Neural networks and visual processing

Mark van Rossum

School of Informatics, University of Edinburgh

January 15, 2018

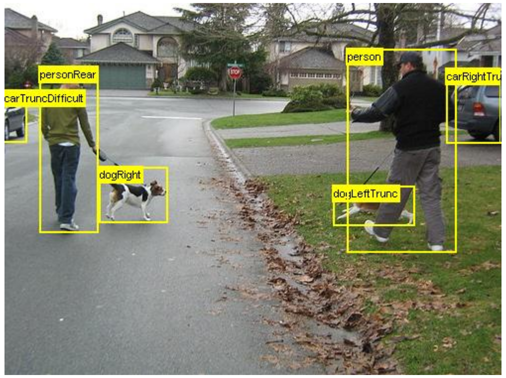
[0] Version: January 15, 2018.

Neurobiology of Vision

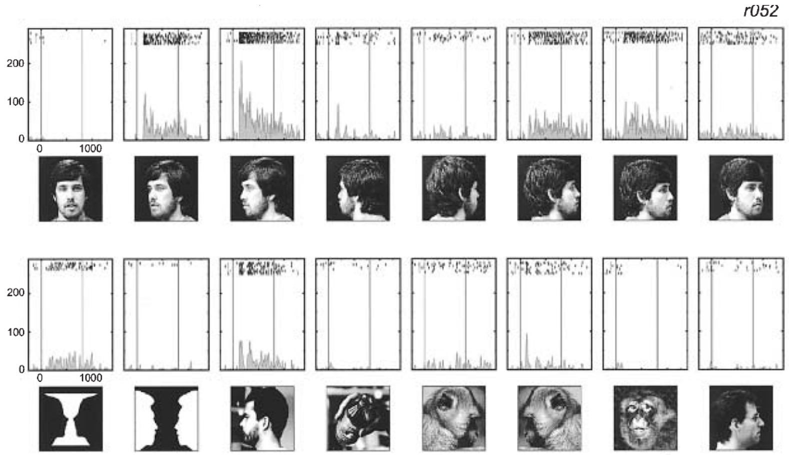
- WHAT pathway: V1 → V2 → V4 → IT (focus of our treatment)
- WHERE pathway: V1 → V2 → V3 → MT/V5 → parietal lobe
- IT (Inferotemporal cortex) has cells that are
  - Highly selective to particular objects (e.g. face cells)
  - Relatively invariant to size and position of objects, but typically variable wrt 3D view
- What and where information must be combined somewhere ('throw the ball at the dog')

Example tasks

- Classification
  - Is there a dog in this image?
- Detection
  - Localize all the people (if any) in this image
- etc..

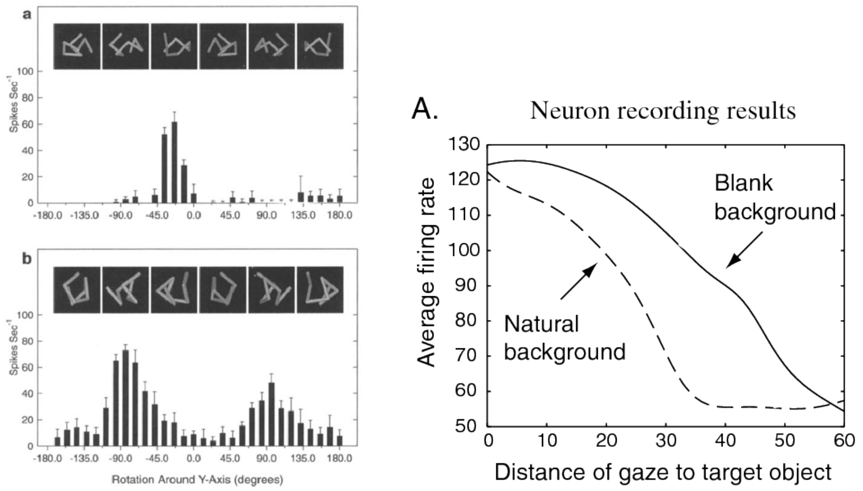


Invariances in higher visual cortex



[?]

