Neural networks and visual processing

Mark van Rossum

School of Informatics, University of Edinburgh

January 24, 2018

[0] Version: January 24, 2018.

Overview

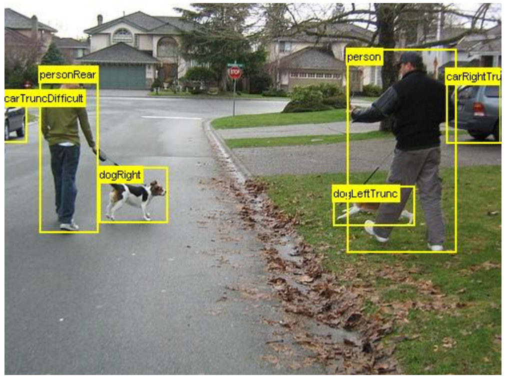
- So far we have discussed unsupervised learning up to V1
- For most technology applications (except perhaps compression), V1 description is not enough. Yet it is not clear how to proceed to higher areas.
- At some point supervised learning will be necessary to attach labels. Hopefully this can be postponed to very high levels.

Neurobiology of Vision

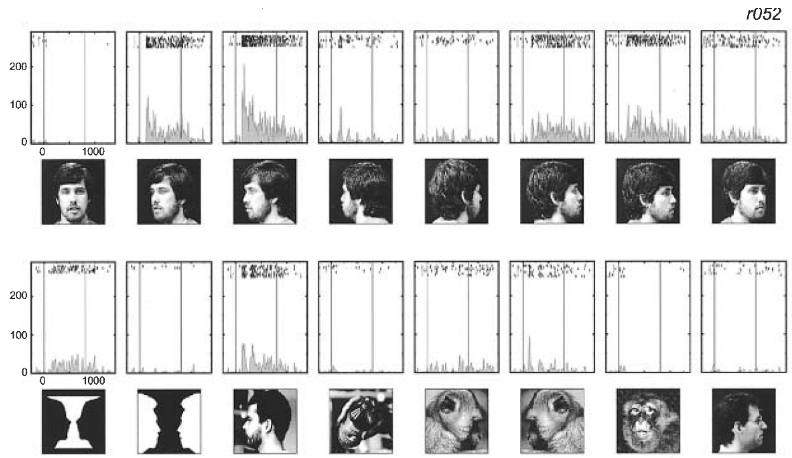
- WHAT pathway: V1 $\to$ V2 $\to$ V4 $\to$ IT (focus of our treatment)
- WHERE pathway: V1 $\to$ V2 $\to$ V3 $\to$ MT/V5 $\to$ parietal lobe
- IT (Inferotemporal cortex) has cells that are
  - Highly selective to particular objects (e.g. face cells)
  - Relatively invariant to size and position of objects, 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..

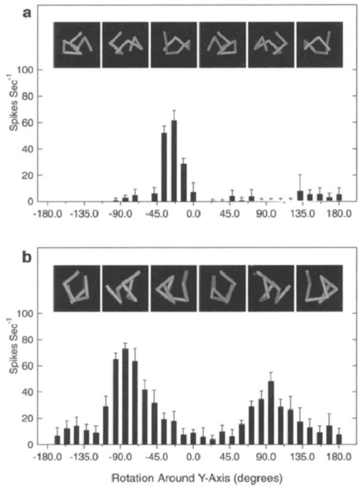


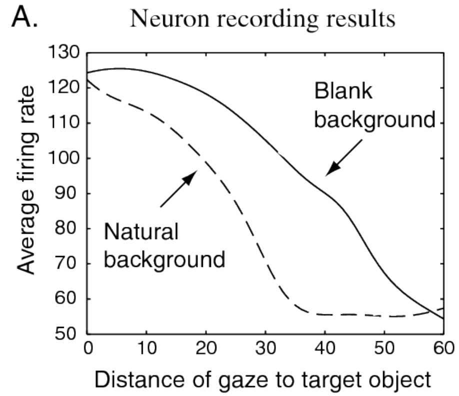
Invariances in higher visual cortex



[Logothetis and Sheinberg, 1996]

Invariance is however limited



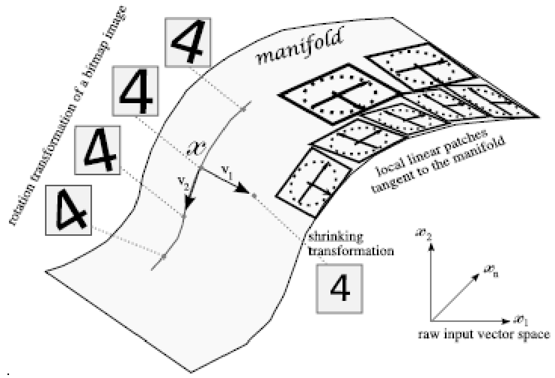


Left: partial rotation invariance [Logothetis and Sheinberg, 1996].
Right: clutter reduces translation invariance [Rolls and Deco, 2002].

