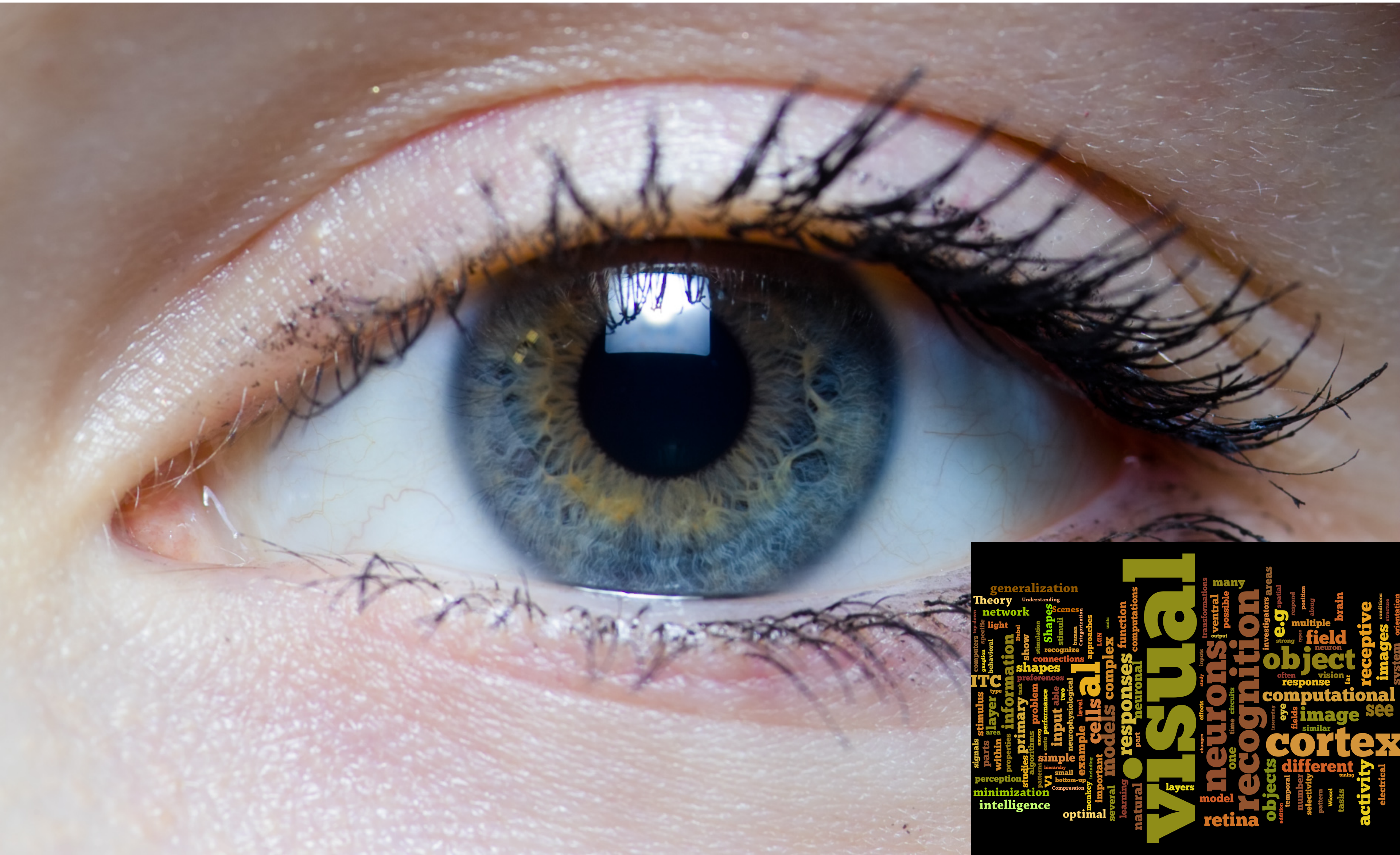
Visual Object Recognition
Computational Models and Neurophysiological Mechanisms

Neuro 130/230. Harvard College/GSAS 78454



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

Class 1 [09/02/2020]. Introduction to Vision
Class 2 [09/14/2020]. Natural image statistics and the retina
Class 3 [09/21/2020]. The Phenomenology of Vision
Class 4 [09/28/2020]. Learning from Lesions
Class 5 [10/05/2020]. Primary Visual Cortex
October 12th: University Holiday
Class 6 [10/19/2020]. Adventures into *terra incognita*
Class 7 [10/26/2020]. From the Highest Echelons of Visual Processing to Cognition
Class 8 [11/02/2020]. First Steps into in silico vision
Class 9 [11/09/2020]. Teaching Computers how to see
Class 10 [11/16/2020]. Computer Vision
Class 11 [11/23/2020]. Connecting Vision to the rest of Cognition
Class 12 [11/30/2020]. Visual Consciousness
FINAL EXAM, PAPER DUE 12/14/2020. No extensions.

OUTLINE

1. **Why build computational models?**
2. Single neuron models
3. Network models
4. Algorithms and methods for data analysis

Why bother with computational models?

"Verbal models" are not real models:
- Vague and prone to subjective interpretation
- Lack of quantitative predictions
- Not falsifiable

-Quantitative models force us to formalize hypotheses and assumptions

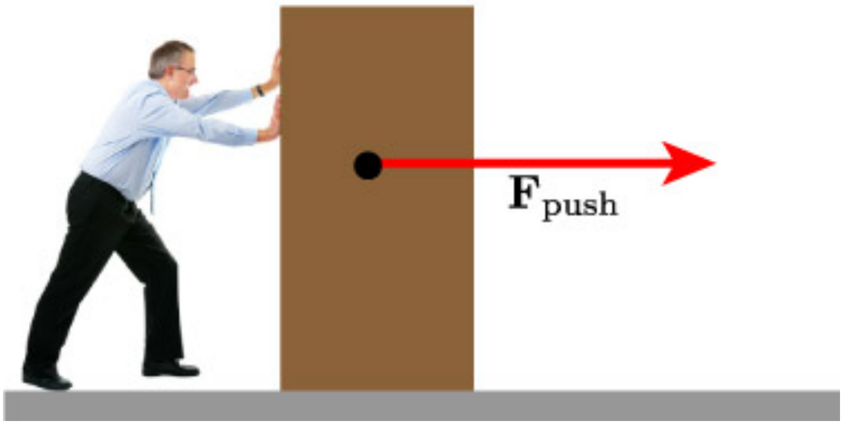
-Models can integrate observations across experiments, resolutions and laboratories

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

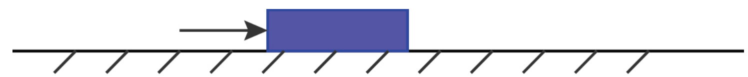
-A model can point to missing data, critical information and decisive experiments

-A model can be useful from an engineering viewpoint (e.g. face recognition)

What is a model, anyway?

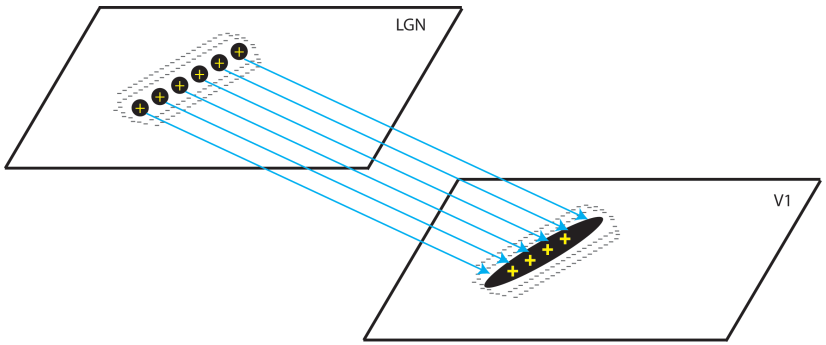


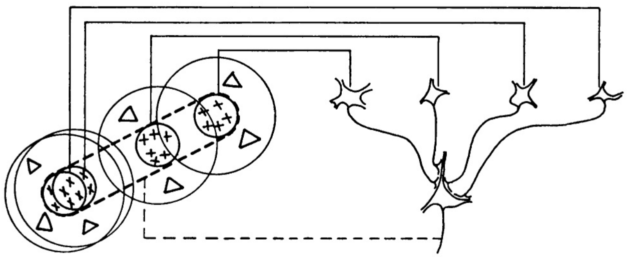
$F = m\,a$

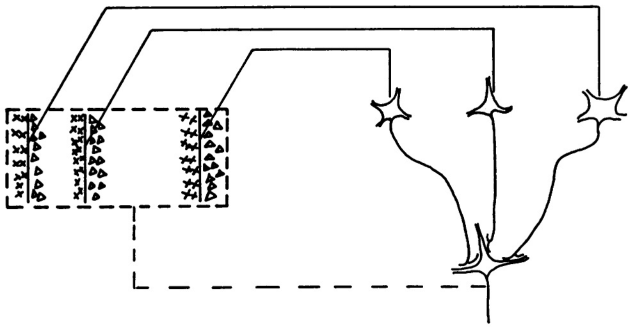


- Which hand was the person using?
- What is the shape/color/material of the object?
- What day of the week is it?
- What type of surface is it?
- What is the temperature/humidity?
- What is the force exerted by the person?
- What is the weight of the object?
- What is the force of gravity on this object?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

A model for orientation tuning in simple cells







A feed-forward model for orientatior selectivity in V1

(by no means the only model)

Hubel and Wiesel. J. Physiology (1962)