Invariance is however limited

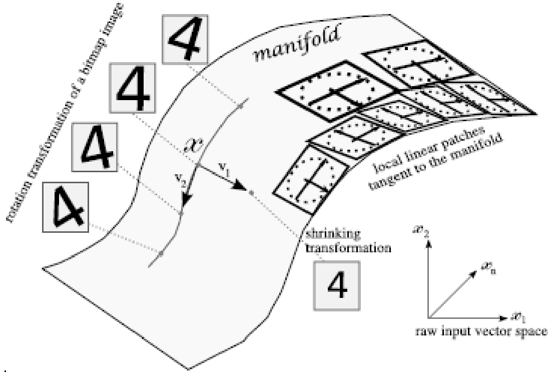


Left: partial rotation invariance [**?**].
Right: clutter reduces translation invariance [**?**].

Computational Object Recognition

- The big problem is creating *invariance* to scaling, translation, rotation (both in-plane and out-of-plane), and partial occlusion, yet at the same time being selective.
- Large input dimension, need enormous (labelled) training set + tricks
- Objects are not generally presented against a neutral background, but are embedded in *clutter*
- Within class variation of objects (e.g. cars, handwritten letters, ..)

Geometrical picture


rotation transformation of a bitmap image
manifold
x
v_1
v_2
local linear patches tangent to the manifold
shrinking transformation
x_2
x_n
x_1
raw input vector space

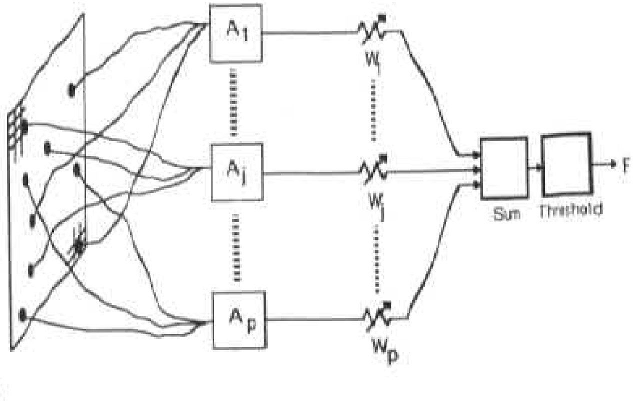
[From Bengio 2009 review]
Pixel space. Same objects form manifold (potentially discontinuous, and disconnected).

History of artificial neural networks

- McCullough & Pitts (1943): Binary neurons can implement any finite state machine. Von Neumann used this for his architecture.
- Rosenblatt (1962): Perceptron learning rule: Learning of (some) binary classification problems.
- Backprop (1980s): Universal function approximator. Generalizes, but has local maxima.
- Boltzmann machines (1980s): Probabilistic models. Long ignored for being exceedingly slow.
- 2005- : Backprop and variants popular again.

Perceptrons

- Supervised binary classification of K N-dimensional $\mathbf{x}^\mu$ pattern vectors.
- $y = H(h) = H(\mathbf{w}.\mathbf{x} + b)$, H is step function, $h = \mathbf{w}.\mathbf{x} + b$ is net input ('field')



[ignore A_i in figure for now, and assume x_i is pixel intensity]

Perceptron learning rule

- Denote desired binary output for pattern μ as d^μ. Rule:

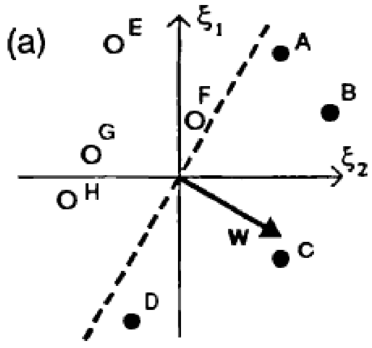
$$\Delta w_i^\mu = \eta x_i^\mu (d^\mu - y^\mu)$$

or, to be more robust, with margin κ

$$\Delta w_i^\mu = \eta H(N\kappa - h^\mu d^\mu) d^\mu x_i^\mu$$

- note, if patterns correct then $\Delta w_i^\mu = 0$ (stop-learning).
- *If* learnable, rule converges in polynomial time.

