Neural Networks and Deep Learning I

CSCI 5622

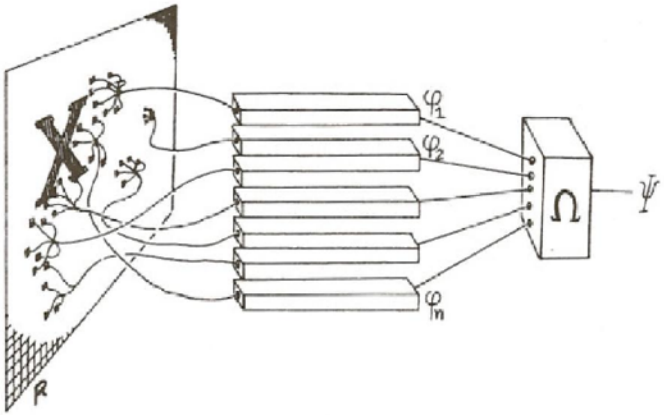
Fall 2015

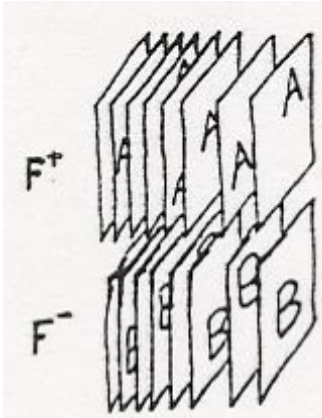
A Brief History Of Machine Learning

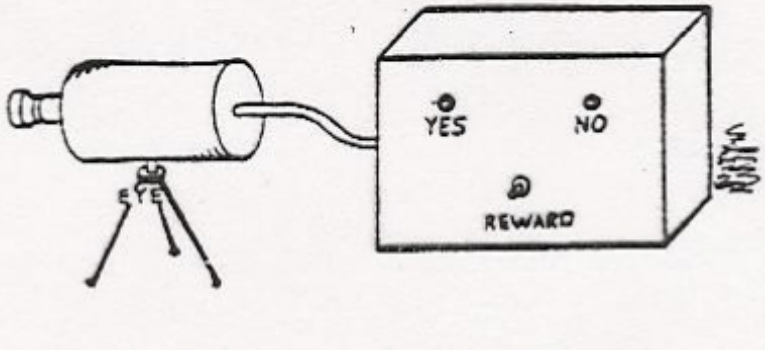
1962

Frank Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*

Perceptron can learn anything you can program it to do.





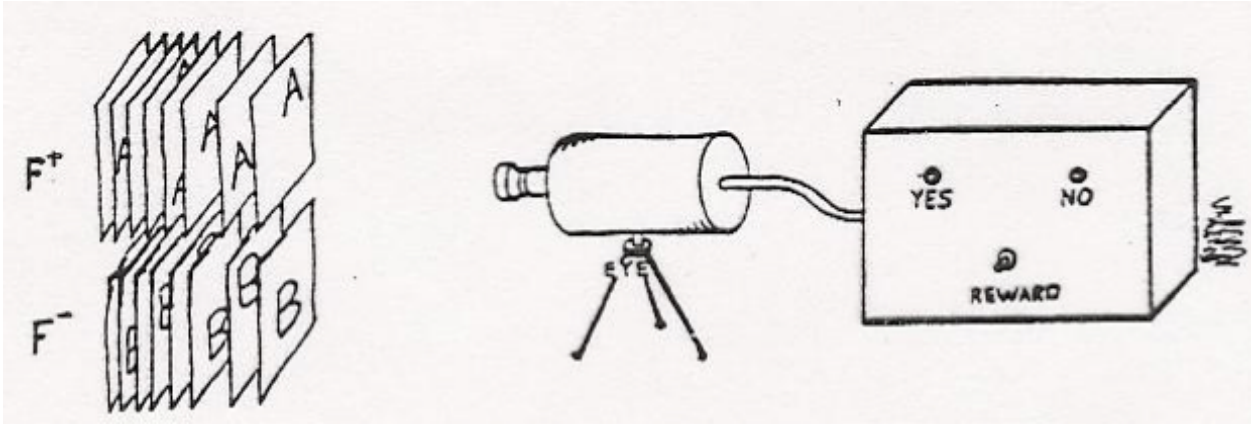


A Brief History Of Machine Learning

1969

Minsky & Papert, *Perceptrons: An introduction to computational geometry*

There are many things a perceptron can't in principle learn to do



A Brief History Of Machine Learning

1970-1985

Attempts to develop symbolic rule discovery algorithms

1986

Rumelhart, Hinton, & Williams, *Back propagation*

Overcame many of the Minsky & Papert objections

Neural nets popular in cog sci and AI

circa 1990

A Brief History Of Machine Learning

1990-2005

Bayesian approaches

- take the best ideas from neural networks – statistical computing, statistical learning

Support-Vector Machines

- convergence proofs (unlike neural nets)

A few old timers keep playing with neural nets

- Hinton, LeCun, Bengio

Neural nets banished from NIPS!