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



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

Some Computational Models

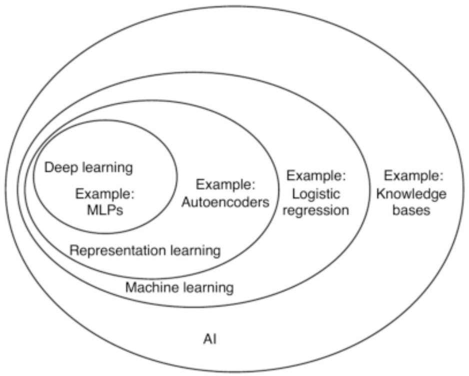
Two extremes:

- Extract 3D description of the world, and match it to stored 3D structural models (e.g. human as generalized cylinders)
- Large collection of 2D views (templates)

Some other methods

- 2D structural description (parts and spatial relationships)
- Match image features to model features, or do pose-space clustering (Hough transforms))
  - What are good types of features?
- Feedforward neural network
- Bag-of-features (no spatial structure; but what about the "binding problem"?)
- Scanning window methods to deal with translation/scale

AI



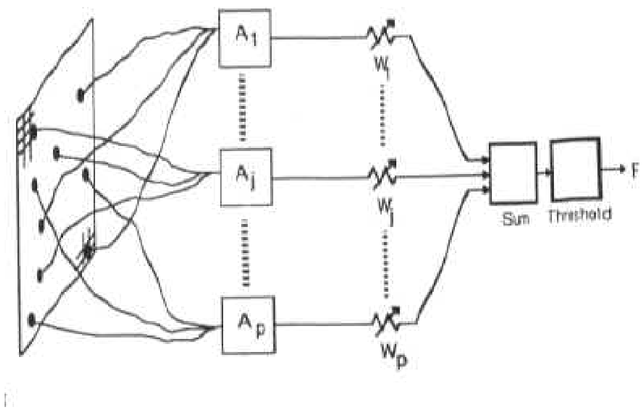
[Bengio et al., 2014]

History

- 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.

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:

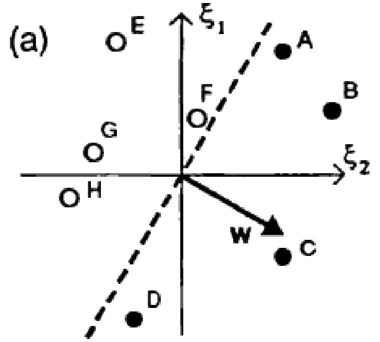
$$\Delta w_i^\mu = \eta x_i^\mu (d^\mu - y^\mu)$$

  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



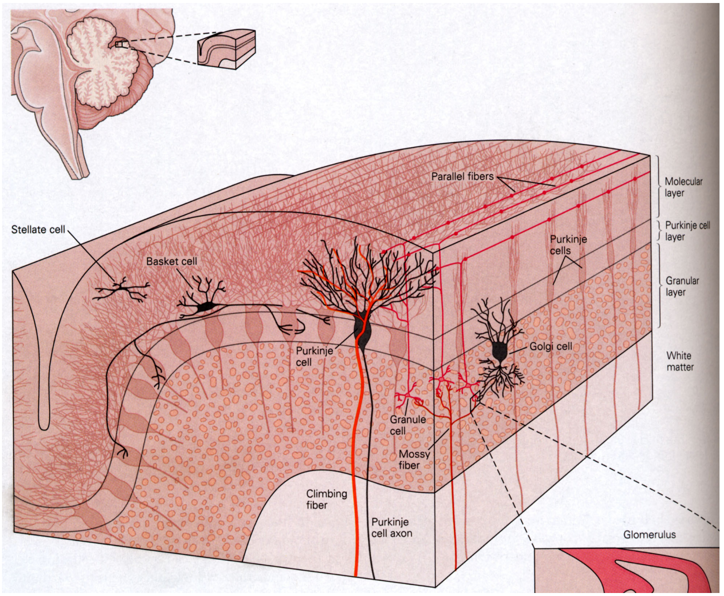
- Learnable if patterns are linearly separable.
- Random patterns are typically learnable if $\#patterns < 2.\#inputs$, $K < 2N$.
- Mathematically solves set of inequalities.
- General trick: replace bias $b = w_b.1$ with 'always on' input.

Perceptron biology

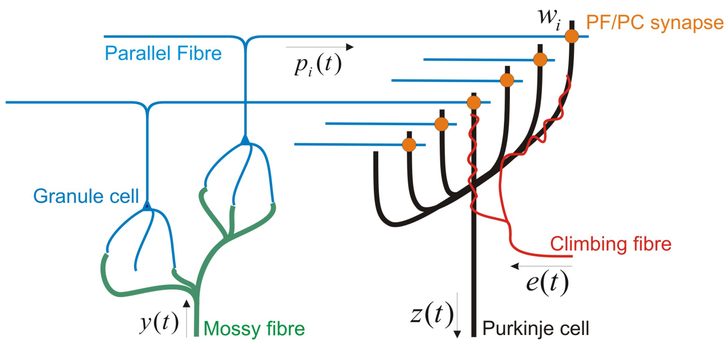
Tricky questions

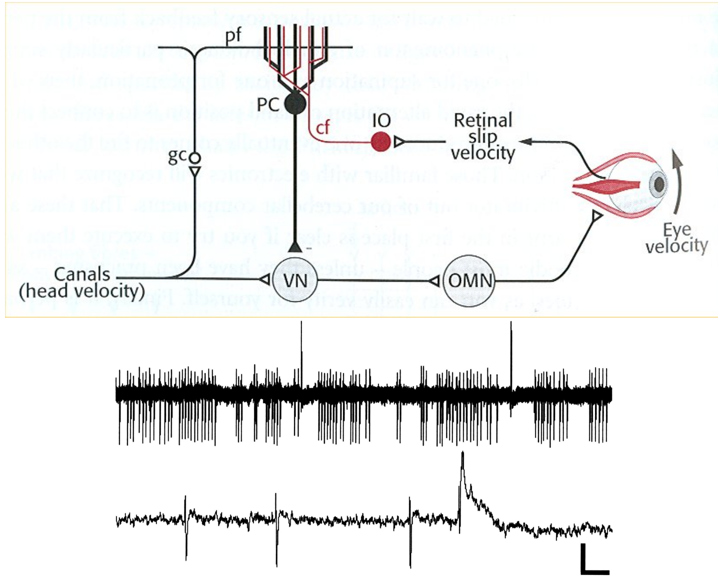
- How is the supervisory signal coming into the neuron?
- How is the stop-learning implemented in Hebbian model where $\Delta w_i \propto x_i y$?
- Perhaps related to cerebellar learning (Marr-Albus theory)