Perceptron learning rule



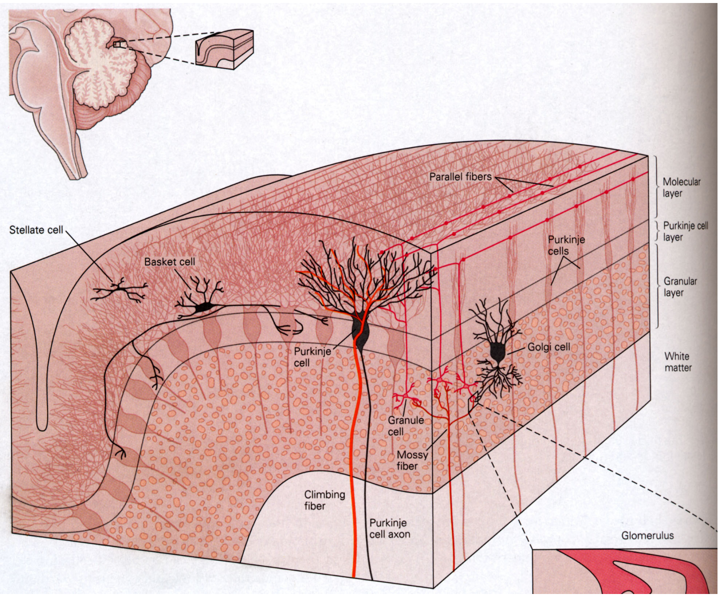
- Learnable if patterns are linearly separable.
- Random patterns are typically learnable if $\#patterns < 2.\#inputs$, $K < 2N$.
- Mathematically solves set of inequalities.
- General trick: replace bias $b = w_b.1$ with 'always on' input.

Perceptron biology

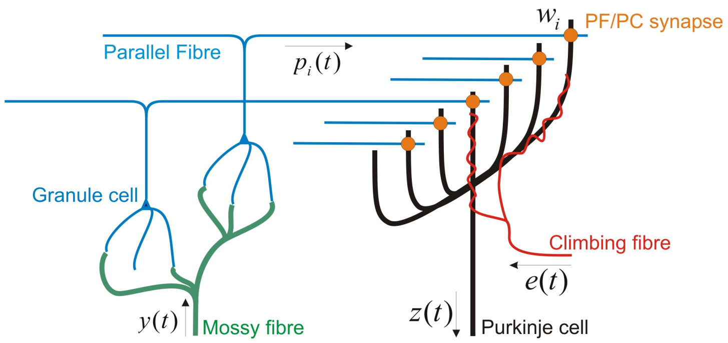
Tricky questions

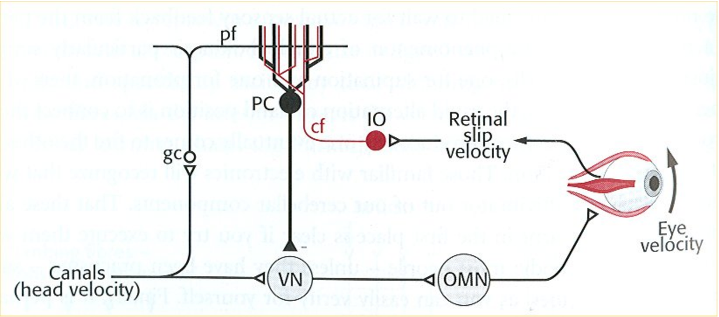
- How is the supervisory signal coming into the neuron?
- How is the stop-learning implemented in Hebbian model where $\Delta w_i \propto x_i y$?
- Related to cerebellar learning (Marr-Albus theory), to learn reduce motor errors.

