired when \( V(t) > V_{\text{thr}} \) (and \( V(t) \) is reset)
- A refractory period \( t_{\text{ref}} \) is imposed after a spike.
- Simple and fast.
- Does not consider spike-rate adaptation, multiple compartments, sub-ms biophysics, neuronal geometry
The leaky integrate-and-fire model

- Lapicque 1907
- Below threshold, the voltage is governed by:
  \[ C \frac{dV(t)}{dt} = -\frac{V(t)}{R} + I(t) \]
- A spike is fired when \( V(t) > V_{\text{thr}} \) (and \( V(t) \) is reset)
- A refractory period \( t_{\text{ref}} \) is imposed after a spike
- Simple and fast
- Does not consider:
  - spike-rate adaptation
  - multiple compartments
  - sub-ms biophysics
  - neuronal geometry

```matlab
function [V,spk]=simpleiandf(E_L,V_res,V_th,tau_m,R_m,I_e,dt,n)
  % ultra-simple implementation of integrate-and-fire model
  % inputs:
  % E_L = leak potential [e.g. -65 mV]
  % V_res = reset potential [e.g. E_L]
  % V_th = threshold potential [e.g. -50 mV]
  % tau_m = membrane time constant [e.g. 10 ms]
  % R_m = membrane resistance [e.g. 10 MOhm]
  % I_e = external input [e.g. white noise]
  % dt = time step [e.g. 0.1 ms]
  % n = number of time points [e.g. 1000]
  %
  % returns
  % V = intracellular voltage [n x 1]
  % spk = 0 or 1 indicating spikes [n x 1]
  V(1)=V_res; % initial voltage
  spk=zeros(n,1);
  for t=2:n
    V(t)=V(t-1)+(dt/tau_m) * (E_L - V(t-1) + R_m * I_e(t)); % Key line computing the change in voltage at time t
    if (V(t)>V_th) % Emit a spike if V is above threshold
      V(t)=V_res;
      % And reset the voltage
      spk(t)=1;
    end
  end
```
function [V,spk]=simpleiandf(E_L,V_res,V_th,tau_m,R_m,I_e,dt,n)

% ultra-simple implementation of integrate-and-fire model
% inputs:
% E_L    = leak potential           [e.g. -65 mV]
% V_res  = reset potential          [e.g. E_L]
% V_th   = threshold potential      [e.g. -50 mV]
% tau_m  = membrane time constant   [e.g. 10 ms]
% R_m    = membrane resistance      [e.g. 10 MOhm]
% I_e    = external input           [e.g. white noise]
% dt     = time step                [e.g. 0.1 ms]
% n      = number of time points    [e.g. 1000]
%
% outputs:
% V      = intracellular voltage    [n x 1]
% spk    = 0 or 1 indicating spikes [n x 1]

V(1)=V_res;       % initial voltage
spk=zeros(n,1);
for t=2:n
    V(t)=V(t-1)+(dt/tau_m) * (E_L - V(t-1) + R_m * I_e(t));  % Change in voltage at time t
    if (V(t)>V_th)                                           % Emit a spike if V is above threshold
        V(t)=V_res;                                          % And reset the voltage
        spk(t)=1;
    end
end

end

C \frac{dV(t)}{dt} = - \frac{V(t)}{R} + I(t)
The Hodgkin-Huxley Model

\[ I(t) = C \frac{dV}{dt} + g_L (V - E_L) + g_K n^4 (V - E_K) + g_{Na} m^3 h (V - E_{Na}) \]

where:
- \( i_m \) = membrane current
- \( V \) = voltage
- \( L \) = leak channel
- \( K \) = potassium channel
- \( Na \) = sodium channel

\( g \) = conductances (e.g. \( g_{Na} \)=120 mS/cm\(^2\); \( g_K \)=36 mS/cm\(^2\); \( g_L \)=0.3 mS/cm\(^2\))
\( E \) = reversal potentials (e.g. \( E_{Na} \)=115mV, \( E_K \)=12 mV, \( E_L \)=10.6 mV)
\( n, m, h \) = “gating variables”, \( n=n(t), m=m(t), h=h(t) \)

OUTLINE

1. Why build computational models?
2. Single neuron models
3. Network models
4. Algorithms and methods for data analysis
From neurons to circuits

• Single neurons can perform many interesting and important computations (e.g. Gabbiani et al (2002). Multiplicative computation in a visual neuron sensitive to looming. Nature 420, 320-324)

• Neurons are not isolated. They are part of circuits. A typical cortical neuron receives input from $\sim10^4$ other neurons.