Perceptron and cerebellum



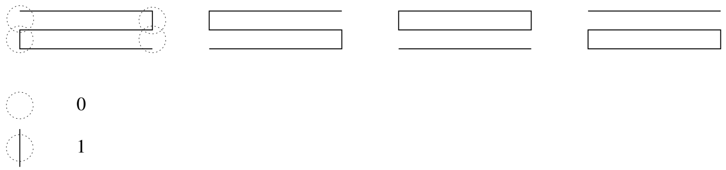
Perceptron and cerebellum





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

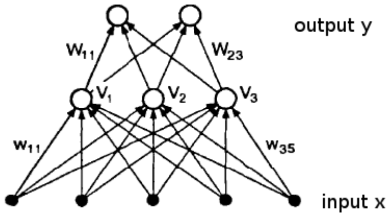
Perceptron limitation



- 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:

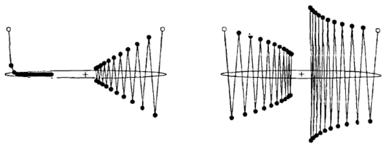
$$E = \sum_{\mu=1}^{P} E_\mu = \frac{1}{2}\sum_{\mu=1}^{P}[d_i^\mu - y_i^\mu(\mathbf{x}^\mu; w, W)]^2$$

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

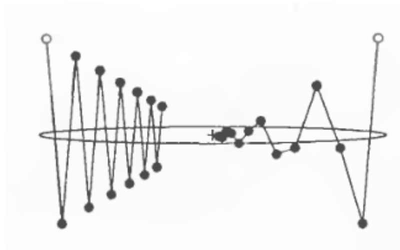


[from HKP, increasing learning rate. 2nd: fastest, 4th: too big]

- 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.



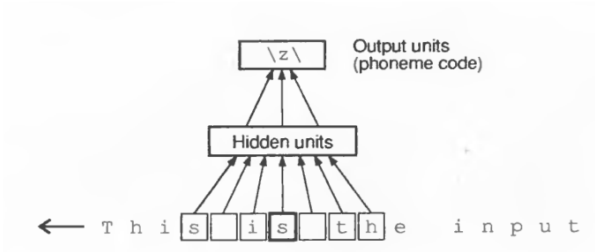
[from HKP. Same learning rate but with (right) and without momentum (left)].

MLP examples

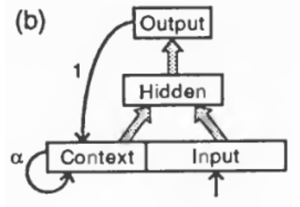
Essentially curve fitting. Best on problems that are not fully understood / hard to formulate.

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

MLP sequence data

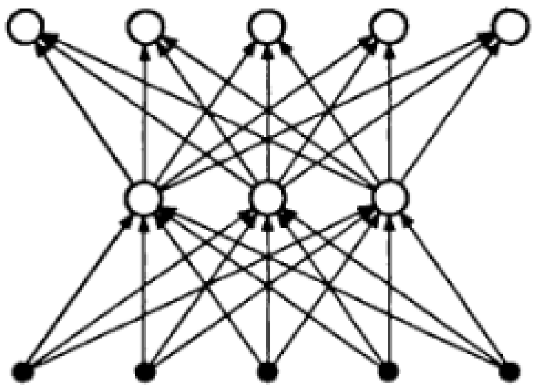


- Temporal patterns by for instance setting input vector as $\{s_1(t), s_2(t), \ldots s_n(t), s_1(t-1), \ldots s_n(t-1)\}$.



- Context units that decay over time (Ellman net)

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?

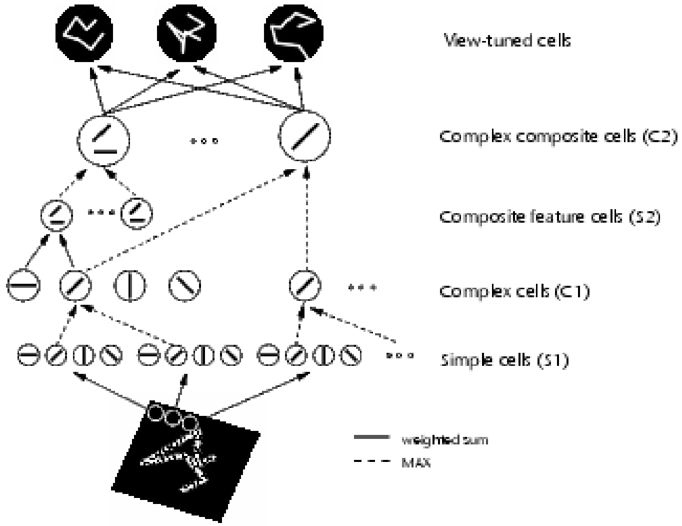
- How to back-propagate in biology?
- O'Reilly (1996) Adds feedback weights (do not have to be exactly symmetric).
- Uses 2-phases. -phase: input clamped; +phase: input and output clamped.
- Approximate $\Delta w_{ij} = \eta(post_i^+ - post_i^-)pre_j^-$
- more when doing Boltzmann machines...

Convolutional networks

Neocognitron
[Fukushima, 1980, Fukushima, 1988, LeCun et al., 1990]

- 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



[Riesenhuber and Poggio, 1999]

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.