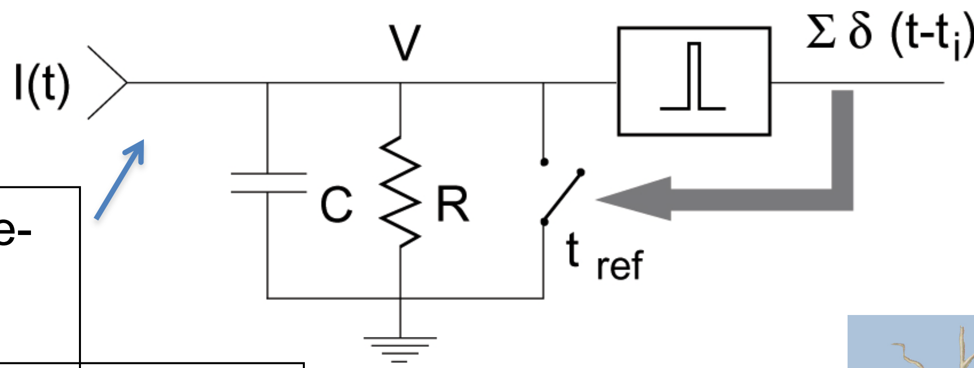
OUTLINE

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

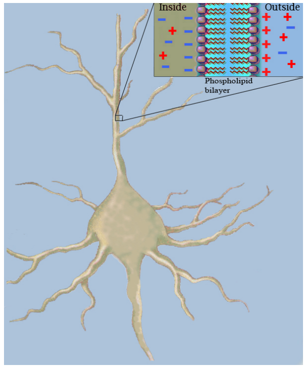
Filter operations

Integrate-and-fire circuit



Hodgkin-Huxley units

Multi-compartmental models



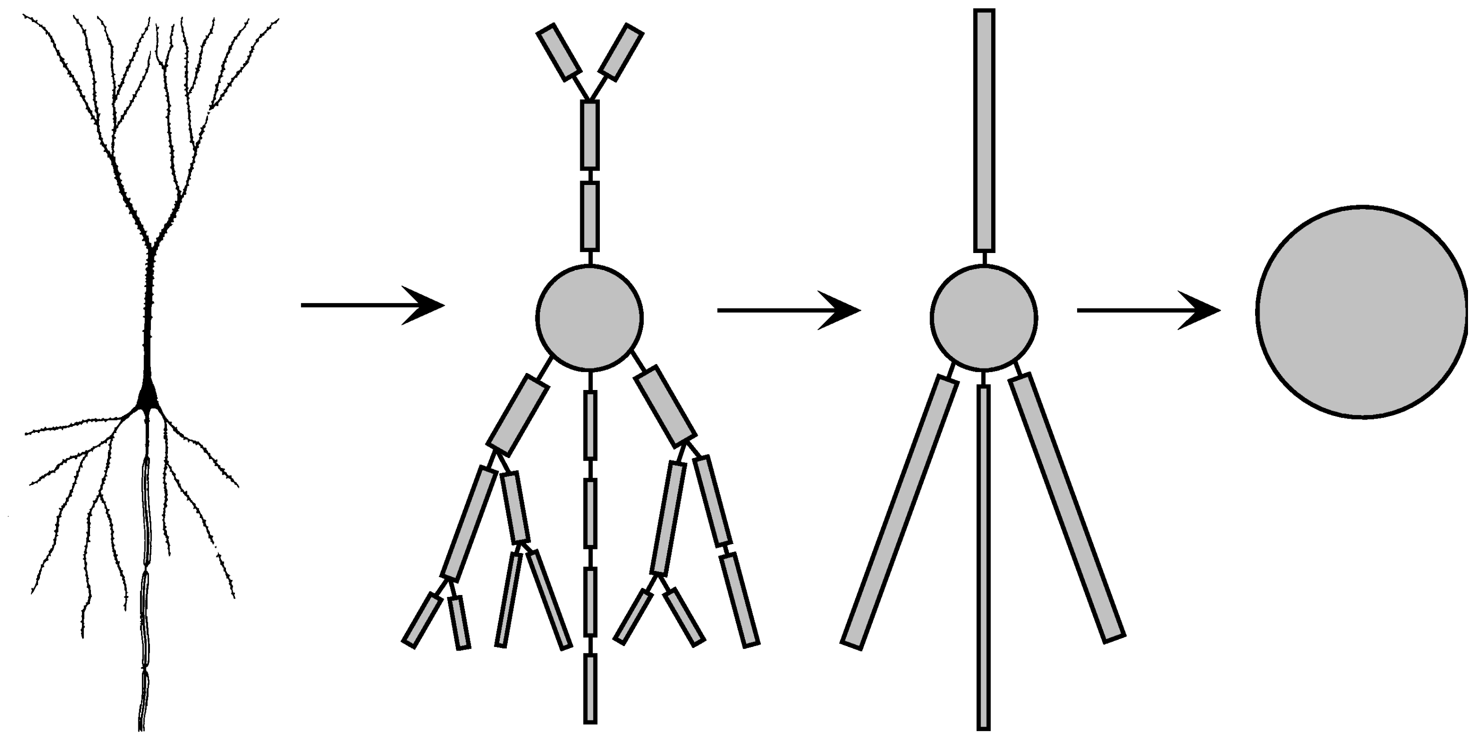
Spines, channels

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 Hodgkin-Huxley Model

$$I(t) = C\frac{dV}{dt} + \bar{g}_L(V - E_L) + \bar{g}_K n^4(V - E_K) + \bar{g}_{Na} m^3 h(V - E_{Na})$$

where:
i_m = membrane current
V = voltage

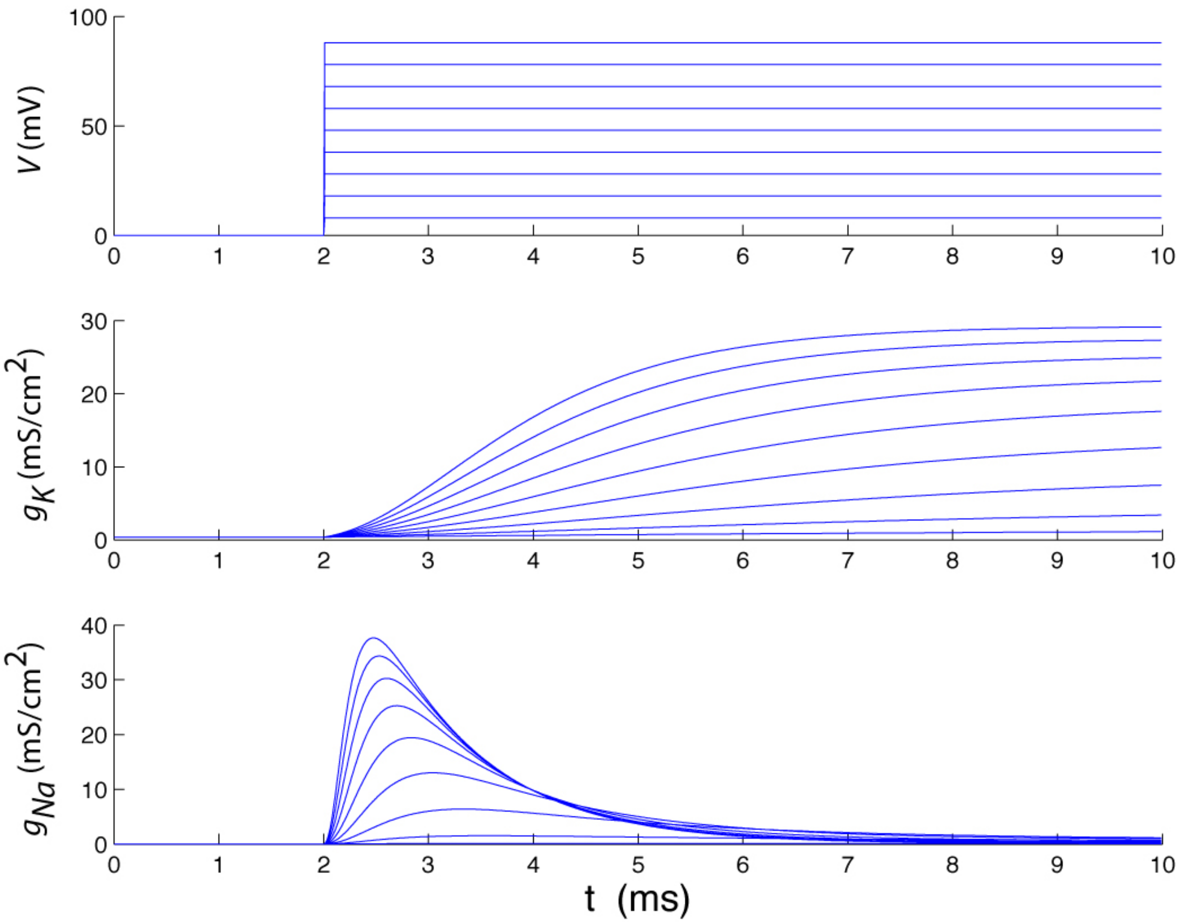
L = leak channel
K = potassium channel
Na = sodium channel

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", $n=n(t)$, $m=m(t)$, $h=h(t)$

Hodgkin, A. L., and Huxley, A. F. (1952).
A quantitative description of membrane current and its application to conduction and excitation in nerve.
Journal of Physiology *117*, 500-544.