A Brief History Of Neural Networks

2005-2010

Attempts to resurrect neural nets with

- unsupervised pretraining
- probabilistic neural nets
- alternative learning rules

A Brief History Of Neural Networks

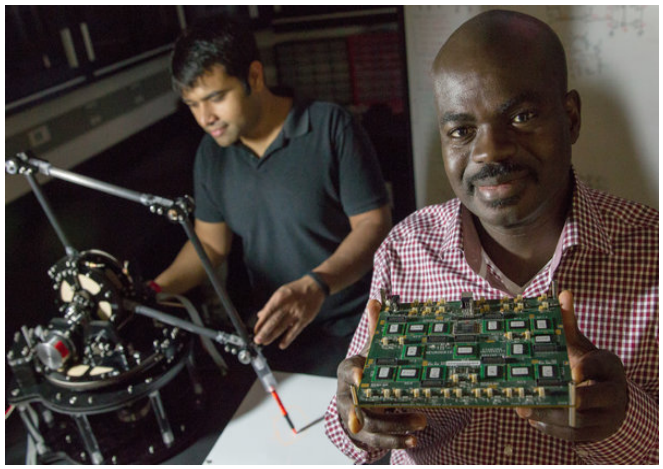
2010-present

Most of the alternative techniques discarded in favor of 1980's style neural nets with …

- **lots more training data**
- **lots more computing cycles**
- **a few important tricks that improve training and generalization (mostly from Hinton)**
- **rebranding: *Deep Learning***

2013

Brainlike Computers, Learning From Experience



Erin Lubin/The New York Times

Kwabena Boahen holding a biologically inspired processor attached to a robotic arm in a laboratory at Stanford University.

By JOHN MARKOFF
Published: December 28, 2013 | 117 Comments

PALO ALTO, Calif. — Computers have entered the age when they are able to learn from their own mistakes, a development that is about to turn the digital world on its head.

The first commercial version of the new kind of computer chip is scheduled to be released in 2014. Not only can it automate tasks that now require painstaking programming — for example, moving a robot's arm smoothly and efficiently — but it can also sidestep and even tolerate errors potentially making the term "computer crash" obsolete.

The new computing approach, already in use by some large technology companies, is based on the biological nervous system, specifically on how neurons react to stimuli and connect with other neurons to interpret information. It allows computers to absorb new information while carrying out a task, and adjust what they do based on the changing signals.

In coming years, the approach will make possible a new generation of artificial intelligence systems that will perform some functions that humans do with ease: see, speak, listen, navigate, manipulate and control. That can hold enormous consequences for tasks like facial and speech recognition, navigation and planning, which are still in elementary stages and rely heavily on human programming.

Designers say the computing style can clear the way for robots that can safely walk and drive in the physical world, though a thinking or conscious computer, a staple of science fiction, is still far off on the digital horizon.

"We're moving from engineering computing systems to something that has many of the characteristics of biological computing," said Larry Smarr, an astrophysicist who directs the California Institute for Telecommunications and Information Technology, one of many research centers devoted to developing these new kinds of computer circuits.

Conventional computers are limited by what they have been programmed to do. Computer vision systems, for example, only "recognize" objects that can be identified by the statistics-oriented algorithms programmed into them. An algorithm is like a recipe, a set of step-by-step instructions to perform a calculation.

But last year, Google researchers were able to get a machine-learning algorithm, known as a neural network, to perform an identification task without supervision. The network scanned a database of 10 million images, and in doing so trained itself to recognize cats.

In June, the company said it had used those neural network techniques to develop a new search service to help customers find specific photos more accurately.

The new approach, used in both hardware and software, is being driven by the explosion of scientific knowledge about the brain. Kwabena Boahen, a computer scientist who leads Stanford's Brains in Silicon research program, said that is also its limitation, as scientists are far from fully understanding how brains function.

"We have no clue," he said. "I'm an engineer, and I build things. There are these highfalutin theories, but give me one that will let me build something."

Until now, the design of computers was dictated by ideas originated by the mathematician John von Neumann about 65 years ago. Microprocessors perform operations at lightning speed, following instructions programmed using long strings of 1s and 0s. They generally store that information separately in what is known, colloquially, as memory, either in the processor itself, in adjacent storage chips or in higher capacity magnetic disk drives.

The data — for instance, temperatures for a climate model or letters for word processing — are shuttled in and out of the processor's short-term memory while the computer carries out the programmed action. The result is then moved to its main memory.

The new processors consist of electronic components that can be connected by wires that mimic biological synapses. Because they are based on large groups of neuron-like elements, they are known as neuromorphic processors, a term credited to the California Institute of Technology physicist Carver Mead, who pioneered the concept in the late 1980s.

They are not "programmed." Rather the connections between the circuits are "weighted" according to correlations in data that the processor has already "learned." Those weights are then altered as data flows in to the chip, causing them to change their values and to "spike." That generates a signal that travels to other components and, in reaction, changes the neural network, in essence programming the next actions much the same way that information alters human thoughts and actions.

"Instead of bringing data to computation as we do today, we can now bring computation to data," said Dharmendra Modha, an I.B.M. computer scientist who leads the company's cognitive computing research effort. "Sensors become the computer, and it opens up a new way to use computer chips that can be everywhere."

The new computers, which are still based on silicon chips, will not replace today's computers, but will augment them, at least for now. Many computer designers see them as coprocessors, meaning they can work in tandem with other circuits that can be embedded in smartphones and in the giant centralized computers that make up the cloud. Modern computers already consist of a variety of coprocessors that perform specialized tasks, like producing graphics on your cellphone and converting visual, audio and other data for your laptop.

One great advantage of the new approach is its ability to tolerate glitches. Traditional computers are precise, but they cannot work around the failure of even a single transistor. With the biological designs, the algorithms are ever changing, allowing the system to continuously adapt and work around failures to complete tasks.

Traditional computers are also remarkably energy inefficient, especially when compared to actual brains, which the new neurons are built to mimic.

I.B.M. announced last year that it had built a supercomputer simulation of the brain that encompassed roughly 10 billion neurons — more than 10 percent of a human brain. It ran about 1,500 times more slowly than an actual brain. Further, it required several megawatts of power, compared with just 20 watts of power used by the biological brain.