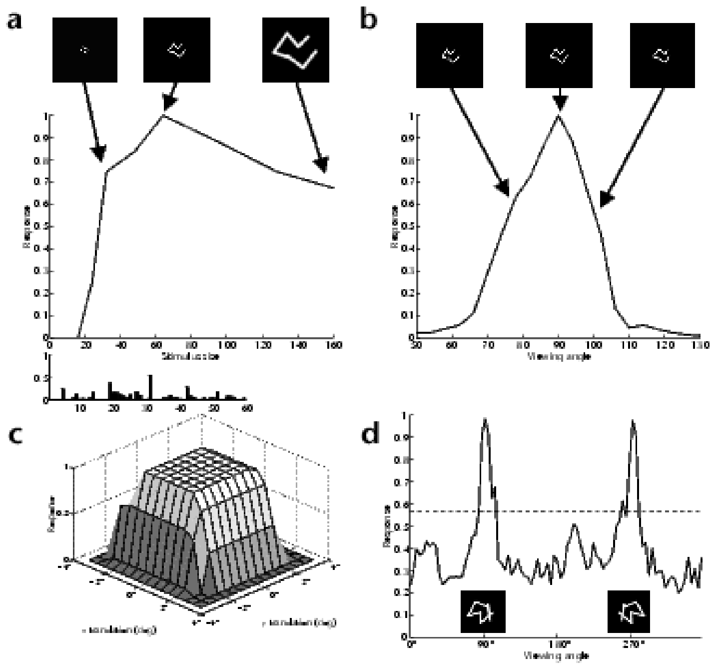
Rather than learning, take refuge in having many, many cells.
(Cover, 1965)*A complex pattern-classification problem, cast in a high-dimensional space non-linearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.*

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.

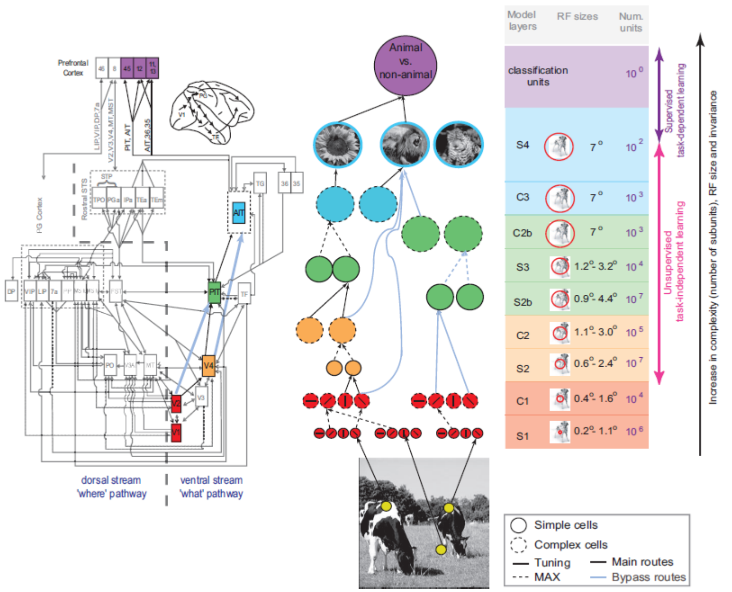


[Riesenhuber and Poggio, 1999]

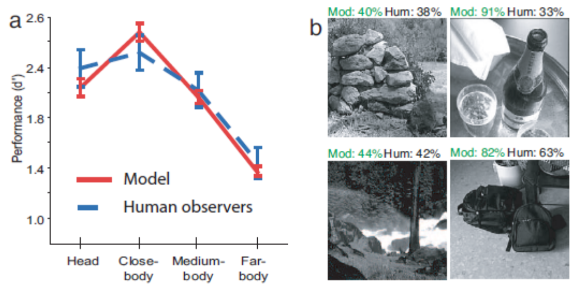
HMAX model: Results

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

More recent version



[Serre et al., 2007]



- Use real images as inputs
- S-cells convolution,e.g. $h = (\frac{\sum_i w_i x_i}{\kappa + \sqrt{\sum_i w_i^2}})$, $y = g(h)$.
- C-cell soft-max pooling $h = \frac{\sum x_i^{q+1}}{\kappa + \sum_k x_i^q}$ (some support from biology for such pooling)
- Some unsupervised learning between layers [Serre et al., 2005]

HMAX model: Results

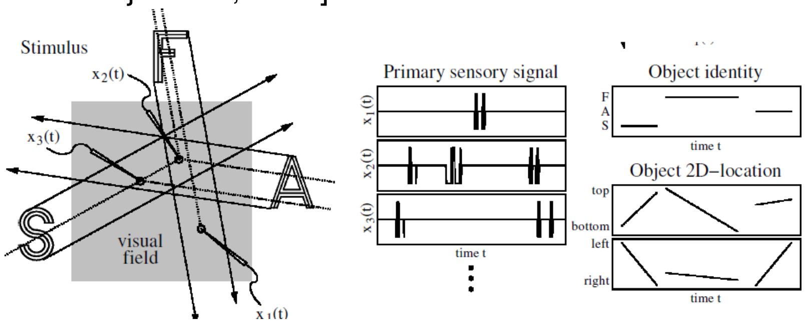
- Localization can be achieved by using a sliding-window method
- Claimed as a model on a “rapid categorization task”, where back-projections are inactive
- Performance similar to human performance on flashed (20ms) images
- The model doesn’t do segmentation (as opposed to bounding boxes)

Learning invariances

- Hard-code (convolutional network) http://yann.lecun.com/exdb/lenet/
- Supervised learning (show samples and require same output)
- 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[Földiák, 1991]
  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 [Rolls and Deco, 2002], Slow feature analysis.