The Hodgkin-Huxley Model

```
% Gabbiani & Cox, Mathematics for Neuroscientists
% clamp.m
% Simulate a voltage clamp experiment
% usage:  clamp(dt,Tfin)
% e.g. clamp(.01,15)

function clamp(dt,Tfin)
vK = -6;     % mV
GK =  36;    % mS/(cm)^2
vNa = 127;   % mV
GNa = 120;   % mS/(cm)^2
for vc = 8:10:90,
    j = 2;t(1) = 0;v(1) = 0;
    n(1) = an(0)/(an(0)+bn(0));   % 0.3177;
    m(1) = am(0)/(am(0)+bm(0));   % 0.0529;
    h(1) = ah(0)/(ah(0)+bh(0));   % 0.5961;
    gK(1) = GK*n(1)^4;
    gNa(1) = GNa*m(1)^3*h(1);
    while j*dt < Tfin,
        t(j) = j*dt;
        v(j) = vc*(t(j)>2)*(t(j)<Tfin);
        n(j) = ( n(j-1) + dt*an(v(j)) )/(1 + dt*(an(v(
);
        m(j) = ( m(j-1) + dt*am(v(j)) )/(1 + dt*(am(v(
);
        h(j) = ( h(j-1) + dt*ah(v(j)) )/(1 + dt*(ah(v(
);
        gK(j) = GK*n(j)^4;
        gNa(j) = GNa*m(j)^3*h(j);
        j = j + 1;
    end
    subplot(3,1,1); plot(t,v); hold on
    subplot(3,1,2); plot(t,gK); hold on
    subplot(3,1,3); plot(t,gNa); hold on
end
 subplot(3,1,1);ylabel('v','fontsize',16);hold off
subplot(3,1,2);ylabel('g_K','fontsize',16);hold off
subplot(3,1,3);xlabel('t
(ms)','fontsize',16);ylabel('g_{Na}','fontsize',16);hold off

function val = an(v)
val = .01*(10-v)./(exp(1-v/10)-1);
function val = bn(v)
```



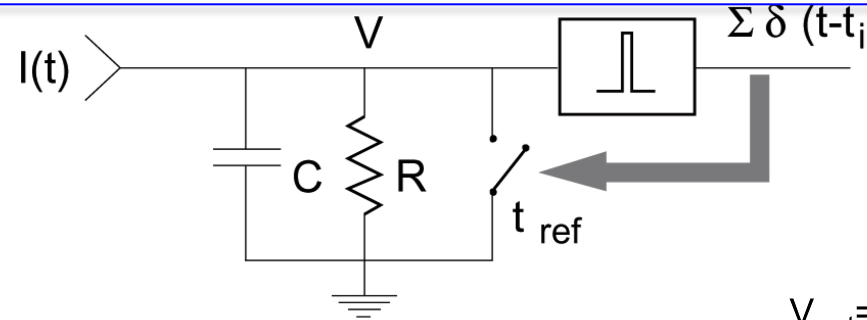
Simulated voltage-clamp experiments of Hodgkin and Huxley (1952). From Gabbiani and Cox 2010.

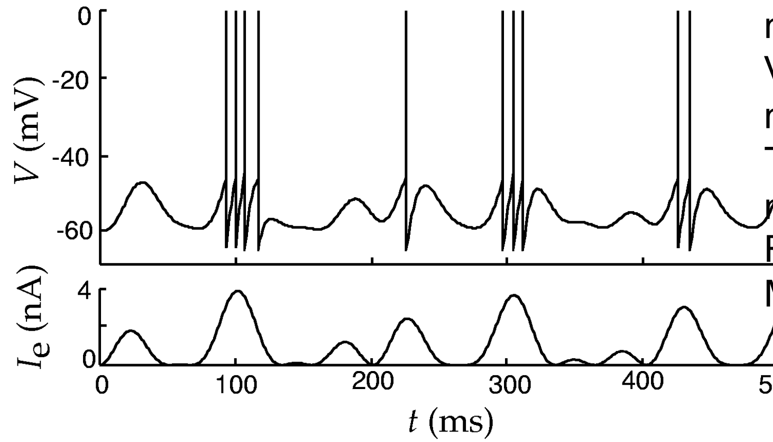
The leaky integrate-and-fire model

- Lapicque 1907
- Below threshold, the voltage is governed by:

$$C\frac{dV(t)}{dt} = -\frac{V(t)}{R} + I(t)$$

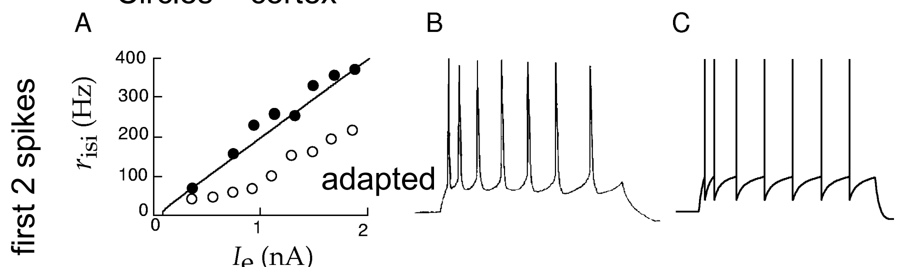
- A spike is fired when $V(t) > V_{thr}$ (and $V(t)$ is reset)
- A refractory period t_{ref} is imposed after a spike.
- Simple and fast.
- Does not consider spike-rate adaptation, multiple compartments, sub-ms biophysics, neuronal geometry





V_{rest}=-65 mV
V_{th} =-50 mV
T_m = 10 ms
R_m = 10 MΩ

Line = I&F model
Circles = cortex

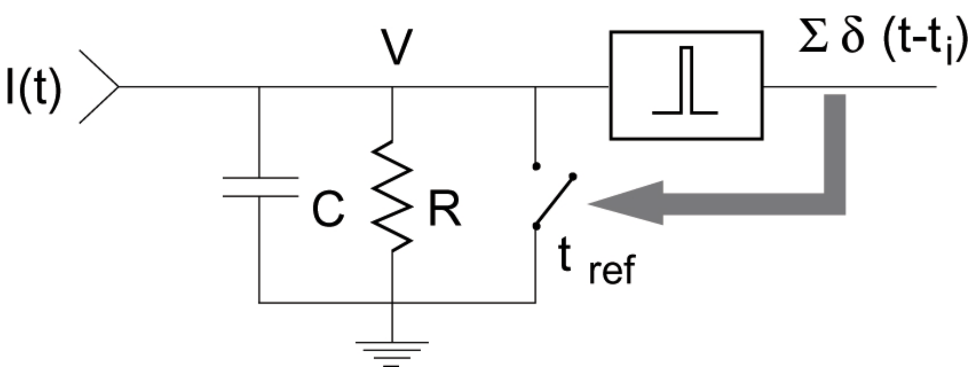


The leaky integrate-and-fire model

- 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 $V(t)>V_{thr}$ (and $V(t)$ is reset)
- A refractory period t_{ref} is imposed after a spike
- Simple and fast
- Does not consider:
  - spike-rate adaptation
  - multiple compartments
  - sub-ms biophysics
  - neuronal geometry




```
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
```

Interlude: MATLAB is easy

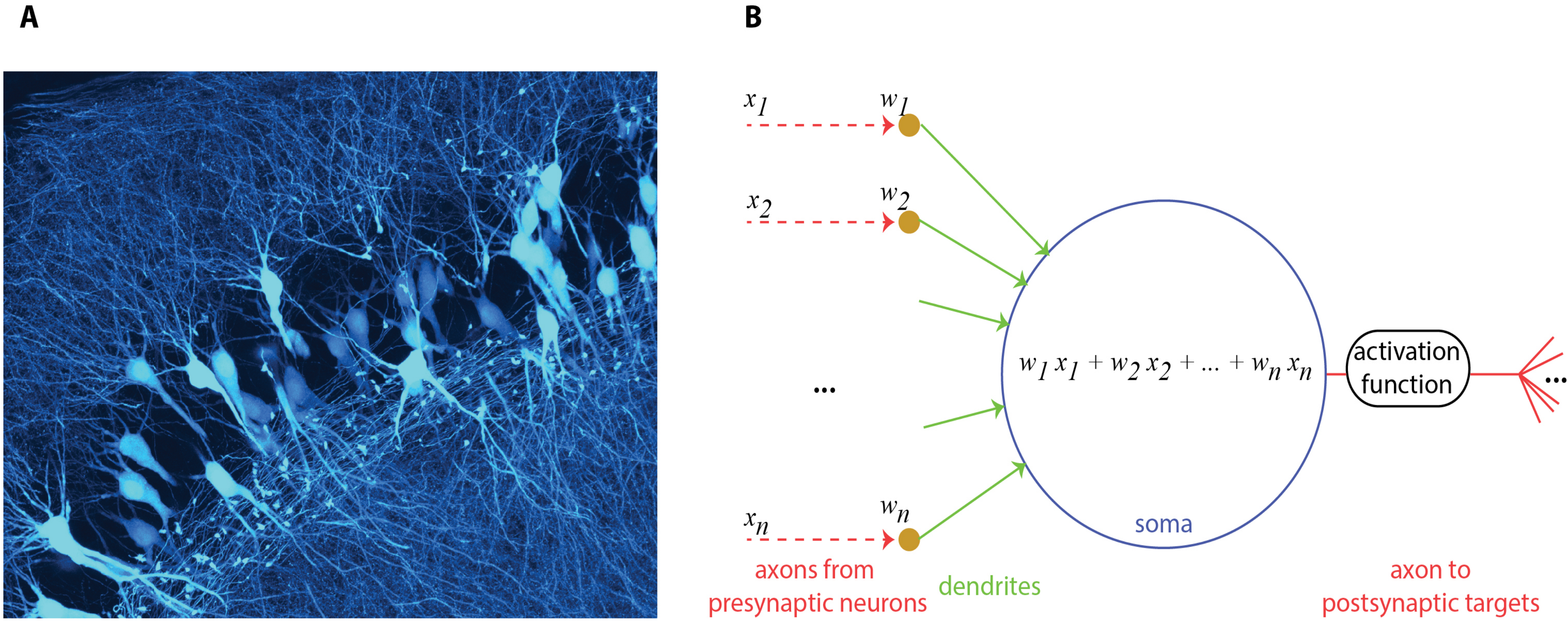
$$C\frac{dV(t)}{dt} = -\frac{V(t)}{R} + I(t)$$

```
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
```