Running the program, known as Compass, which attempts to simulate a brain, at the speed of a human brain would require a flow of electricity in a conventional computer that is equivalent to what is needed to power both San Francisco and New York, Dr. Modha said.

I.B.M. and Qualcomm, as well as the Stanford research team, have already designed neuromorphic processors, and Qualcomm has said that it is coming out in 2014 with a commercial version, which is expected to be used largely for further development. Moreover, many universities are now focused on this new style of computing. This fall the National Science Foundation financed the Center for Brains, Minds and Machines, a new research center based at the Massachusetts Institute of Technology, with Harvard and Cornell.

The largest class on campus this fall at Stanford was a graduate level machine-learning course covering both statistical and biological approaches, taught by the computer scientist Andrew Ng. More than 760 students enrolled. "That reflects the zeitgeist," said Terry Sejnowski, a computational neuroscientist at the Salk Institute, who pioneered early biologically inspired algorithms. "Everyone knows there is something big happening, and they're trying find out what it is."

A version of this article appears in print on December 29, 2013, on page A1 of the New York edition with the headline: Brainlike Computers, Learning From Experience.

Varrior' a ations

Deadly Heat Is Forecast in Persian Gulf by 2100

MATTER
DNA of Ancient Children Offers Clues on How People Settled the...

Saving a Rare Tree Away

SCIENCE

Scientists See Promise in Deep-Learning Programs

By JOHN MARKOFF NOV. 23, 2012

Email

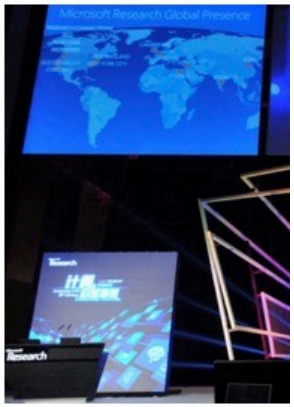
Share

Tweet

Save

More

Using an artificial intelligence technique inspired by theories about how the brain recognizes patterns, technology companies are reporting startling gains in fields as diverse as computer vision, speech recognition and the identification of promising new molecules for designing drugs.



A voice recognition program translat scientist, into Mandarin Chinese. Hao

The advances have led to widespread enthusiasm among researchers who design software to perform human activities like seeing, listening and thinking. They offer the promise of machines that converse with humans and perform tasks like driving cars and working in factories, raising the specter of automated robots that could replace human workers.

The technology, called deep learning, has already been put to use in services like Apple's Siri virtual personal assistant, which is based on Nuance Communications' speech recognition service, and in Google's Street View, which uses machine vision to identify specific addresses.

But what is new in recent months is the growing speed and accuracy of deep-learning programs, often called artificial neural networks or just "neural nets" for their resemblance to the neural connections in the brain.

A student team led by the computer scientist Geoffrey E. Hinton used deep-learning technology to design software.

"There has been a number of stunning new results with deep-learning methods," said Yann LeCun, a computer scientist at New York University who did pioneering research in handwriting recognition at Bell Laboratories. "The kind of jump we are seeing in the accuracy of these systems is very rare indeed."

Artificial intelligence researchers are acutely aware of the dangers of being overly optimistic. Their field has long been plagued by outbursts of misplaced enthusiasm followed by equally striking declines.

In the 1960s, some computer scientists

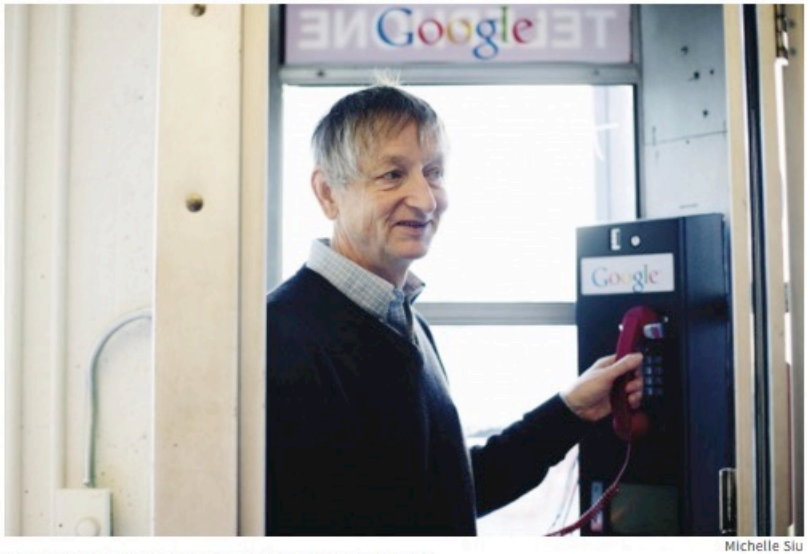
11/23/2012

THE CHRONICLE OF HIGHER EDUCATION

THE CHRONICLE REVIEW

The Believers

The hidden story behind the code that runs our lives



Michelle Siu

Geoffrey Hinton splits his time between the U. of Toronto and Google.