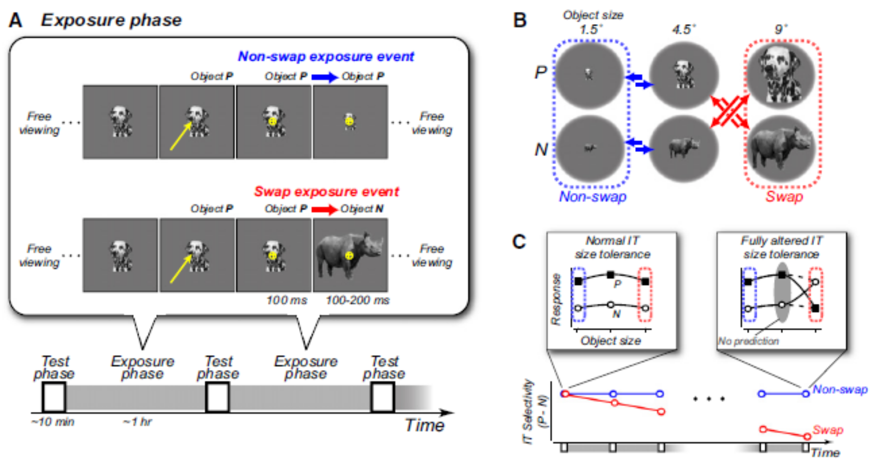
Slow feature analysis

Find slow varying features, these are likely relevant [Wiskott and Sejnowski, 2002]



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 [Li and DiCarlo, 2010]



Including top-down interaction

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

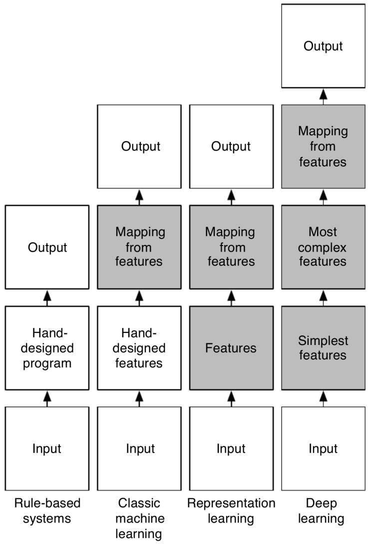
[Epshtein et al., 2008]

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.

AI



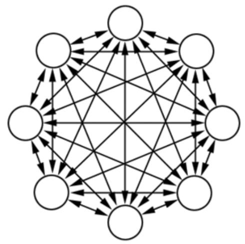
[Bengio et al., 2014]

Role of representation

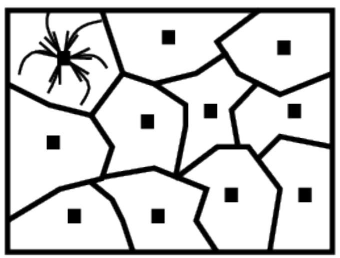
- Finding good representation solves most problems 90%
- Similarly, bad representation can make problem very hard.
- E.g. odd/even number categorization using base-2 (only last bit matters) vs base-3 (all bits matter) representation.
- E.g. recognition of images after fixed, random scrambling is difficult for humans. This is the task naive MLPs are faced with.

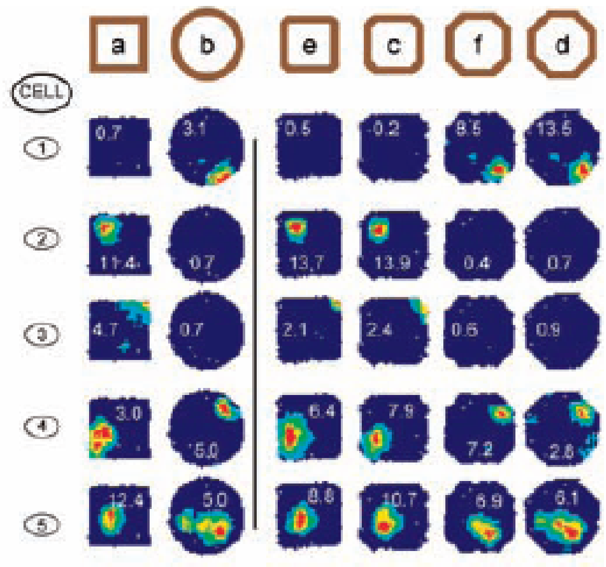
- MLPs have no dynamics
- Recurrent networks are dynamic. Could be steady state(s), periodic, or chaotic. With symmetric weights there can only be fixed points (point or line attractors).
- In recurrent networks it is much harder to find weights to be altered. Often restrict to cases where dynamics has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

- All to all connected network (can be relaxed)
- Binary units $s_i = \pm 1$, or rate with sigmoidal transfer.
- Dynamics $s_i(t+1) = sign[\sum_j w_{ij} s_j(t)]$ or continuous version $\frac{d\mathbf{r}(t)}{dt} = -\mathbf{r} + g(W\mathbf{r}(t))$.
- Using symmetric weights $w_{ij} = w_{ji}$, we can define energy $E = -\frac{1}{2}\sum_{ij} s_i w_{ij} s_j$.



- Under these conditions network moves from initial condition (stimulus, $\mathbf{s}(t=0) = \mathbf{x}$) into the closest attractor state ('memory') and stays there.
- Auto-associative, pattern completion
- Simple (suboptimal) learning rule: $w_{ij} = \sum_\mu^M x_i^\mu x_j^\mu$ (μ indexes patterns $\mathbf{x}^\mu$).

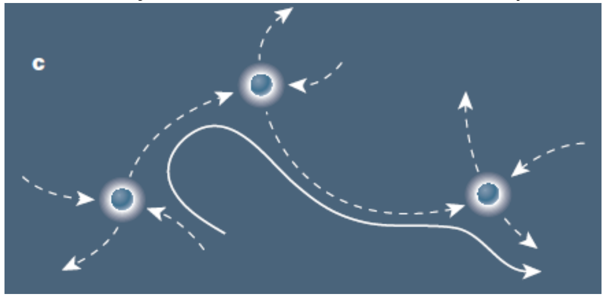




Indirect experimental evidence using maze deformation[Wills et al., 2005]

Winner-less competition

How to escape from attractor states?
Noise, asymmetric connections, adaptation.