All of these lines are comments

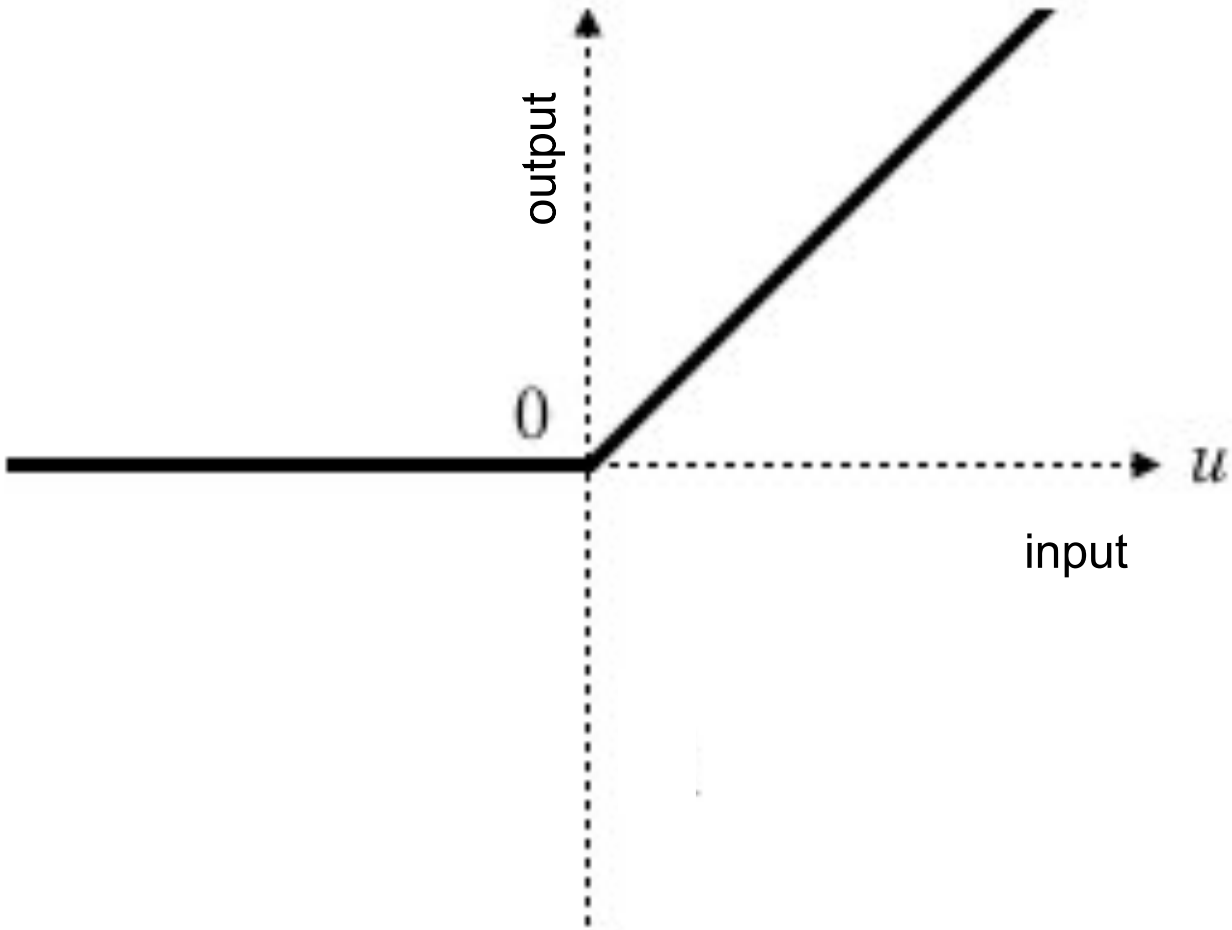
This is the key line integrating the differential equation

Typical units in neural networks



ReLU

$$f(u) = \max(0, u)$$



OUTLINE

1. Why build computational models?
2. Single neuron models
3. **Network models**
4. Algorithms and methods for data analysis

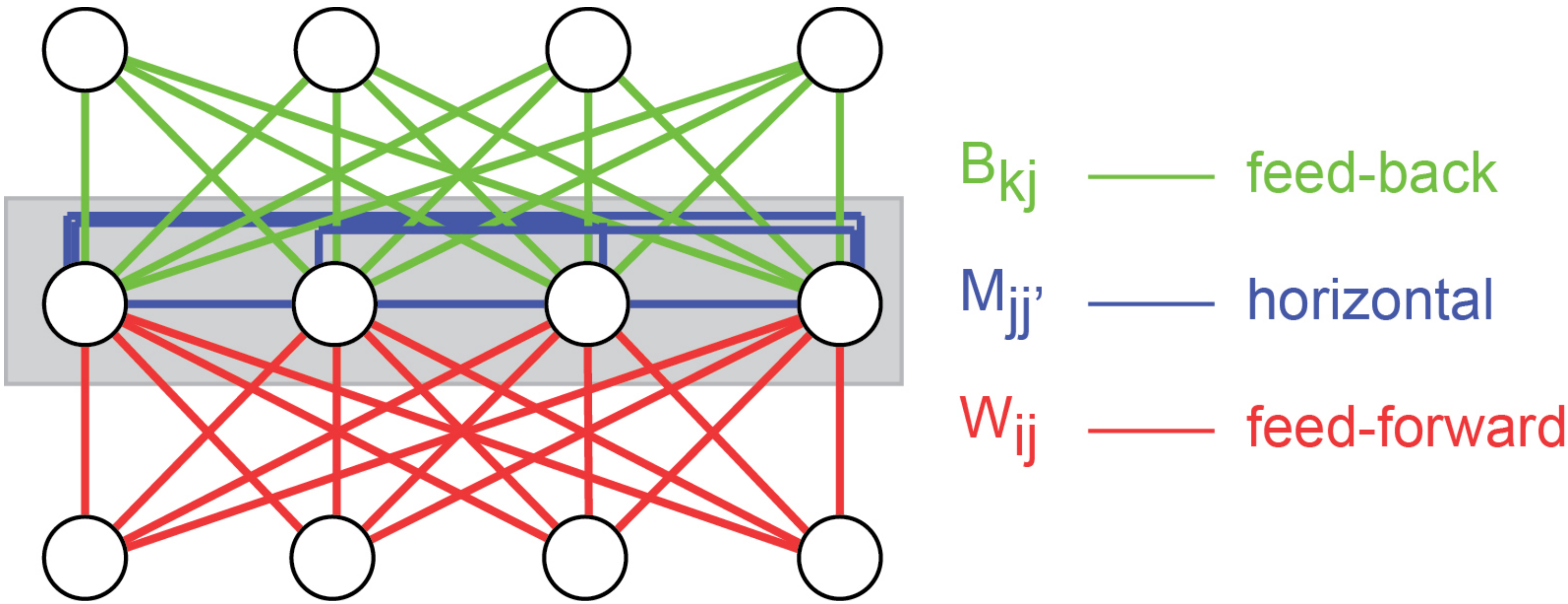
From neurons to circuits

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

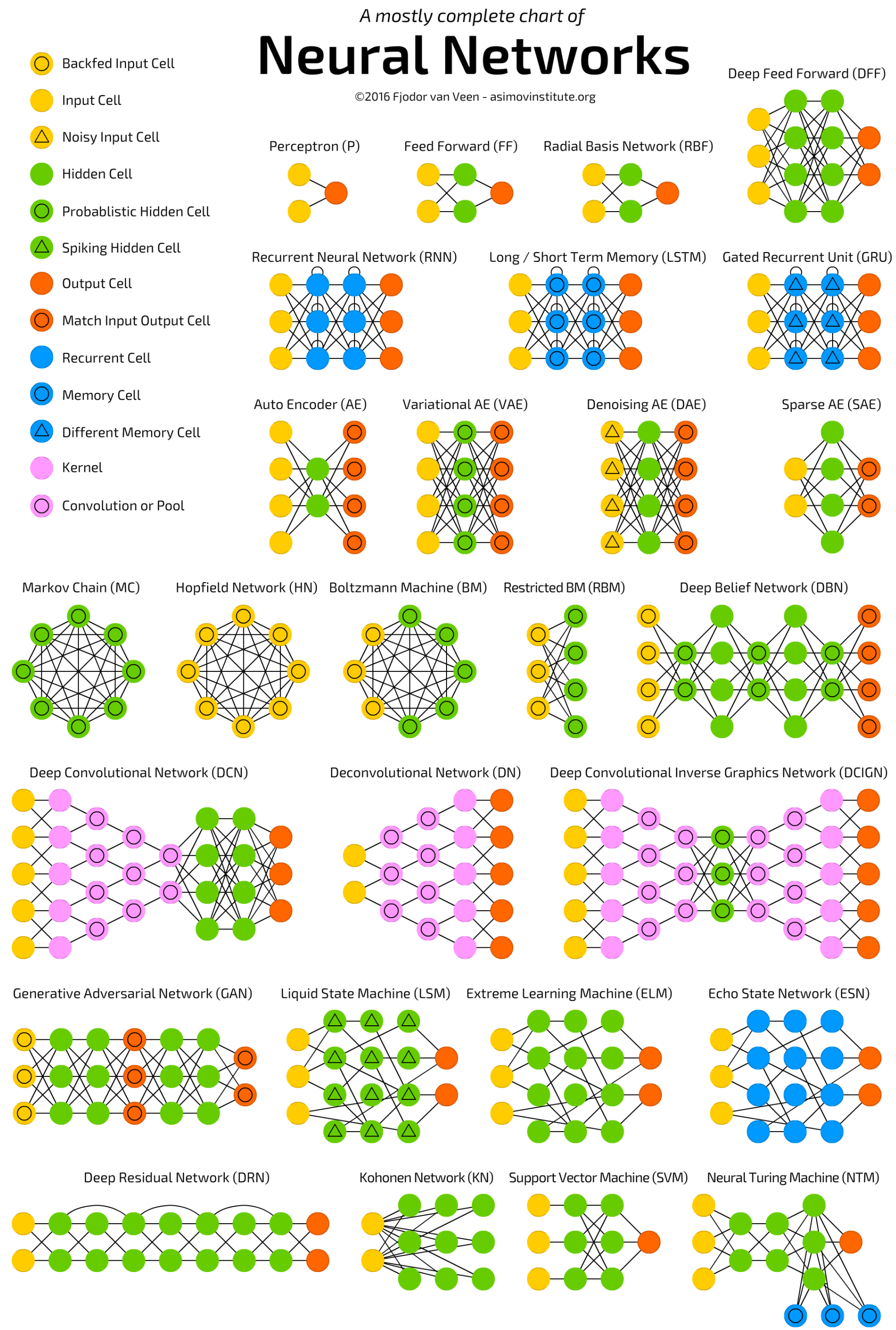
•But neurons are not isolated. They are part of circuits. A typical cortical neuron receives input from $\sim 10^4$ other neurons.