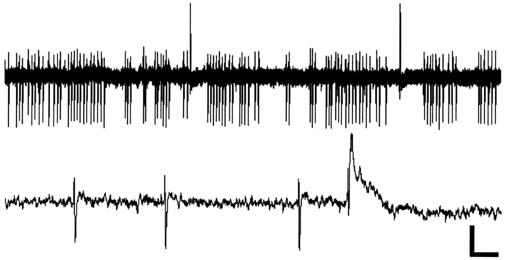
Perceptron and cerebellum



Perceptron and cerebellum

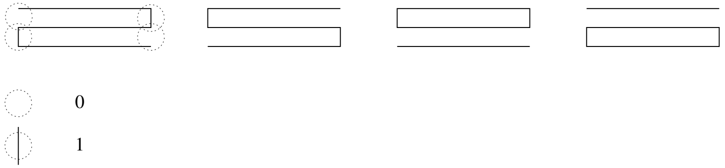






[Purkinje cell spikes recorded extra-cellularly + zoom]
Simple spikes: standard output. Complex spikes: IO feedback, trigger plasticity.

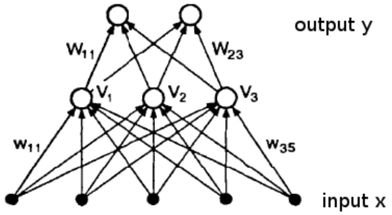
Perceptron limitation



- Perceptron with limited receptive field cannot determine connectedness (give output 1 for connected patterns and 0 for dis-connected).
- This is the XOR problem, $d = 1$ if $x_1 \neq x_2$. This is the simplest parity problem, $d = (\sum_i x_i) mod 2$.
- Equivalently, identity function problem, $d = 1$ if $x_1 = x_2$.
- In general: categorizations that are not linearly separable cannot be learned (weight vector keeps wandering).

Multi-layer perceptron (MLP)

- Supervised algorithm that overcomes limited functions of the single perceptron.
- With continuous units and large enough single hidden layer, MLP can approximate any continuous function! (and two hidden layers approximate any function). Argument: write function as sum of localized bumps, implement bumps in hidden layer.
- Ultimate goal is not the learning of the patterns (after all we could just make a database), but a sensible generalization. The performance on test-set, not training set, matters.



- $y_i^\mu(\mathbf{x}^\mu; w, W) = g(\sum_j W_{ij} v_j) = g\left(\sum_j W_{ij} g(\sum_k w_{jk} x_k)\right)$
- Learning: back-propagation of errors. Mean squared error of P training patterns:

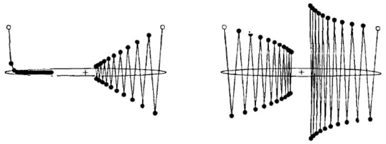
$$E = \sum_{\mu=1}^{P} E_\mu = \frac{1}{2}\sum_{\mu=1}^{P} [d_i^\mu - y_i^\mu(\mathbf{x}^\mu; w, W)]^2$$

Gradient descent (batch) "$\Delta w \propto -\eta \frac{\partial E}{\partial w}$" where w are all the weights (input → hidden, hidden → output, biases).

- Stochastic descent: Pick arbitrary pattern, use $\Delta w = -\eta \frac{\partial E_\mu}{\partial w}$ instead of $\Delta w = -\eta \frac{\partial E}{\partial w}$. Quicker to calculate, and randomness helps learning.
- $\frac{\partial E^\mu}{\partial W_{ij}} = (y_i - d_i) g'(\sum_k W_{ik} v_k) v_j \equiv \delta_i v_j$
- $\frac{\partial E^\mu}{\partial w_{jk}} = \sum_i \delta_i W_{ij} g'(\sum_l w_{jl} x_l) x_k$
- Start from random, smallish weights. Convergence time depends strongly on lucky choice.
- If $g(x) = [1 + exp(-x)]^{-1}$, one can use $g'(x) = g(x)(1 - g(x))$.
- Normalize input (e.g. z-score)

MLP tricks

Learning MLPs is slow and local maxima are present.