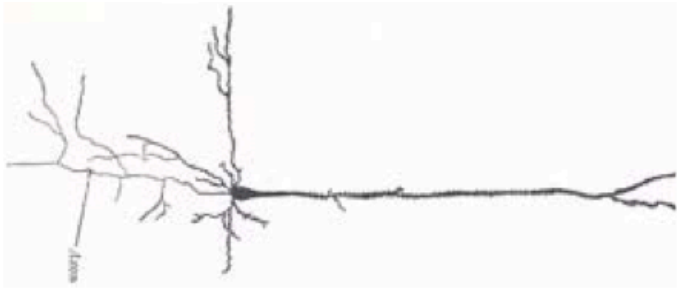
By Paul Voosen | FEBRUARY 23, 2015

Magic has entered our world. In the pockets of many Americans today are thin black slabs that, somehow, understand and anticipate our desires. Linked to the digital cloud and satellites beyond, churning through personal data, these machines listen and assist, decoding our language, viewing and labeling reality with their cameras. This summer, as I walked to an appointment at the University of Toronto, stepping out of my downtown hotel into brisk hints of fall, my phone already had directions at hand. I asked where to find coffee on the way. It told me. What did the machine know? How did it learn? A gap broader than any we've known has opened between our use of technology and our understanding of it. How did the machine work? As I would discover, no one could say for certain. But as I walked with my coffee, I was on the way to meet the man most qualified to bridge the gap between what the machine knows and what you know.

Geoffrey Hinton is a torchbearer, an academic computer scientist who has spent his career, along with a small band of fellow travelers, devoted to an idea of artificial intelligence that has been discarded multiple times over. A brilliant but peripheral figure. A believer. A brusque coder who had to hide his ideas in obscure language to get them past peer review. A devotee of the notion that, despite how little we understand the brain, even a toy model of it could present more computational power and flexibility than the rigid logic or programmed knowledge of traditional artificial intelligence. A man whose ideas and algorithms might now help power nearly every aspect of our lives. A guru of the artificial neural network.

Such networks, which have been rebranded "deep learning," have had an unparalleled ascent over the past few years. They've hit the front page of *The New York Times*. Adept at