•It is not trivial to predict circuit-level properties from single neuron properties. There can be interesting properties emerging at the network level.

Circuits – some basic definitions



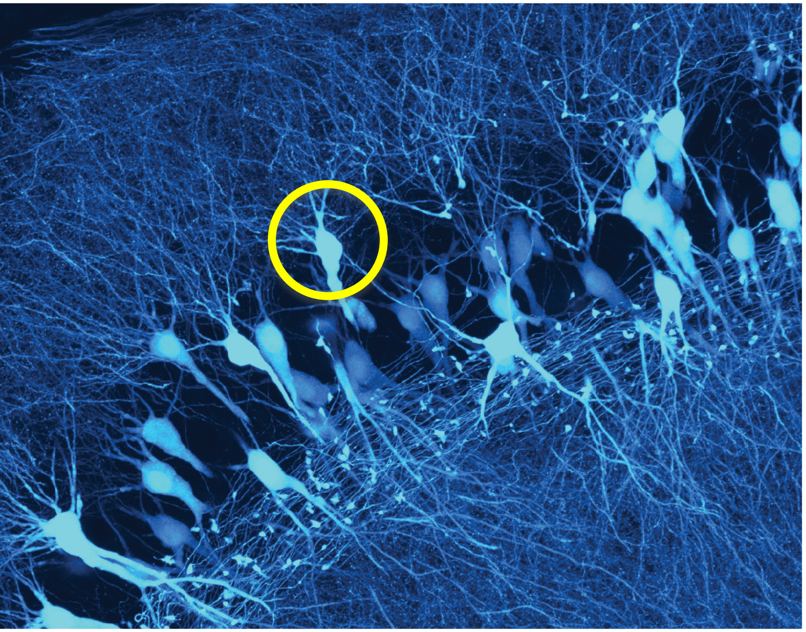
A big happy family of neural networks



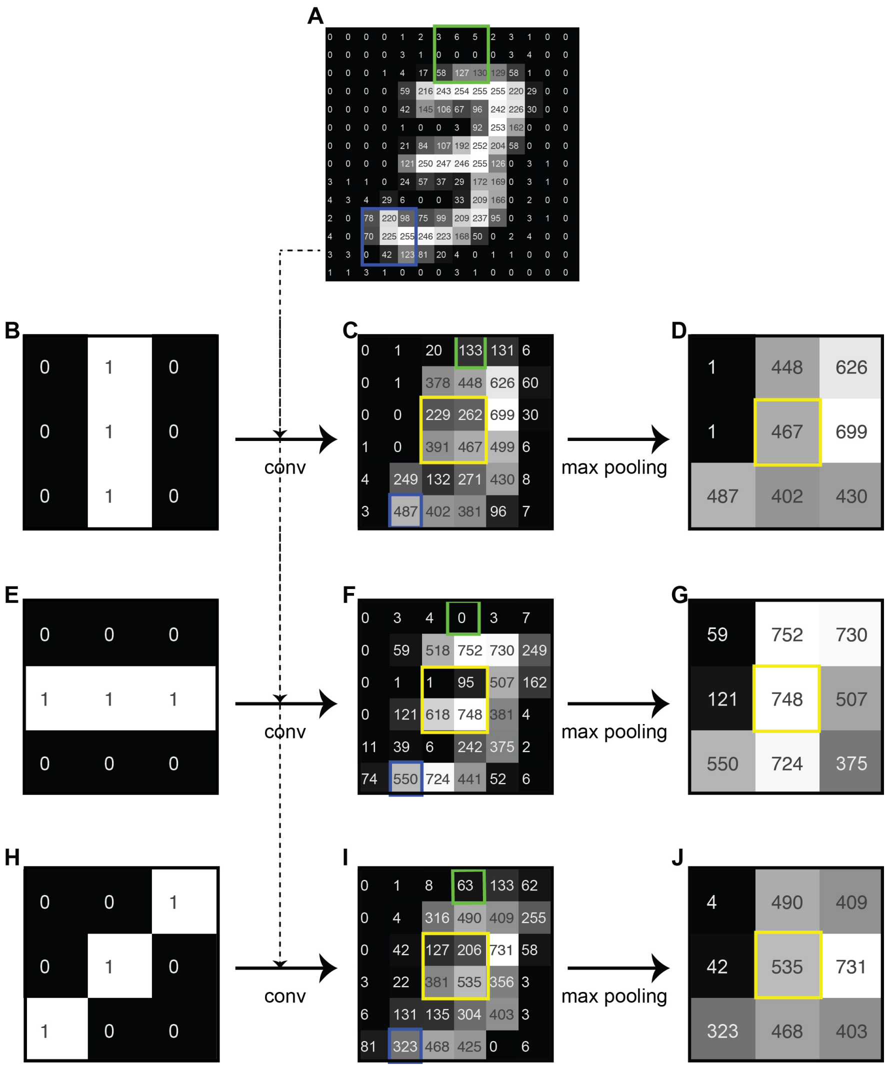
https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

From neural circuits to neural networks

A

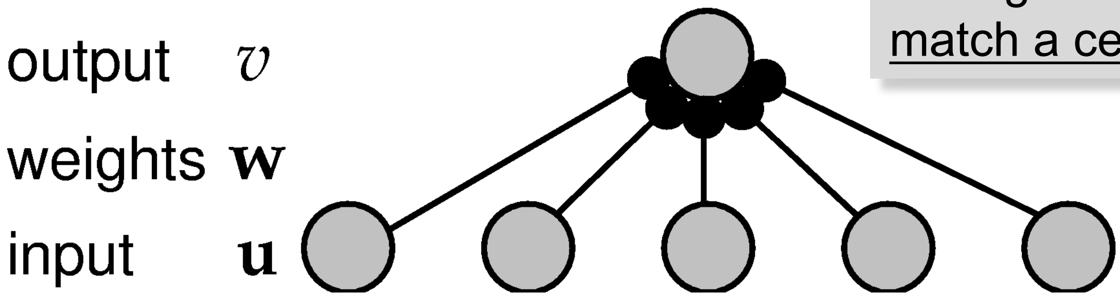


The convolution operation



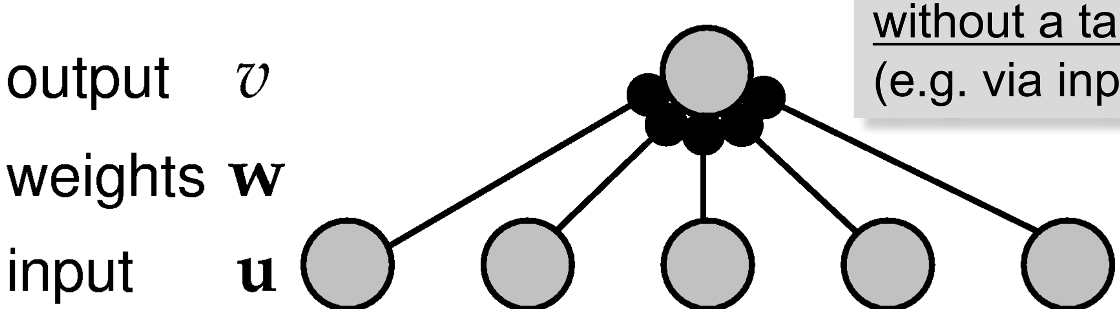
Supervised versus unsupervised learning

Supervised learning



Change the weights $\mathbf{w}$ to <u>match a certain output v</u>

Unsupervised learning



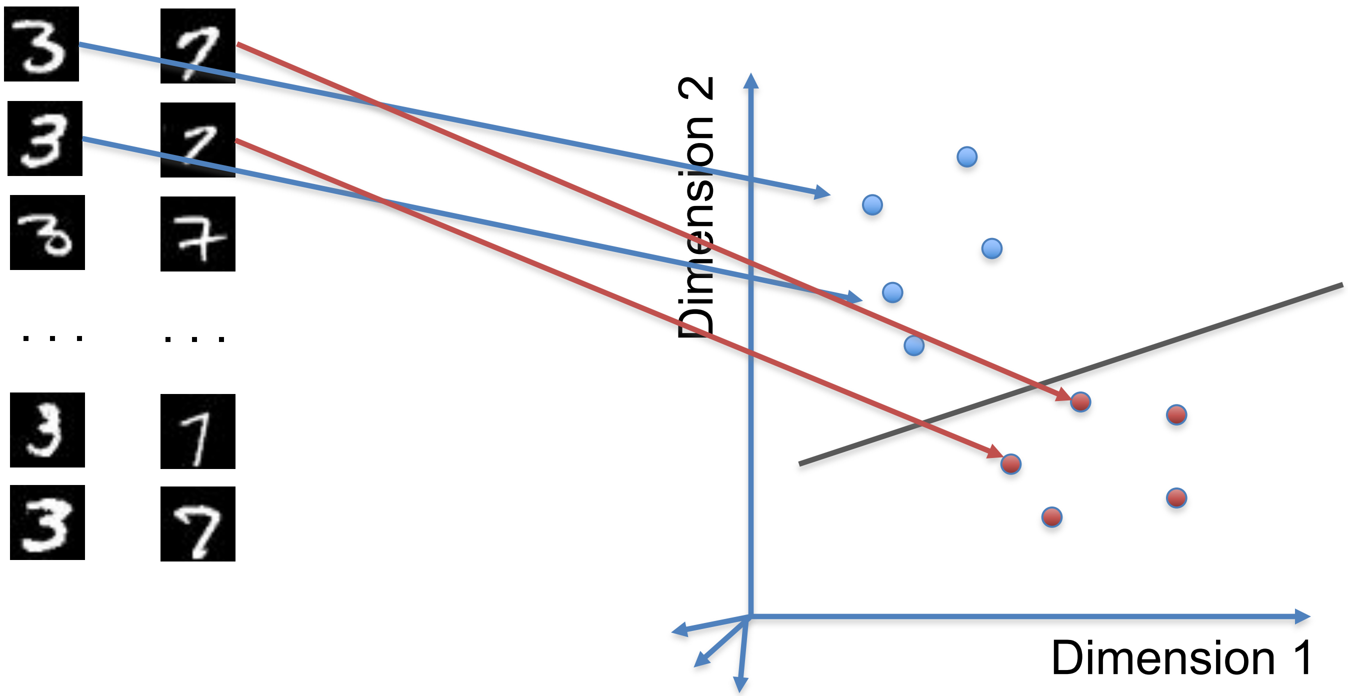
Change the weights $\mathbf{w}$ <u>without a target output</u> (e.g. via input correlations)