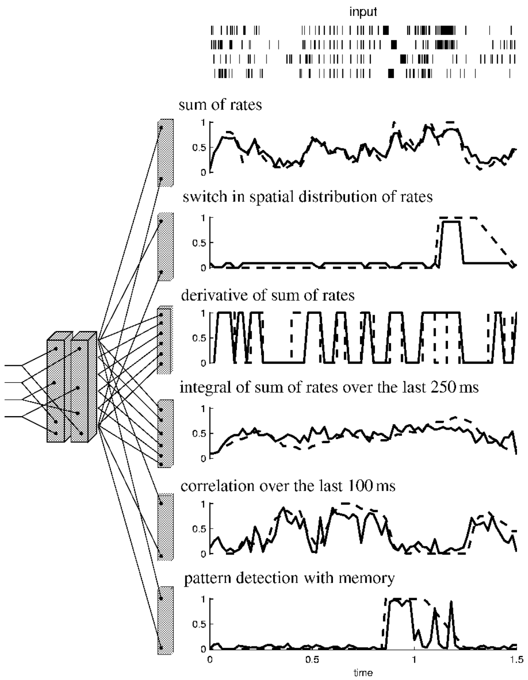


From [Ashwin and Timme, 2005].

Liquid state machines

[Maass et al., 2002]

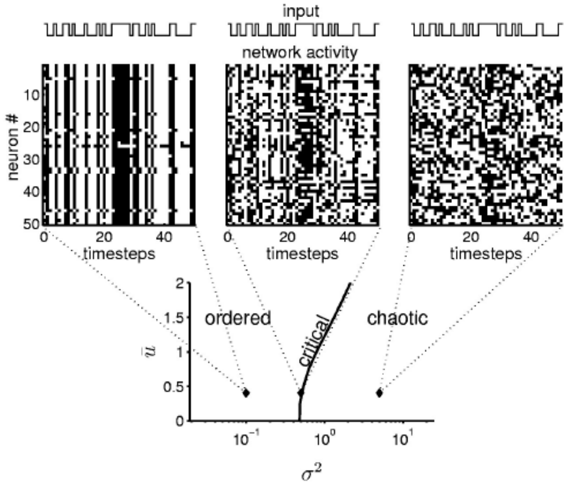
- Motivation: arbitrary spatio-temporal computation without precise design.
- Create pool of spiking neurons with random connections.
- Results in very complex dynamics if weights are strong enough
- Similar to echo state networks (but those are rate based).
- Both are known as reservoir computing
-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



Various functions can be implemented by varying readout.

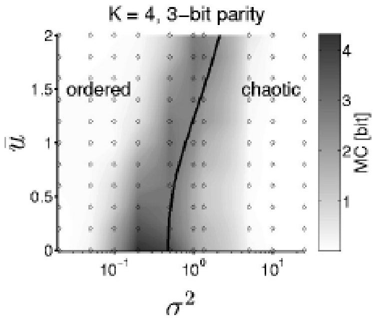
Optimal reservoir?

Best reservoir has rich yet predictable dynamics.
Edge of Chaos [Bertschinger and Natschlaeger, 2004]



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

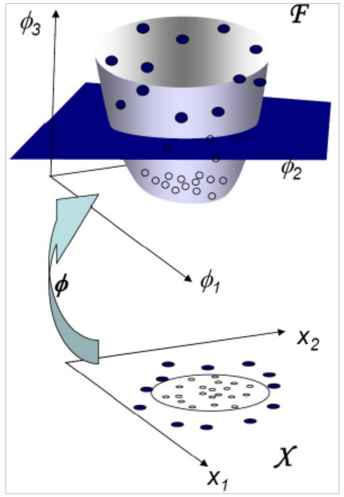
Optimal reservoir?



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 [van Vreeswijk and Sompolinsky, 1996]
- In real brains: ?

Relation to Support Vector Machines

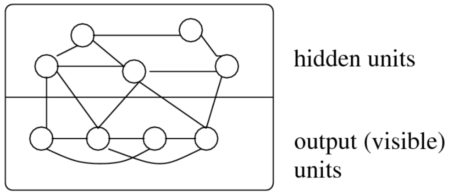


Map problem in to high dimensional space $\mathcal{F}$; there it often becomes linearly separable.
This can be done without much computational overhead (kernel trick).

Boltzmann machines

Hopfield network is not perfect. It is impossible to learn only $(1,1,-1),(-1,-1,-1),(1,-1,1),(-1,1,1)$ but not $(-1,-1,1),(1,1,1),(-1,1,-1),(1,-1,1)$ (XOR again)...
Because $\langle x_i \rangle = \langle x_i x_j \rangle = 0$
Boltmann machines have ± 1 units and include two, somewhat unrelated, modifications:

- Introduce hidden units, these can extract abstract features.



Boltzmann machines

- Stochastic updating: $p(s_i = 1) = \frac{1}{1+e^{-2\beta E_i}}$
  $E_i = \sum_j w_{ij} s_j - \theta_i$, $E = \sum_i E_i$.
  $T = 1/\beta$ is temperature (set to some arbitrary value).
- Boltzmann distribution

$$P(\mathbf{s}) = \frac{\exp(-\beta E(\mathbf{s}))}{Z}$$

where $Z = \sum_{\mathbf{s}} \exp(-\beta E(\mathbf{s}))$

Boltzmann machines

- Boltzmann machine learns arbitrary $P(\mathbf{v})$.
- Can thus be used for auto-association (pattern completion)
- Or, by labelling some visible units as inputs and others as output, can be used as if it were a associator like an MLP.