Modeling Individual Neurons



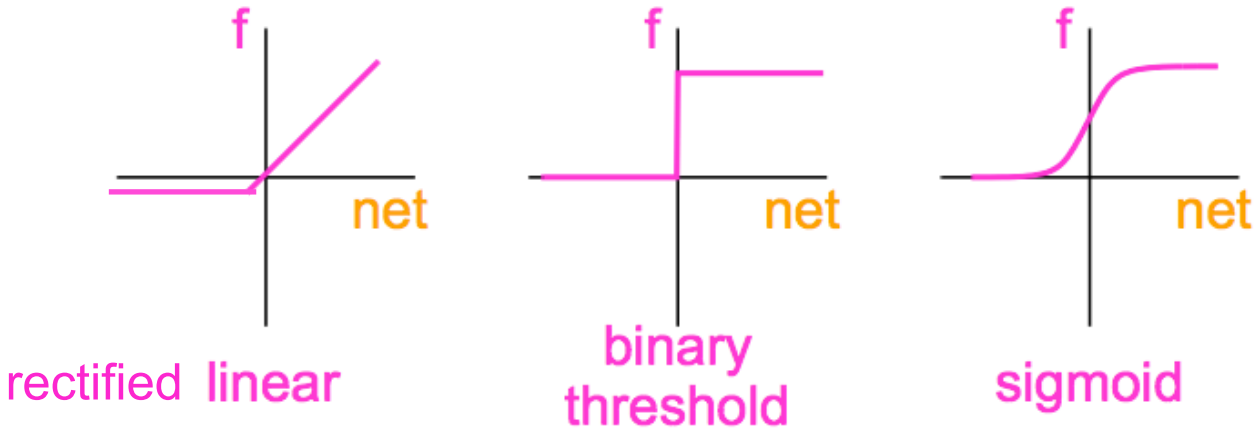
flow of information

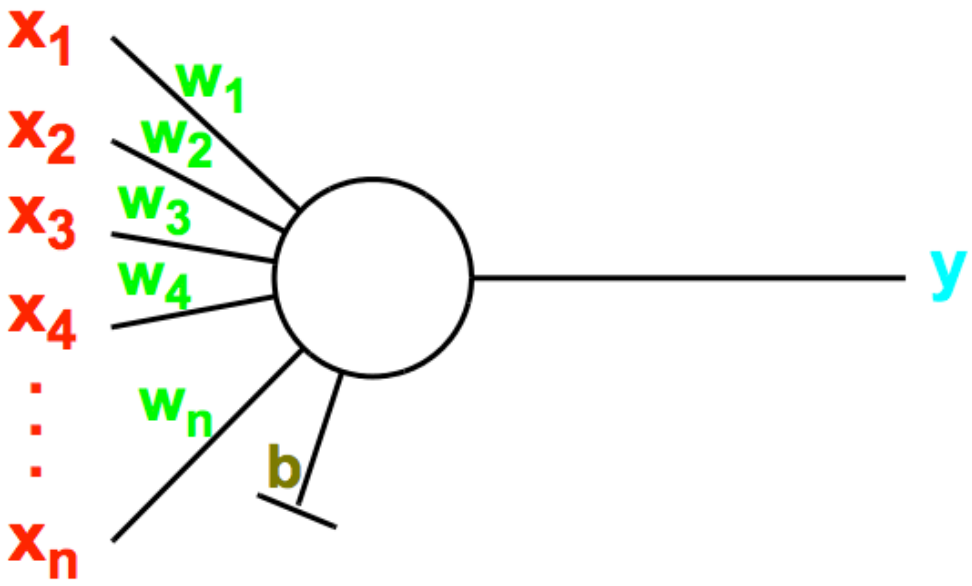
Modeling Individual Neurons

Activation function

$$net = \sum_i w_i x_i + b$$

$$y = f(net)$$





input activities

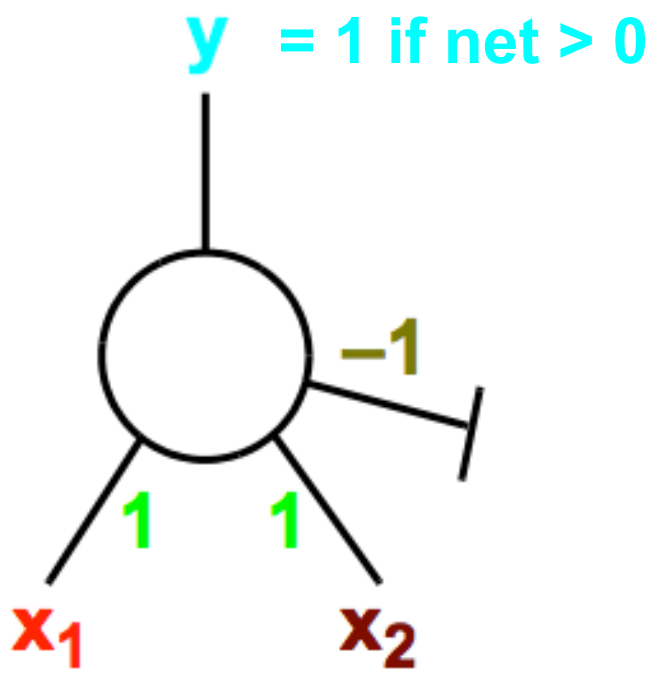
weights and bias

output activity

Computation With A Binary Threshold Unit

"And" gate

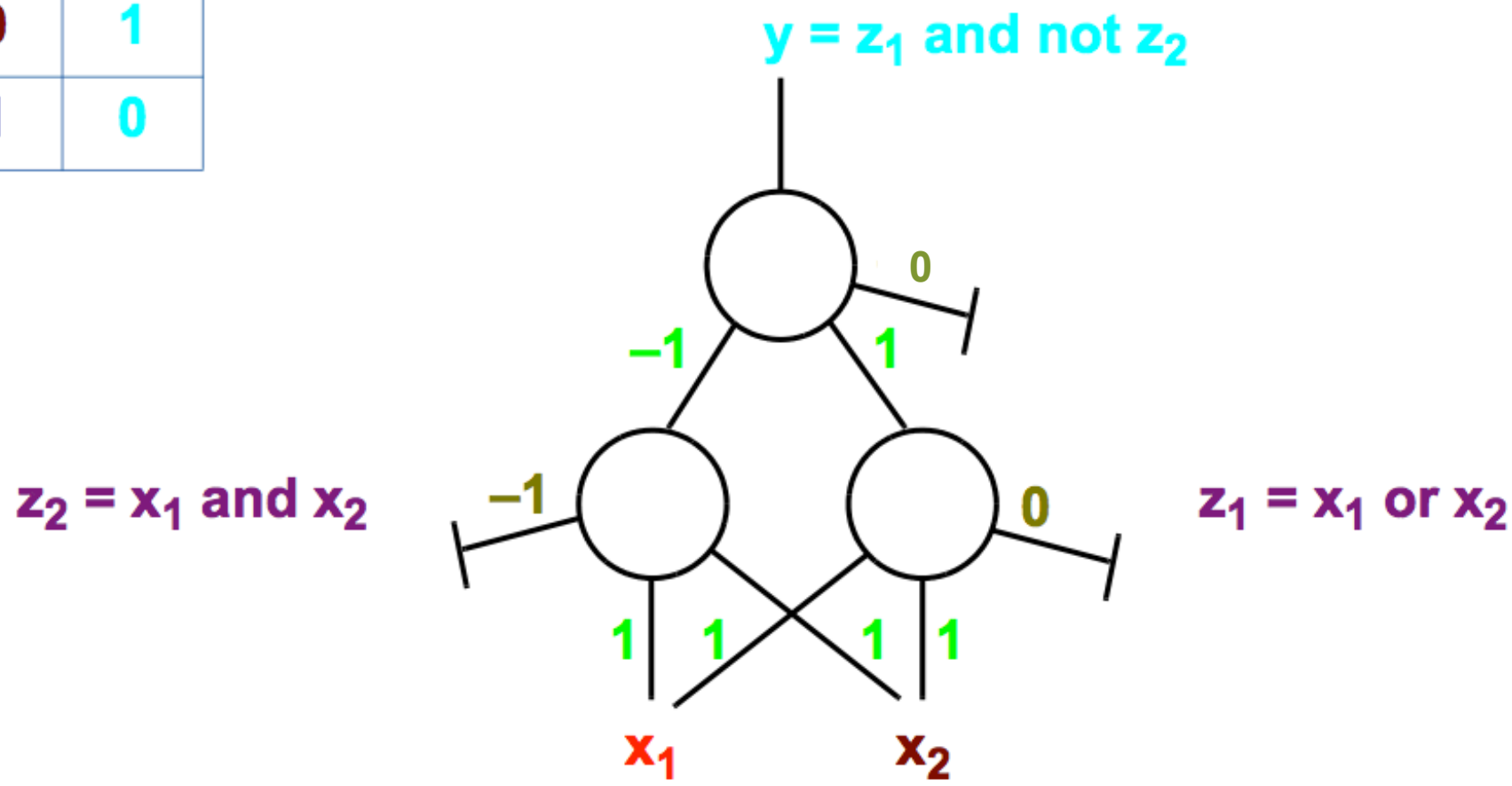
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1