Learning from examples – Digit classification

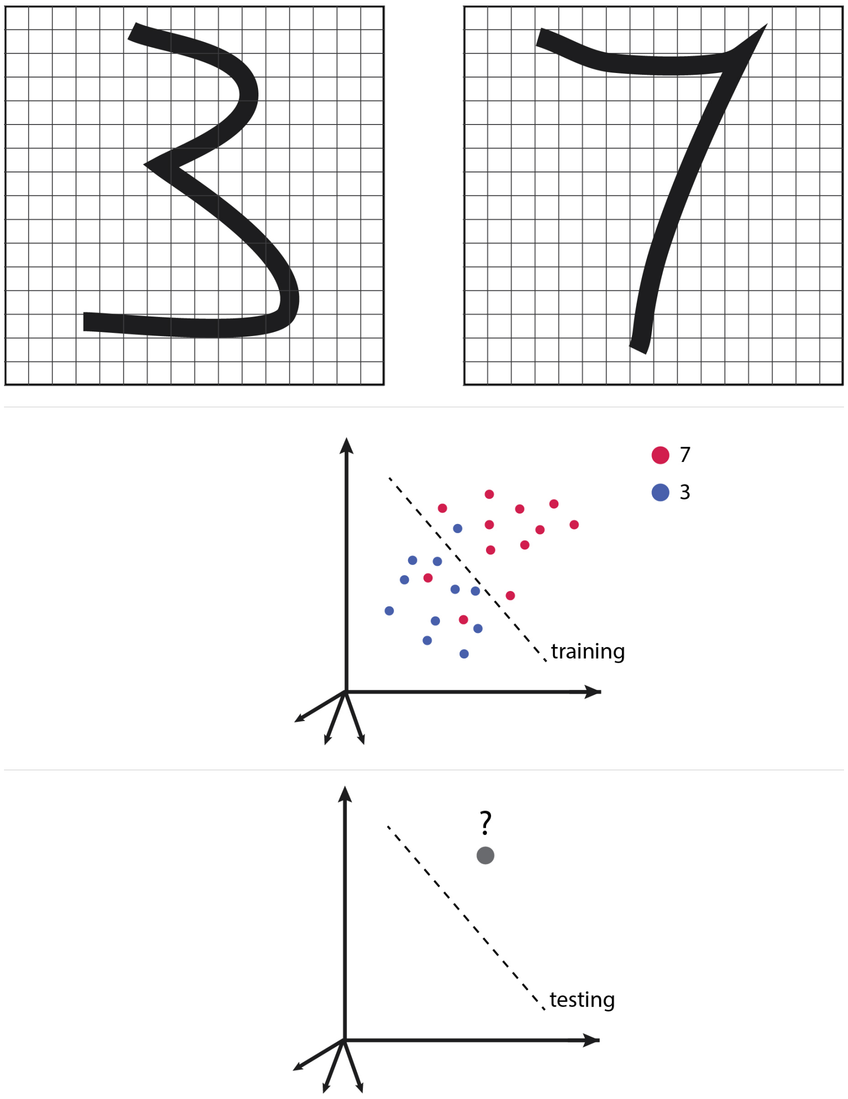
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	115	121	162	253	253	213	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	63	107	170	251	252	252	252	252	250	214	0	0	0	0	0	0
0	0	0	0	0	0	0	0	25	192	226	226	241	252	253	202	252	252	252	252	252	225	0	0	0	0	0	0
0	0	0	0	0	0	0	68	223	252	252	252	252	252	39	19	39	65	224	252	252	183	0	0	0	0	0	0
0	0	0	0	0	0	0	186	252	252	252	245	108	53	0	0	0	150	252	252	220	20	0	0	0	0	0	0
0	0	0	0	0	0	70	242	252	252	222	59	0	0	0	0	0	178	252	252	141	0	0	0	0	0	0	0
0	0	0	0	0	0	185	252	252	194	67	0	0	0	0	17	90	240	252	194	67	0	0	0	0	0	0	0
0	0	0	0	0	0	83	205	190	24	0	0	0	0	0	121	252	252	209	24	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	77	247	252	248	106	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	253	252	252	102	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	134	255	253	253	39	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	183	253	252	107	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	10	102	252	253	163	16	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	13	168	252	252	110	2	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	41	252	252	217	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	40	155	252	214	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	165	252	252	106	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	43	179	252	150	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	137	252	221	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	67	252	79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[Download MNIST](http://yann.lecun.com/exdb/mnist/)
http://yann.lecun.com/exdb/mnist/

Learning from examples – Classifiers

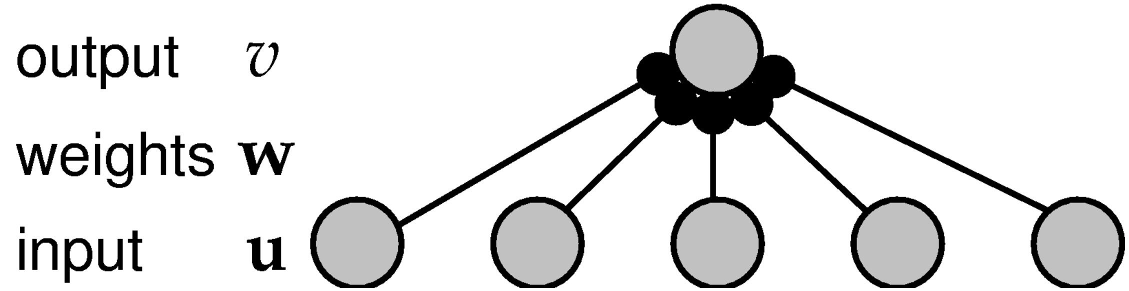


Learning from examples – Classifiers and cross-validation



Learning from examples – The perceptron

Imagine that we want to classify the inputs ***u*** into two groups "+1" (=3) and "-1" (=7)



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

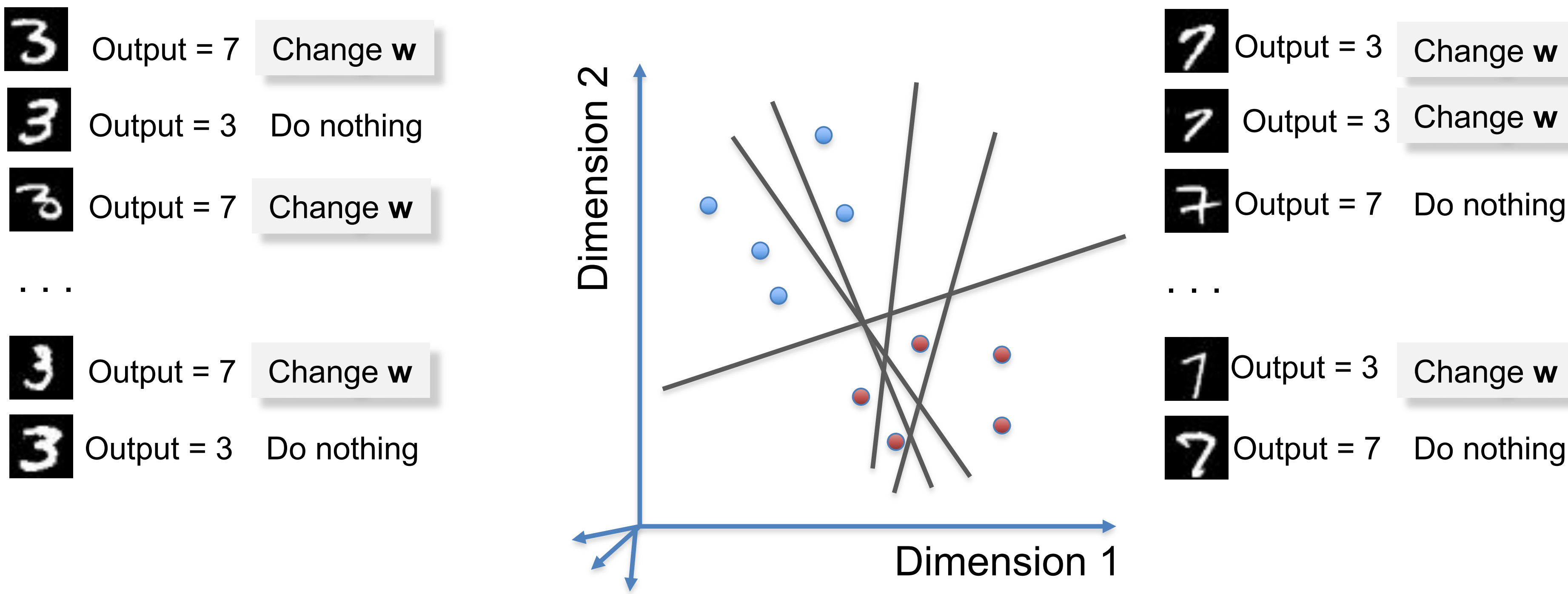
Training examples: $\{\mathbf{u}_{\mathrm{m}}, v_m\}$

$$\mathbf{w} \rightarrow \mathbf{w} + \frac{\epsilon}{2}\left(v_m - v(\mathbf{u}_m)\right)\mathbf{u}_m$$