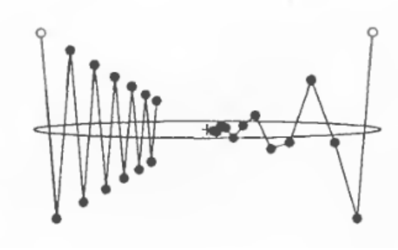


[from HKP, increasing learning rate. 2nd: fastest, 4th: too big]

- Learning rate often made adaptive (first large, later small).
- Sparseness priors are often added to prevent large negative weights cancelling large positive weights.
  e.g $E = \frac{1}{2}\sum_\mu (d^\mu - y^\mu(\mathbf{x}^\mu; w))^2 + \lambda \sum_{i,j} w_{ij}^2$
- Other cost functions are possible.
- Traditionally one hidden layer. More layers do not enhance repertoire and slow down learning (but see below).

MLP tricks

Momentum: previous update is added, hence wild direction fluctuations in updates are smoothed.



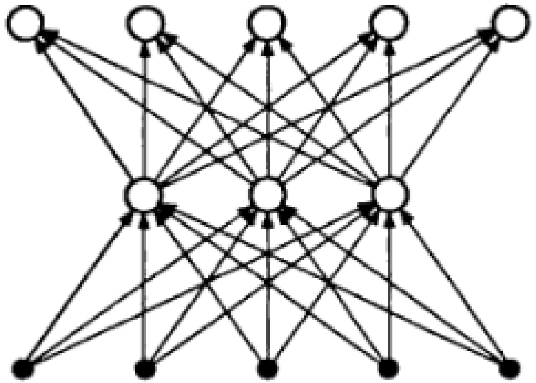
[from HKP. Same learning rate but with (right) and without momentum (left)].

MLP examples

Essentially curve fitting. Best on problems that are not fully understood / hard to formulate.

- Hand-written postcodes.
- Self-driving car at 5km/h ($\sim$ 1990)
- Backgammon game

Auto-encoders



Autoencoders: Minimize $E(input, output)$
Fewer hidden units than input units: find optimal compression (PCA when using linear units).

Biology of back-propagation?

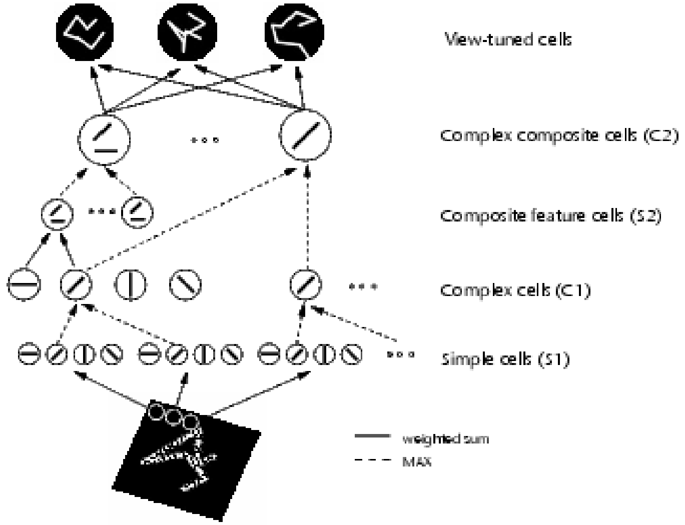
- How to back-propagate in biology?
- O'Reilly (1996) Adds feedback weights (do not have to be exactly symmetric).
- Uses 2-phases. -phase: input clamped; +phase: input and output clamped.
- Approximate $\Delta w_{ij} = \eta(post_i^+ - post_i^-)pre_j^-$
- more when doing Boltzmann machines...
- More recent work (Bengio, Lillicrap)

Convolutional networks

Neocognitron [**?**, **?**, **?**]

- To implement location invariance, “clone” (or replicate) a detector over a region of space (weight-sharing), and then pool the responses of the cloned units
- This strategy can then be repeated at higher levels, giving rise to greater invariance and faster training