• It is not always trivial to predict circuit-level properties from single neuron properties. There could be interesting properties emerging at the network level.
Circuits – some basic definitions

Notes:
1. Connectivity does not need to be all-to-all
2. There are excitatory neurons and inhibitory neurons (and many types of inhibitory neurons)
3. Most models assume balance between excitation and inhibition
4. Most models do not include layers and the anatomical separation of forward and back pathways
5. There are many more recurrent+feedback connections than feed-forward connections (the opposite is true about models...)
Firing rate network models – A simple feedforward circuit

- Time scales $> \sim 1$ ms
- Analytic calculations in some cases
- Fewer free parameters than spiking models
- Easier/faster to simulate

\[
I_s = \sum_{b=1}^{N} w_b \int_{-\infty}^{t} d\tau K_s(t-\tau) u_b(\tau)
\]

\[
\tau_s \frac{dI_s}{dt} = -I_s + \sum_{b=1}^{N} w_b u_b
\]

\[
v = F(I_s)
\]

\[
f can be a sigmoid function
Or a threshold linear function:
F(I_s) = [I_s - \gamma]_+
\]
Imagine that we want to classify the inputs $u$ into two groups “+1” and “-1”

$$v = \begin{cases} 
+1 & \text{if } w.u - \gamma \geq 0 \\
-1 & \text{if } w.u - \gamma < 0 
\end{cases}$$

Training examples: $\{u_m, v_m\}$

$$w \rightarrow w + \frac{\epsilon}{2}(v_m - v(u_m))u_m$$

*Perceptron* learning rule

Linear separability: can attain zero error
Cross-validation: use separate training and test data
There are several more sophisticated learning algorithms
Learning from examples – Gradient descent

Now imagine that $v$ is a real value (as opposed to binary)

$$u = f(s)$$

$$v(s) = w \cdot u$$

We want to choose the weights so that the output approximates some function $h(s)$

$$E = \frac{1}{2N_s} \sum_{m=1}^{N_s} (h(s^m) - v(s^m))^2$$

$$w \rightarrow w + \epsilon \nabla_w E$$

$$\nabla_w E = \begin{bmatrix} \frac{\partial E}{\partial w_b} \end{bmatrix}$$
Example: digit recognition in a feed-forward network trained by gradient descent

Example of handwritten digits (MINT database)

Classification error rates

Misclassified examples

The “blue brain” modeling project

- http://bluebrain.epfl.ch
- IBM’s Blue gene supercomputer
- “Reverse engineer” the brain in a “biologically accurate” way
- November 2007 milestone: 30 million synapses in “precise” locations to model a neocortical column
- Compartmental simulations for neurons
- Needs another supercomputer for visualization (10,000 neurons, high quality mesh, 1 billion triangles, 100 Gb)

QUESTION: What is the “right” level of abstraction needed to understand the function of cortical circuitry?
A case study in collective computation

Primer on Hopfield networks
1. Why theoretical neuroscience?
2. Single neuron models
3. Network models
4. Algorithms and methods for data analysis
Some examples of computational algorithms and methods

• Different techniques for time-frequency analysis of neural signals (e.g. Pesaran et al 2002, Fries et al 2001)

• Spike sorting (e.g. Lewicki 1998)

• Machine learning approaches to decoding neuronal responses (e.g. Hung et al 2005, Wilson et al 1993, Musallam et al 2004)

• Information theory (e.g. Abbott et al 1996, Bialek et al 1991)

• Neural coding (e.g. Gabbiani et al 1998, Bialek et al 1991)

• Definition of spatio-temporal receptive fields, phenomenological models, measures of neuronal synchrony, spike train statistics
Further reading