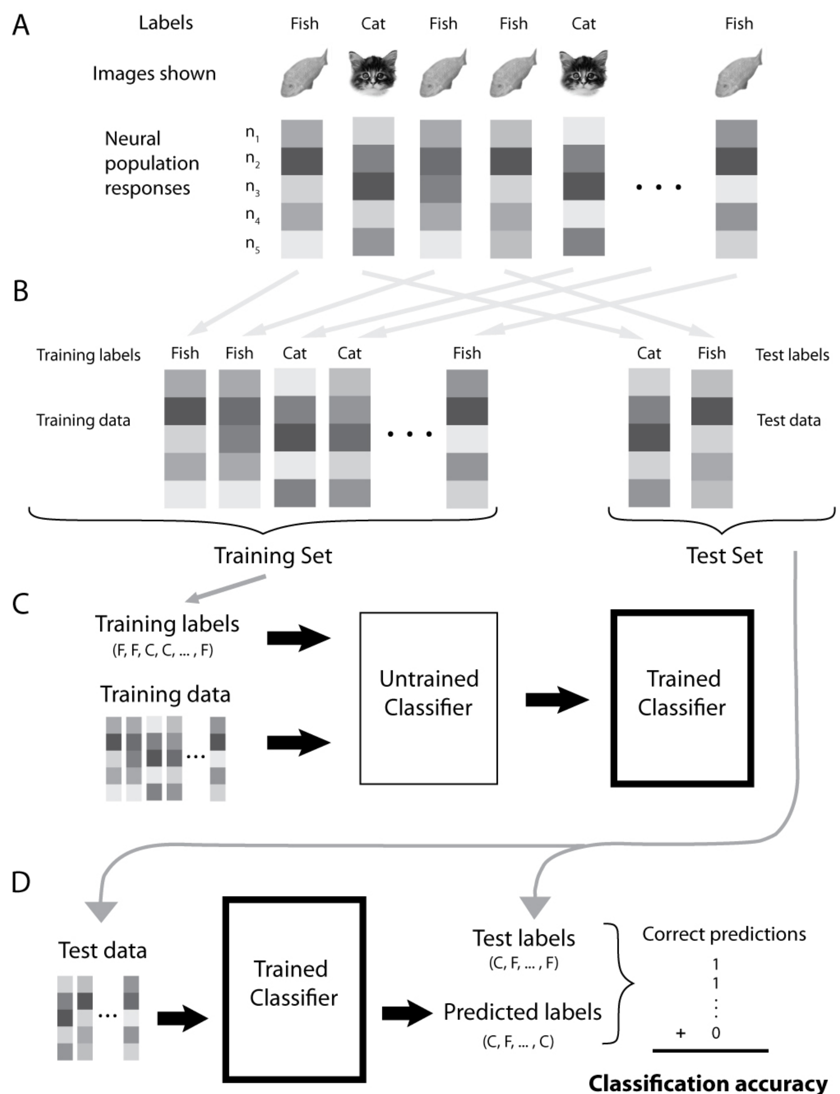
Perceptron learning rule

Linear separability: can attain zero error
Cross-validation: use separate training and test data
There are several more sophisticated learning algorithms

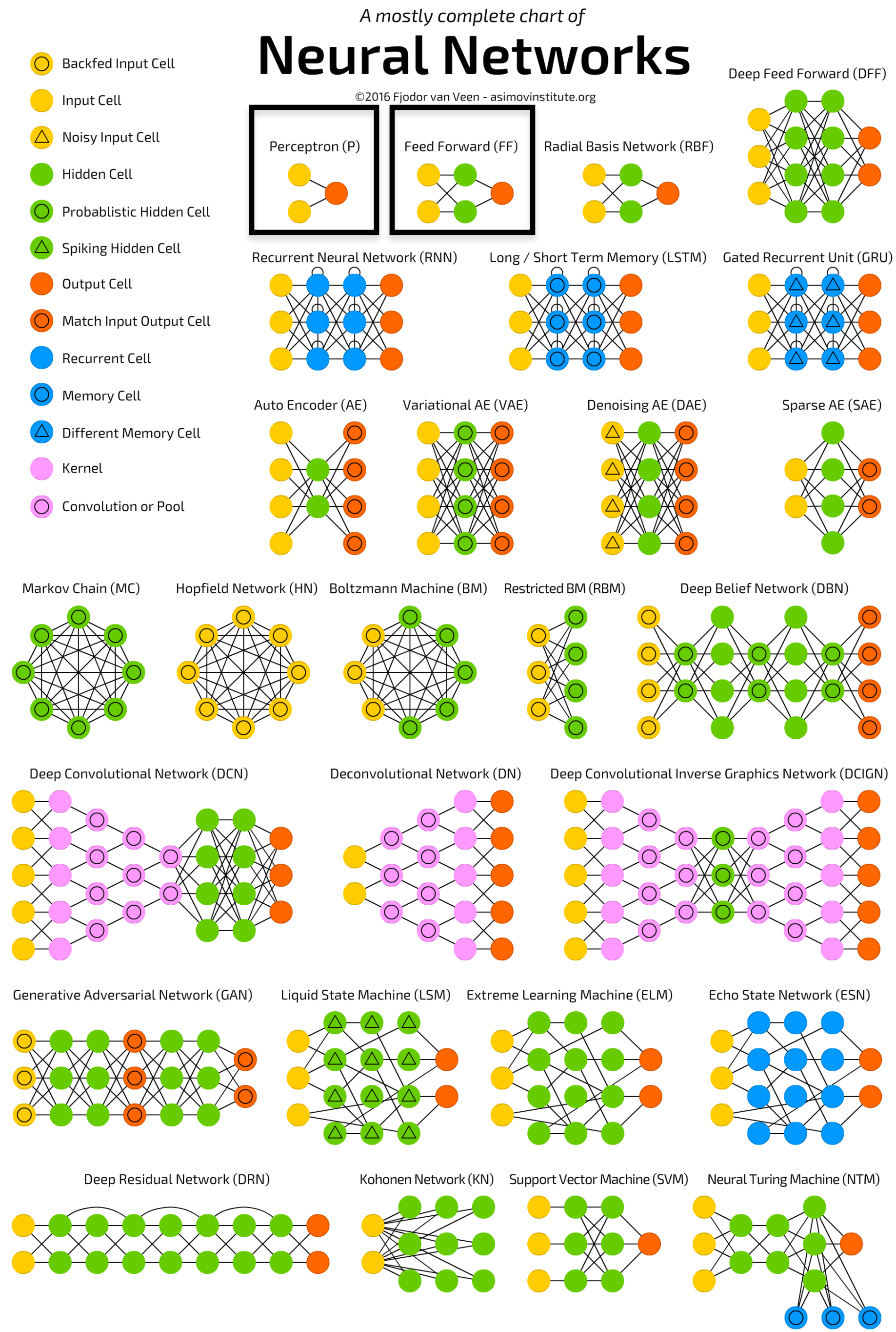
Learning from examples – Training



Learning from examples – Classifiers and cross-validation



A big happy family of neural networks



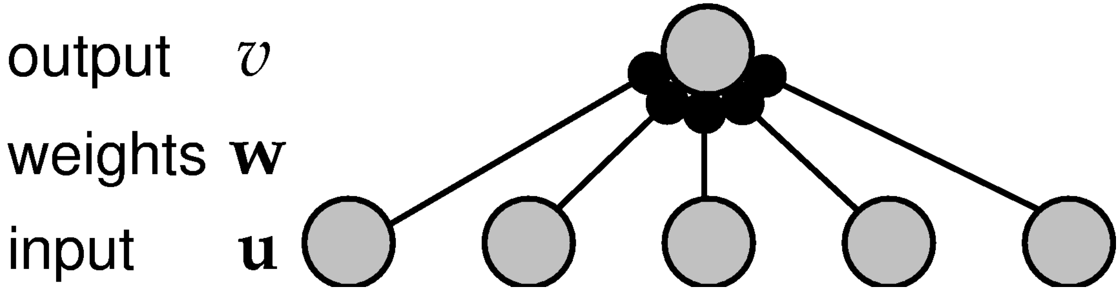
https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Learning from examples – Gradient descent

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

$$\mathbf{u} = \mathbf{f}(s)$$

$$v(s) = \mathbf{w}.\mathbf{u}$$



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

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

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \nabla_w E \qquad \nabla_w E = \left[\frac{\partial E}{\partial w_b}\right]$$

Example: digit recognition in a feed-forward network trained by gradient descent

Example of hand-written digits (MINT database)

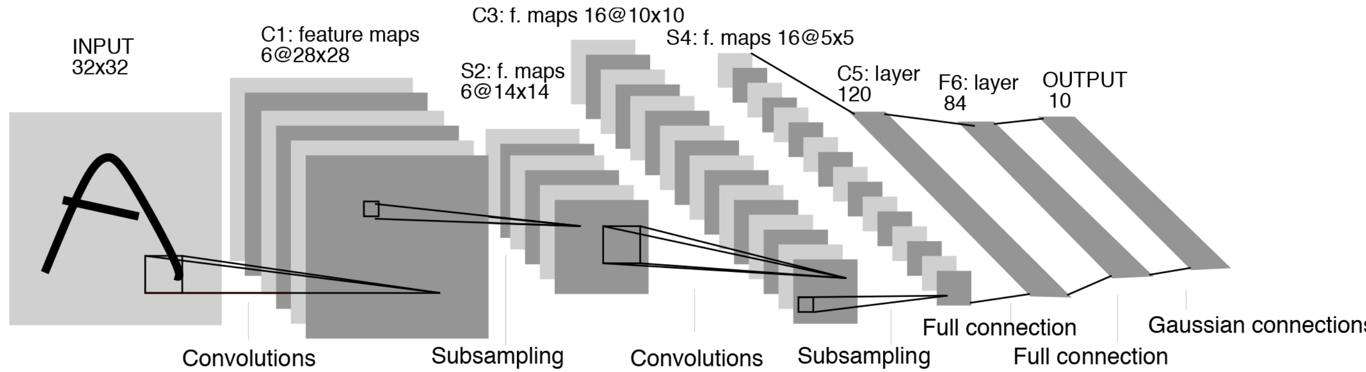
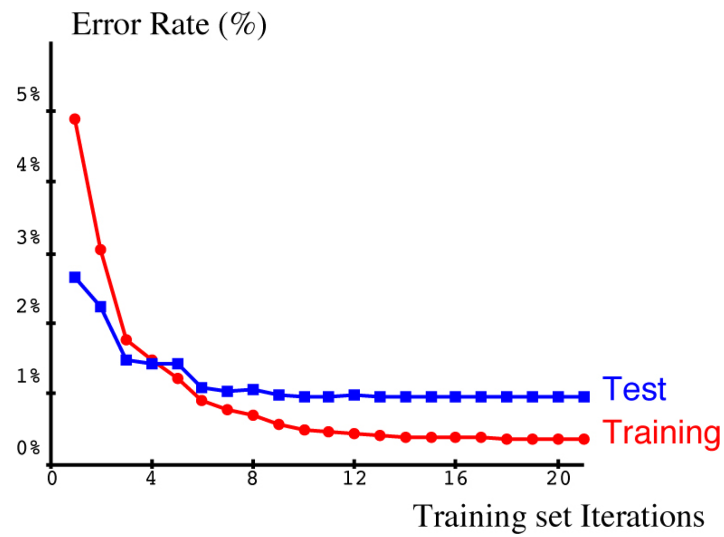


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of unit whose weights are constrained to be identical.