HMAX model

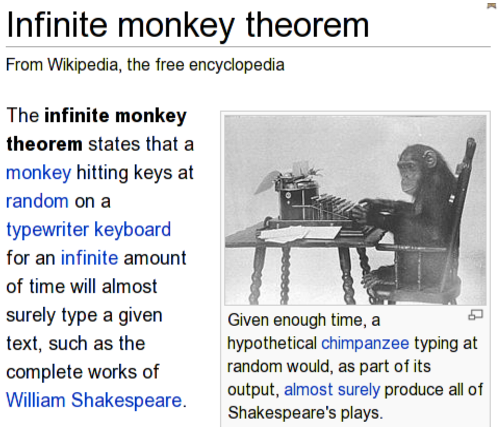


[**?**]

HMAX model

- Deep, *hard-wired* network
- S1 detectors based on Gabor filters at various scales, rotations and positions
- S-cells (simple cells) convolve with local filters
- C-cells (complex cells) pool S-responses with maximum
- No learning between layers !
- Object recognition: Supervised learning on the output of C2 cells.

Rather than learning, take refuge in having many, many cells. (Cover, 1965)*A complex pattern-classification problem, cast in a high-dimensional space non-linearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.*

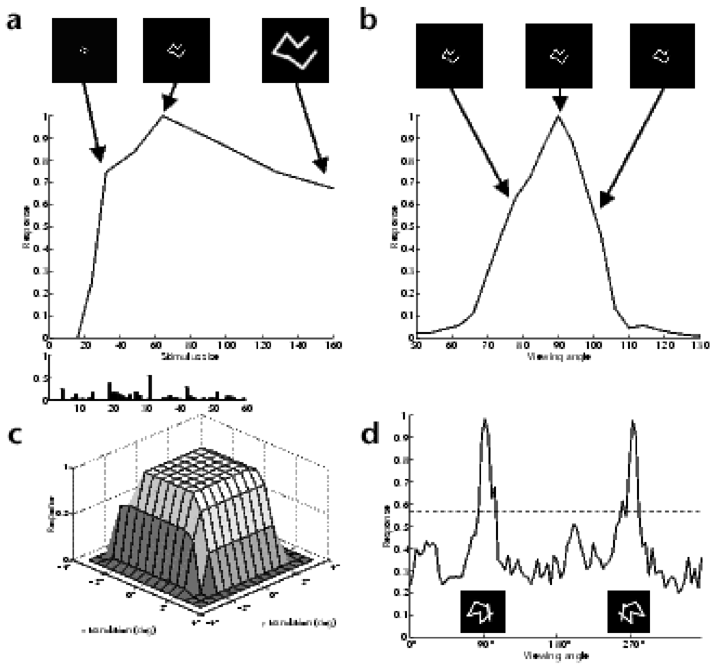


Infinite monkey theorem

From Wikipedia, the free encyclopedia

The **infinite monkey theorem** states that a monkey hitting keys at random on a typewriter keyboard for an infinite amount of time will almost surely type a given text, such as the complete works of William Shakespeare.

Given enough time, a hypothetical chimpanzee typing at random would, as part of its output, almost surely produce all of Shakespeare's plays.



[?]

HMAX model: Results

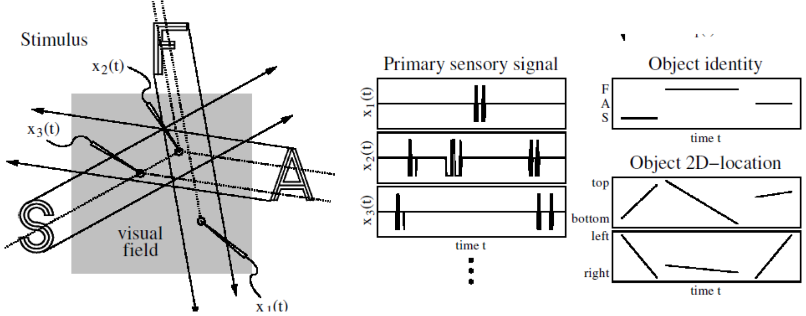
- “paper clip” stimuli
- Broad tuning curves wrt size, translation
- Scrambled input image does not give rise to object detections: not all conjunctions are preserved