Learning in Boltzmann machines

The generated probability for state $\mathbf{s}_\alpha$, after equilibrium is reached, is given by the Boltzmann distribution

$$P_\alpha = \frac{1}{Z} \sum_\gamma e^{-\beta H_{\alpha\gamma}}$$

$$H_{\alpha\gamma} = -\frac{1}{2} \sum_{ij} w_{ij} s_i s_j$$

$$Z = \sum_{\alpha\beta} e^{-\beta H_{\alpha\gamma}}$$

where α labels states of visible units, γ the hidden states.

As in other generative models, we match true distribution to generated one. Minimize KL divergence between input and generated distribution.

$$KL = \sum_\alpha G_\alpha \log \frac{G_\alpha}{P_\alpha}$$

Minimize to get [Ackley et al., 1985, Hertz et al., 1991]

$$\Delta w_{ij} = \eta\beta[\langle s_i s_j \rangle_{clamped} - \langle s_i s_j \rangle_{free}]$$

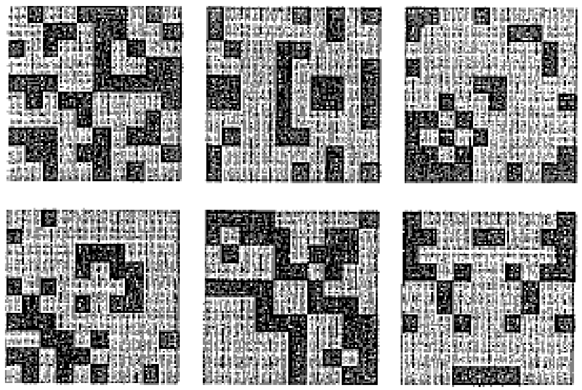
(note, $w_{ij} = w_{ji}$)
Wake ('clamped') phase vs. sleep ('dreaming') phase

- Clamped phase: Hebbian type learning. Average over input patterns and hidden states.
- Sleep phase: unlearn erroneous correlations.

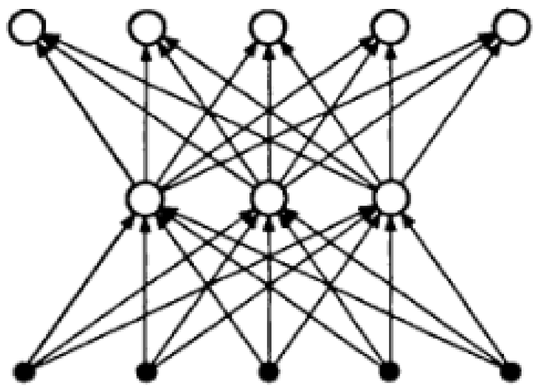
The hidden units will 'discover' statistical regularities.// Biology of phases unknown.

Boltzmann machines: applications

- Shifter circuit.
- Learning symmetry [Sejnowski et al., 1986]. Create a network that categorizes horizontal, vertical, diagonal symmetry (2nd order predicate).



Boltzmann machines: auto-encoders

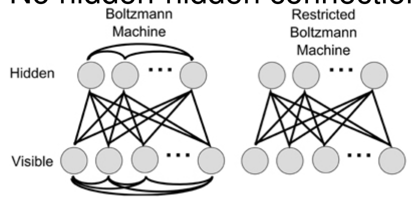


Autoencoders: Minimize $E(input, output)$
Fewer hidden units than input units: find optimal compression (PCA).
More hidden units: impose for instance sparseness.

Restricted Boltzmann

Need for multiple relaxation runs for every weight update (triple loop), makes training Boltzmann networks very slow.
Speed up learning in restricted Boltzmann:

- No hidden-hidden connections



- Don't wait for the sleep state to fully settle, one step is enough.
- Stack multiple layers (deep-learning)
- Application: high quality auto-encoder (i.e. compression) [Hinton and Salakhutdinov, 2006]

[also good webtalks/tutorials by Hinton on this]

Sparse deep belief net model for visual area V2

[Lee et al., 2008]

- Consider an RBM with Gaussian visible units

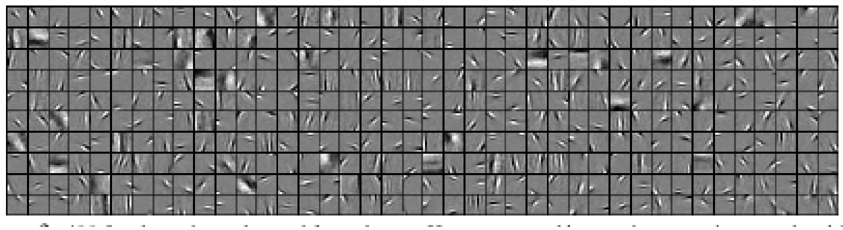
$$E(\mathbf{u}, \mathbf{v}) = \frac{1}{2\sigma^2} \sum_i u_i^2 - \frac{1}{\sigma^2} \left(\sum_i c_i u_i + \sum_j b_j v_j + \sum_{i,j} u_i v_j w_{ij} \right)$$

- $p(u_i|\mathbf{v}) \sim N(c_i + \sum_j w_{ij} v_j, \sigma^2)$
- Also impose a *sparsity prior* on the hidden units, with target sparseness p

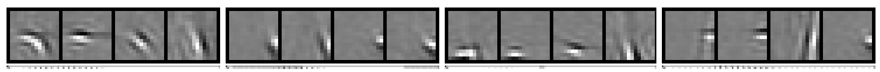
$$\sum_j ||p - \frac{1}{m} \sum_{k=1}^{m} \mathbb{E}[v_j^{(k)} | \mathbf{u}^{(k)}]||^2$$

- Layer 2 trained after layer 1 has learned (DBN)

First layer filters



Second layer: each unit "looks at" a small number of first layer units, e.g.