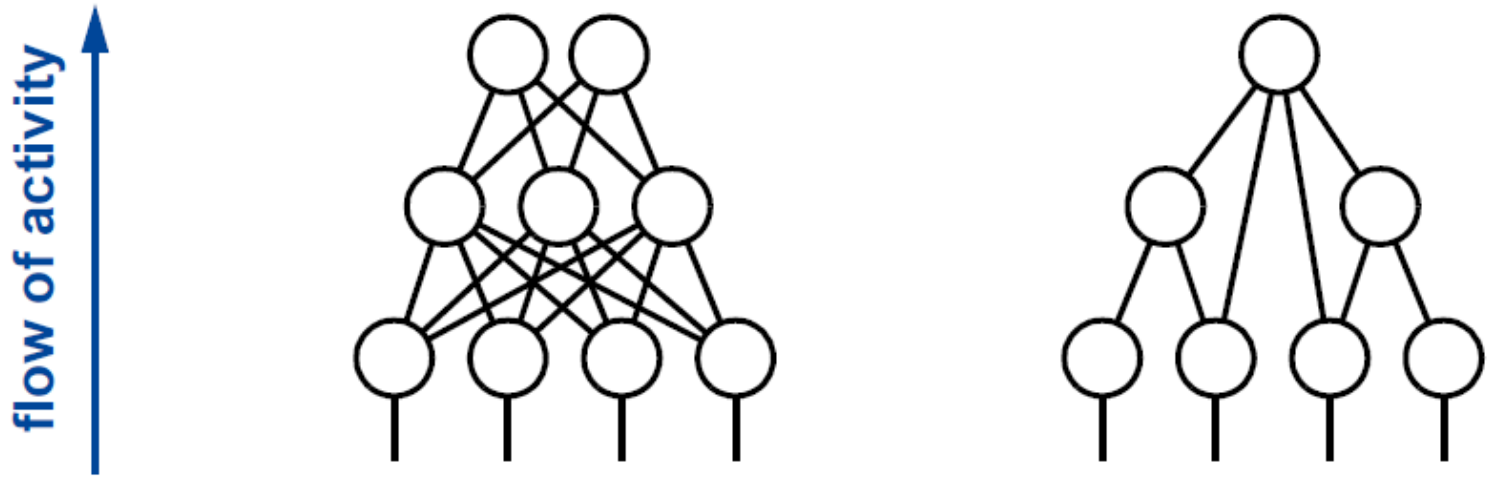
Computation With A Binary Threshold Unit

"Exclusive or" gate

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



Feedforward Architectures



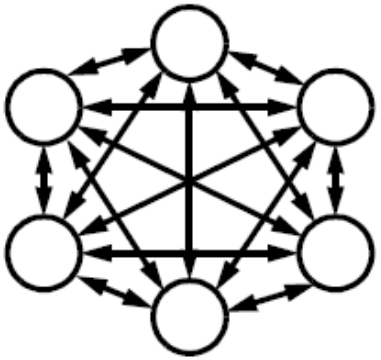
Activation flows in one direction; no closed loops

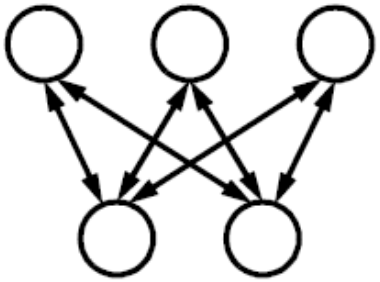
Performs association from input pattern to output pattern

big, hairy, stinky => run away
small, round, orange => eat
big, round, soft => eat
small, orange, hairy => run away
stinky, yellow => eat

Learning: adjust connections to achieve input-output mapping

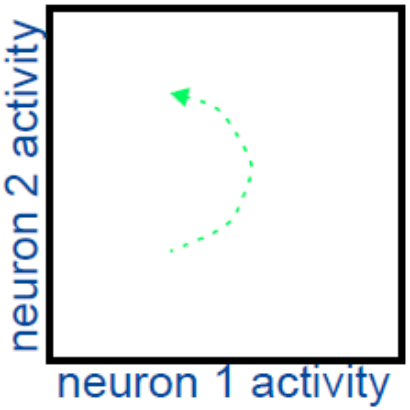
Recurrent Architectures



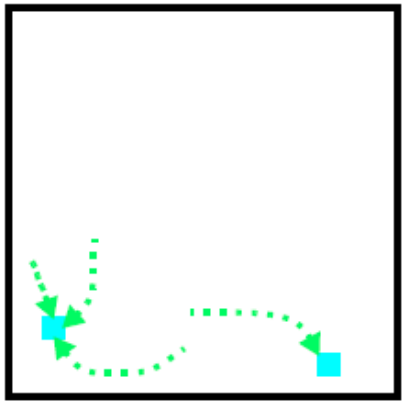


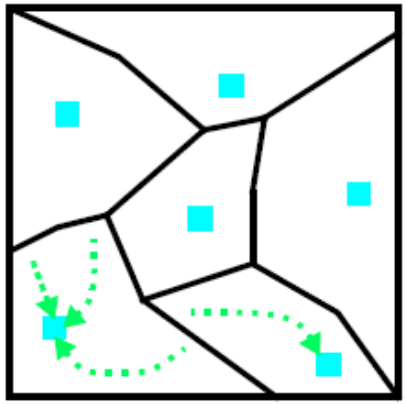
Achieves *best interpretation* of partial or noisy patterns, e.g.,
MAR – – M – LLOW

State space dynamics



Attractor dynamics





Learning: establishes new attractors and shifts attractor basin boundaries

Let's Get Serious…

Training data

$$\{(\boldsymbol{x}^1,\boldsymbol{d}^1),(\boldsymbol{x}^2,\boldsymbol{d}^2),\ldots,(\boldsymbol{x}^p,\boldsymbol{d}^p)\}$$