Learning invariances

- Hard-code (convolutional network) http://yann.lecun.com/exdb/lenet/
- Supervised learning: show samples and require same output. Augmentation with mirror, partial and scaled images.
- Use temporal continuity of the world. Learn invariance by seeing object change, e.g. it rotates, it changes colour, it changes shape.
  Algorithms: trace rule[**?**]
  E.g. replace
  $\Delta w = x(t).y(t)$ with $\Delta w = x(t).\tilde{y}(t)$
  where $\tilde{y}(t)$ is temporally filtered $y(t)$.
- Similar principles: VisNet [**?**], Slow feature analysis.

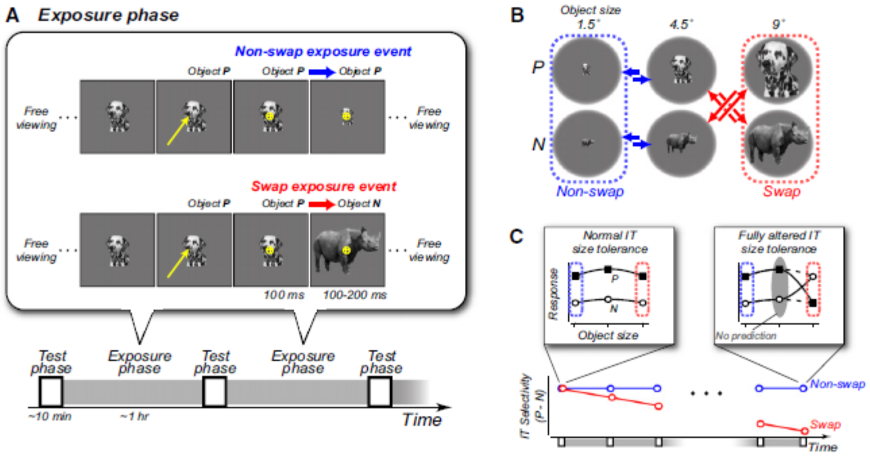
Slow feature analysis

Find slow varying features, these are likely relevant [**?**]



Find output y for which: $\langle(\frac{dy(t)}{dt})^2\rangle$ minimal,
while $\langle y\rangle = 0, \langle y^2\rangle = 1$

Experiments: Altered visual world [**?**]



Including top-down interaction

- Extensive top-down connections everywhere in the brain
- One known role: attention. For the rest: many theories

[**?**]

Local parts can be ambiguous, but knowing global object at helps.
Top-down to set priors.
Improvement in object recognition is actually small,
but recognition and localization of parts is much better.

Deep MLPs

- Traditional MLPs are also called shallow (1 or 2 hidden layers).
- While deeper nets do not have more computational power. 1) Some tasks require less nodes (e.g. 1 hidden layer: parity requires exp. many hidden layer units) 2) they can lead to better representations. Better representations lead to better generalization and better learning.
- Learning slows down in deep networks, as transfer functions $g()$ saturate at 0 or 1. ($\Delta w \propto g'() \to 0$) So:
  - Pre-training, e.g. with Boltzmann machines (see below)
  - Convolutional networks
  - Use non-saturating activation function.
- Better representation by adding noisy/partial stimuli. This artificially increases the training set and forces invariances.