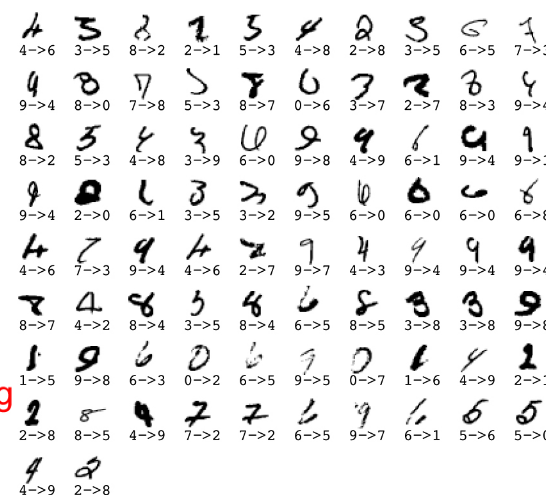
LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. Proc of the IEEE 86:2278-2324.

Example: digit recognition in a feed-forward network trained by gradient descent

Example of hand-written digits (MINT database)



Classification error rates



Misclassified examples

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. Proc of the IEEE 86:2278-2324.

The “blue brain” modeling project

-http://bluebrain.epfl.ch

- IBM’s Blue gene supercomputer

- “Reverse engineer” the brain in a “biologically accurate” way

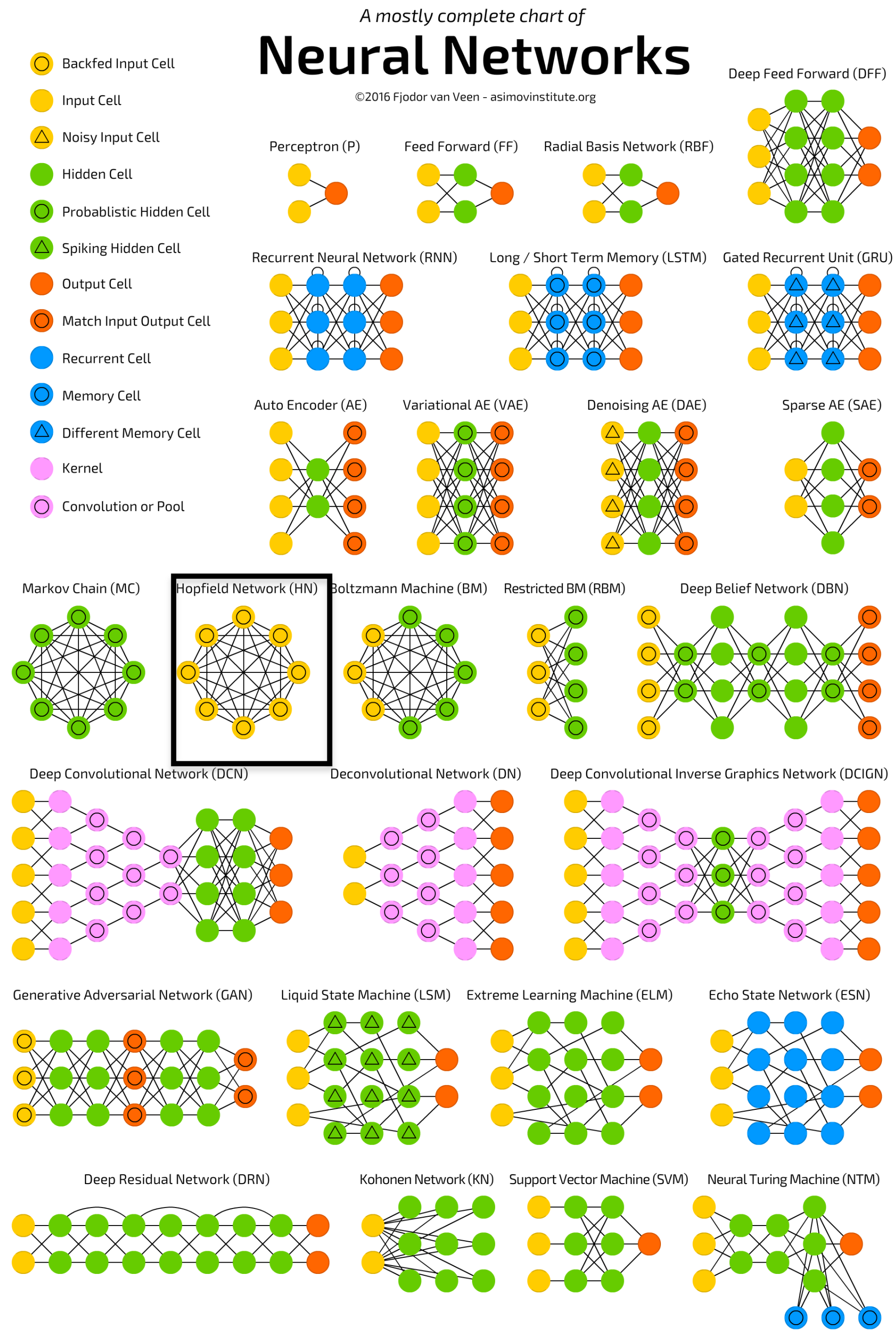
- November 2007 milestone: 30 million synapses in “precise” locations to model a neocortical column

- 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 big happy family of neural networks

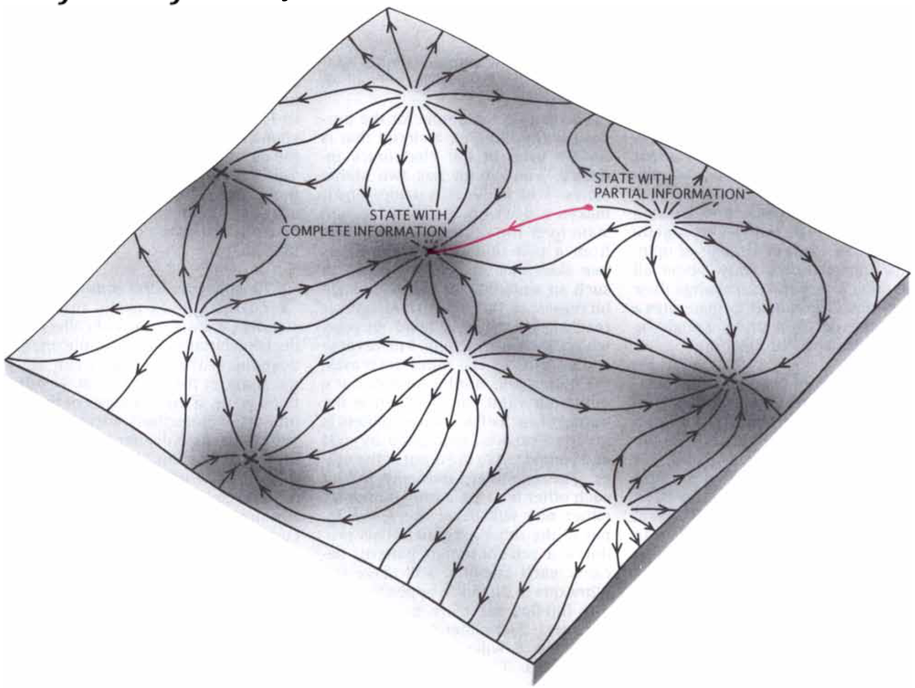


https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Hopfield networks:
A case study in collective computation

$w_{ii} = 0$ No self connections

$w_{ij} = w_{ji}$ Symmetric connections



$\boldsymbol{s} = [s_1, \dots, s_N]$ State vector

$$s_i[t+1] = sign(\sum_{j \neq i}^{N} w_{ij} s_j[t] - \theta)$$

State update

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i s_i \theta_i$$

Energy function

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{n} \epsilon_i^{\mu} \epsilon_j^{\mu}$$

Hebbian learning

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. **PNAS** 79:2554-2558.

Tank, D., and J. Hopfield. 1987. Collective computation in neuron-like circuits. **Scientific American** 257:104-114

Summary

- To understand vision, it is essential to build computational models
- We use abstract models where biological properties are simplified
- The integrate-and-fire neuron captures essential input-output properties
- The convolution operation allows extracting the same visual features throughout the entire visual field
- Basic elementary computations: filtering, normalization, pooling, thresholding
- Neural networks show emergent computational properties
- Neural networks include feedforward, horizontal and top-down connections
- Attractor-based recurrent neural networks show dynamic properties that save energy, provide flexible computations, and robustness to perturbations

Further reading

•Abbott and Dayan. Theoretical Neuroscience - Computational and Mathematical Modeling of Neural Systems [2001] (ISBN 0-262-04199-5). MIT Press.

•Koch. Biophysics of computation [1999] (ISBN 0-19-510491-9). Oxford University Press.

•Hertz, Krogh, and Palmer, *Introduction to the theory of neural computation*. [1991] (ISBN 0-20151560-1). Santa Fe Institute Studies in the Sciences of Complexity.

•Gabbiani and Cox. [2010]. Mathematics for Neuroscientists (London: Academic Press).