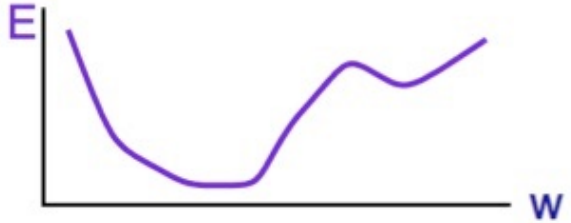
big, hairy, stinky => run away

Network model

$$\boldsymbol{y}^\alpha = f_w(\boldsymbol{x}^\alpha)$$

Objective function

$$E = \sum_\alpha \sum_k (d_k^\alpha - y_k^\alpha)^2$$



Learning rule

$$\Delta w_{ji} = -\varepsilon \frac{\partial E}{\partial w_{ji}}$$

Linear Activation Function

$$y_j^\alpha = \sum_i w_{ji} x_i^\alpha$$

=> Linear Regression Via Stochastic Gradient Descent

Computing $\partial E / \partial w_{ji}$

$$E = \sum_{\alpha=1}^{P} \sum_{k=1}^{B} (d_k^\alpha - y_k^\alpha)^2$$

$$E = \sum_\alpha \sum_k (d_k^\alpha - \sum_l w_{kl} x_l^\alpha)^2$$

$$\frac{\partial E}{\partial w_{ji}} = \sum_\alpha \sum_k 2 \cdot (d_k^\alpha - \sum_l w_{kl} x_l^\alpha) \frac{\partial}{\partial w_{ji}} \left[d_k^\alpha - \sum_l w_{kl} x_l^\alpha \right]$$

$= 0$ if $j \neq k$

$= -x_i^\alpha$ otherwise

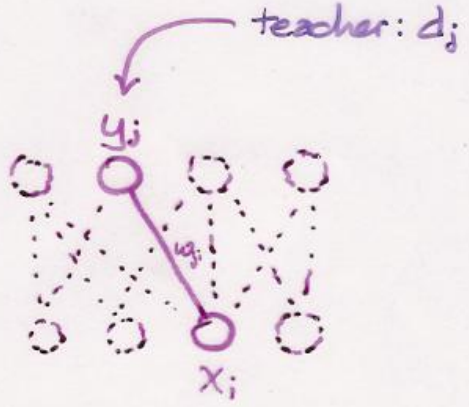
$$\frac{\partial E}{\partial w_{ji}} = \sum_\alpha 2 \cdot (d_j^\alpha - \Sigma w_{jl} x_l^\alpha) \cdot -x_i^\alpha$$

$$= -2 \sum_\alpha (d_j^\alpha - y_j^\alpha) x_i^\alpha$$

$\Rightarrow$

$$\Delta w_{ji} = -\epsilon \frac{\partial E}{\partial w_{ji}}$$

$$= \epsilon \sum_\alpha (d_j^\alpha - y_j^\alpha) x_i^\alpha$$



For each pattern, α,

$$\Delta w_{ji} = \epsilon (d_j^\alpha - y_j^\alpha) x_i^\alpha$$

ONLY LOCAL INFO IS REQUIRED!

Δb_j is like Δw_{ji}, assuming input is always 1:

$$\Delta b_j = \epsilon (d_j^\alpha - y_j^\alpha)$$

Shape of error surface

Suppose we have 1 input, 1 output unit (no bias)

$$E = \sum_{\alpha=1}^{p} \sum_{k=1}^{3} (d_k^\alpha - y_k^\alpha)^2$$

$$= \sum_{\alpha} (d^\alpha - wx^\alpha)^2$$

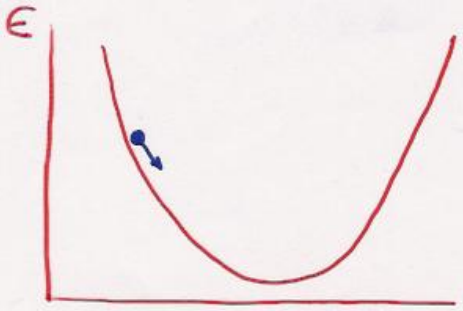
$$= \Sigma (d^{\alpha 2} - 2x^\alpha d^\alpha w + x^{\alpha 2} w^2)$$

$$= [\Sigma x^{\alpha 2}] w^2 - [\Sigma 2x^\alpha d^\alpha] w + [\Sigma d^{\alpha 2}]$$

∴ Error surface is quadratic in w (i.e., highest exponent of w is (2))

This is true no matter how many weights are in network

Only one minimum in quadratic surfaces ("hyperparabolic")



slope of error surface = dE/dw

Weight update procedure:

(1) start off at some random point in weight space (•)

(2) modify weights so as to move downhill in error

Batch Versus Online Training
(True Versus Stochastic Gradient Descent)

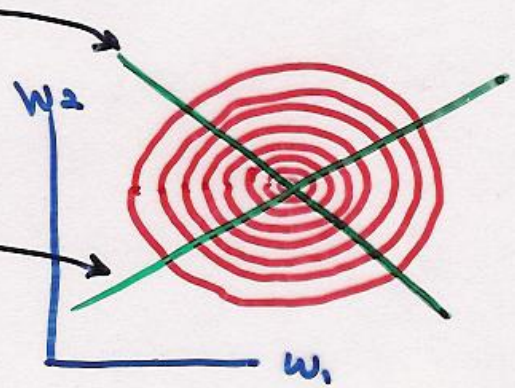
Suppose we have 2 inputs, one output, and two associations to learn, x^1-d^1, x^2-d^2.