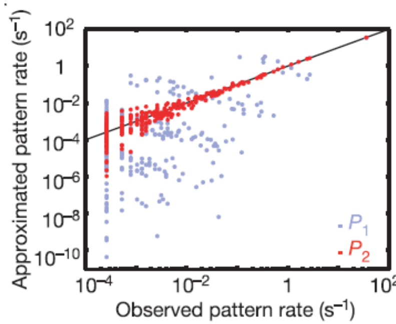
The leftmost patch in each group is a visualization of the model V2 basis, obtained by taking a weighted linear combination of the first layer bases to which it is connected.

Properties of "V2" units can be compared to neural data.

Recurrent models: Ising model of neural activity

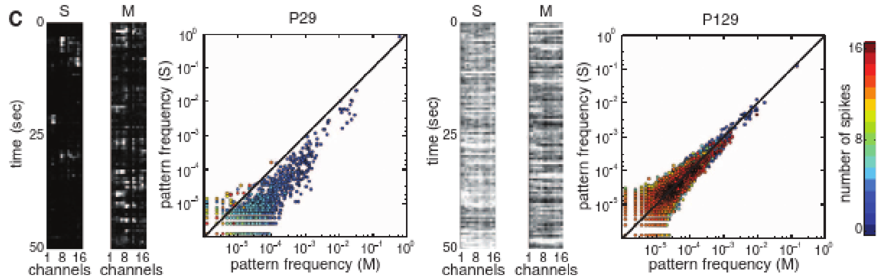
To describe data of retinal network, use Ising model
[Schneidman et al., 2006]

$$P(\mathbf{r}) = \frac{1}{Z} e^{(-\sum_i h_i r_i - \sum_{ij} w_{ij} r_i r_j)}$$



(But maybe it does not work well in large networks [Roudi et al., 2009])

Generative models



[Berkes et al., 2011]
During development spontaneous activity matched stimulus-evoked activity better and better.

References I

- Ackley, D., Hinton, G., and Sejnowski, T. (1985).
  A learning algorithm for Boltzmann machines,.
  Cognitive Science, 9:147–169.
- Ashwin, P. and Timme, M. (2005).
  Nonlinear dynamics: when instability makes sense.
  Nature, 436(7047):36–37.
- Bengio, Y., Goodfellow, I. J., and Courville, A. (2014).
  Deep learning.
  Book in preparation for MIT Press.
- Berkes, P., Orban, G., Lengyel, M., and Fiser, J. (2011).
  Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment.
  Science, 331(6013):83–87.
- Bertschinger, N. and Natschlaeger, T. (2004).
  Real-time computation at the edge of chaos in recurrent neural networks.
  Neural Comput, 16(7):1413–1436.
- Epshtein, B., Lifshitz, I., and Ullman, S. (2008).
  Image interpretation by a single bottom-up top-down cycle.
  Proc Natl Acad Sci U S A, 105(38):14298–14303.

References II

- Földiák, P. (1991).
  Learning invariance from transformation sequences.
  Neural Comp., 3:194–200.
- Fukushima, K. (1980).
  Neocognitron: A self-organising multi-layered neural network.
  Biol Cybern, 20:121–136.
- Fukushima, K. (1988).
  Neocognitron: A hierarchical neural network capable of visual pattern recognition.
  Neural Networks, 1:119–130.
- Hertz, J., Krogh, A., and Palmer, R. G. (1991).
  Introduction to the theory of neural computation.
  Perseus, Reading, MA.
- Hinton, G. E. and Salakhutdinov, R. R. (2006).
  Reducing the dimensionality of data with neural networks.
  Science, 313(5786):504–507.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990).
  Handwritten digit recognition with a back-propagation network.
  In *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann.

References III

Lee, H., Ekanadham, C., and Ng, A. (2008).
Sparse deep belief net model for visual area v2.
NIPS, 20.

Li, N. and DiCarlo, J. J. (2010).
Unsupervised natural visual experience rapidly reshapes size-invariant object representation in inferior temporal cortex.
Neuron, 67(6):1062–1075.

Logothetis, N. K. and Sheinberg, D. L. (1996).
Visual object recognition.
Annu Rev Neurosci, 19:577–621.

Maass, W., Natschlaeger, T., and Markram, H. (2002).
Real-time computing without stable states: a new framework for neural computation based on perturbations.
Neural Comput, 14(11):2531–2560.

Riesenhuber, M. and Poggio, T. (1999).
Hierarchical models of object recognition in cortex.
Nat. Neuro., 2:1019–1025.

Rolls, E. T. and Deco, G. (2002).
Computational neuroscience of vision.
Oxford.

References IV

Roudi, Y., Aurell, E., and Hertz, J. A. (2009).
Statistical physics of pairwise probability models.
Front Comput Neurosci, 3(22).

Schneidman, E., Berry, M. J., Segev, R., and Bialek, W. (2006).
Weak pairwise correlations imply strongly correlated network states in a neural population.
Nature, 440(7087):1007–1012.

Sejnowski, Kienker, and Hinton (1986).
Learning symmetry groups with hidden units: Beyond the perceptron.
Physica D, 22:260.

Serre, T., Kouh, M., Cadieu, C., Knoblich, U., Kreiman, G., and Poggio, T. (2005).
A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex.
MIT AI Memo 2005-036.

Serre, T., Oliva, A., and Poggio, T. (2007).
A feedforward architecture accounts for rapid categorization.
Proc Natl Acad Sci U S A, 104(15):6424–6429.

van Vreeswijk, C. and Sompolinsky, H. (1996).
Chaos in neuronal networks with balanced excitatory and inhibitory activity.
Science, 274:1724–1726.

References V

Wills, T. J., Lever, C., Cacucci, F., Burgess, N., and O'Keefe, J. (2005).
Attractor dynamics in the hippocampal representation of the local environment.
Science, 308(5723):873–876.

Wiskott, L. and Sejnowski, T. J. (2002).
Slow feature analysis: Unsupervised learning of invariances.
Neural Comp., 15:715–770.