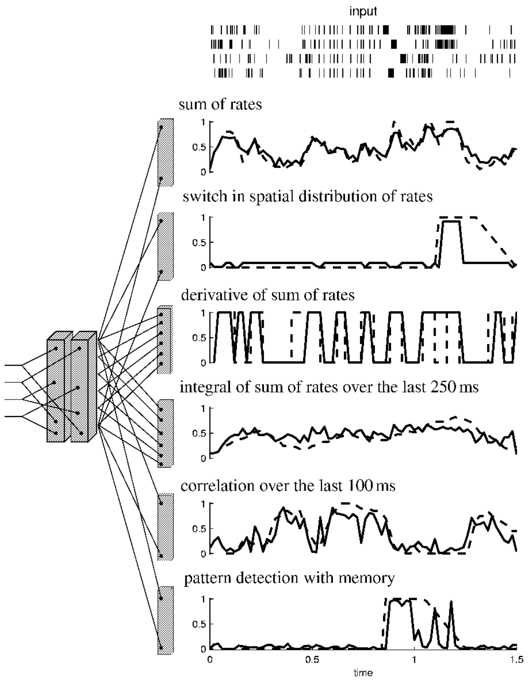
Recurrent networks

- MLPs have no dynamics
- Recurrent networks are dynamic. Could be steady state(s), periodic, or chaotic. With symmetric weights there can only be fixed points (point or line attractors).
- In recurrent networks it is much harder to find weights to be altered (credit assignment). Often restrict to cases where dynamics has fixed points.
- Hopfield net; Boltzman machine; Liquid state machine

Liquid state machines

[**?**]

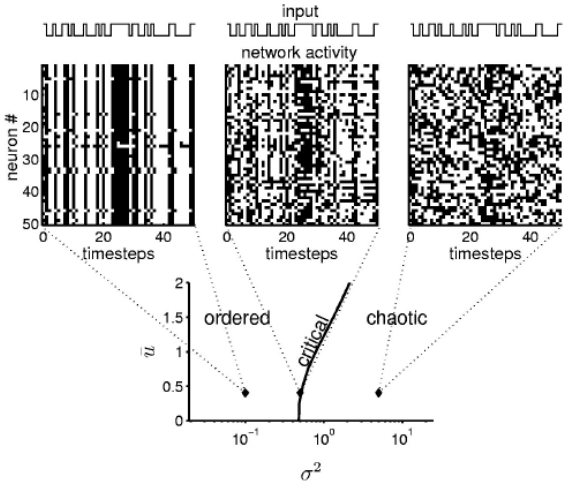
- Motivation: arbitrary spatio-temporal computation without precise design.
- Create pool of spiking neurons with random connections.
- Results in very complex dynamics if weights are strong enough
- Similar to echo state networks (but those are rate based).
- Both are known as reservoir computing
- Similar theme as HMAX model: create rich repetoire and only learn at the output layer.



Various functions can be implemented by varying readout.

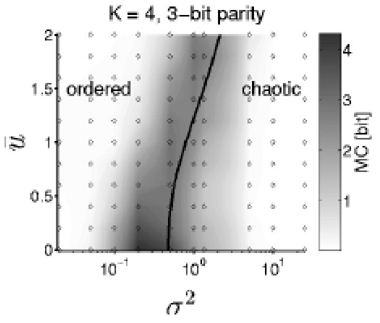
Optimal reservoir?

Best reservoir has rich yet predictable dynamics.
Edge of Chaos [**?**]



Network 250 binary nodes, $w_{ij} = \mathcal{N}(0, \sigma^2)$
(x-axis is recurrent strength)

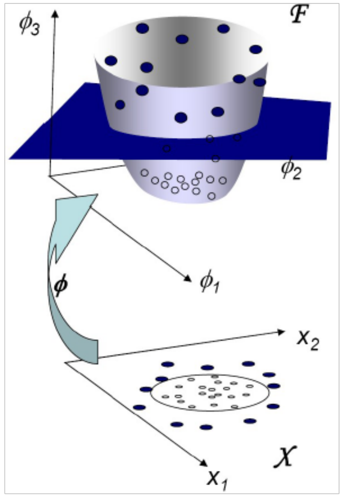
Optimal reservoir?



Task: $Parity(in(t), in(t-1), in(t-2))$
Best (darkest in plot) at edge of chaos.
Does chaos exist in the brain?

- In spiking network models: yes [**?**]
- In real brains: ?

Relation to Support Vector Machines



Map problem in to high dimensional space $\mathcal{F}$; there it often becomes linearly separable.
This can be done without much computational overhead (kernel trick).

References I