Weights suitable for first association satisfy the constraint

$$x_1^1 w_1 + x_2^1 w_2 = d^1$$

And for the second association,

$$x_1^2 w_1 + x_2^2 w_2 = d^2$$



Batch Versus Online Training (True Versus Stochastic Gradient Descent)

Two ways of updating weights:

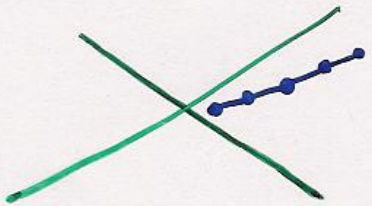
(1) sweep through all patterns and then modify weights

i.e., $\Delta w_{ji} = \sum_{\alpha=1}^{p} \epsilon (d_j^\alpha - y_j^\alpha) x_i^\alpha$

(2) modify weights after each pattern, α, has been presented

i.e., $\Delta w_{ji} = \epsilon (d_j^\alpha - y_j^\alpha) x_i^\alpha$

(1) "BATCH"



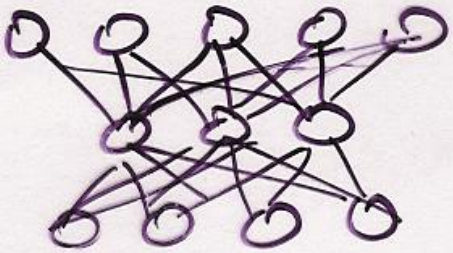
true gradient descent
i.e., $\Delta w \sim \partial E / \partial w$

(2) "ON-LINE"

noisy or "stochastic" descent

On-line can be faster if associations are similar!

Extending LMS To Handle Nonlinear Activation Functions And Multilayered Networks



First, let's rederive LMS rule

$$\Delta w_{ji} \sim -\frac{\partial E}{\partial w_{ji}}$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial w_{ji}}$$

$$\sim -(d_j - o_j)\, o_i$$

$$E = \sum_k (d_k - o_k)^2$$

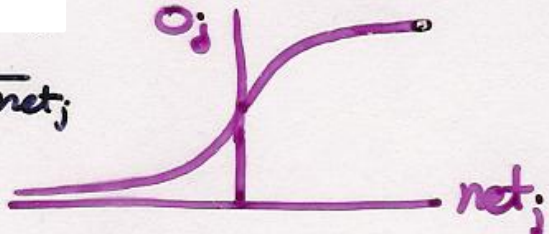
$$\frac{\partial E}{\partial o_j} = -2(d_j - o_j)$$

$$o_j = \sum_k w_{jk} o_k$$

$$\frac{\partial o_j}{\partial w_{ji}} = o_i$$

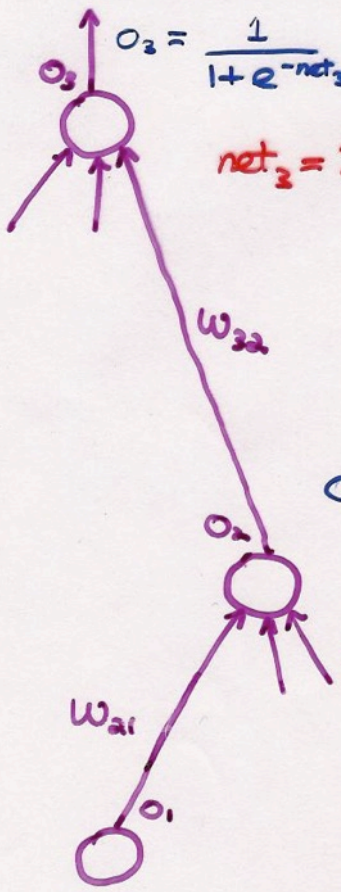
Logistic Activation Function

$$net_j = \sum_k w_{jk} o_k \qquad o_j = \frac{1}{1 + e^{-net_j}}$$



Back Propagation (Rumelhart, Hinton, & Williams; le Cun; Parker; Werbos)

$e = (d_3 - o_3)^2 \longrightarrow \frac{\partial e}{\partial o_3} = -2(d_3 - o_3)$



$$\frac{\partial e}{\partial w_{32}} = \frac{\partial e}{\partial o_3}\frac{\partial o_3}{\partial net_3}\frac{\partial net_3}{\partial w_{32}} \sim -(d_3 - o_3)\, o_3(1-o_3)\, o_2$$

$$\frac{\partial e}{\partial w_{21}} = \frac{\partial e}{\partial o_2}\frac{\partial o_2}{\partial net_2}\frac{\partial net_2}{\partial w_{21}} \sim (d_3 - o_3)\, o_3(1-o_3)\, w_{32}\, o_2(1-o_2)\, o_1$$

$$\frac{\partial e}{\partial o_2} = \frac{\partial e}{\partial o_3}\frac{\partial o_3}{\partial net_3}\frac{\partial net_3}{\partial o_2} = -(d_3 - o_3)\, o_3(1-o_3)\, w_{32}$$

What it boils down to

$$\Delta w_{ji} = \epsilon \delta_j o_i$$

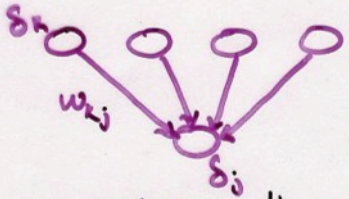
For output unit,

$$\delta_j = (d_j - o_j) \cdot o_j(1-o_j)$$

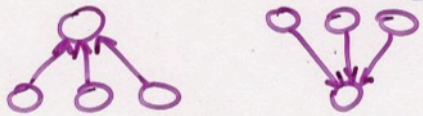
this term is simply $\frac{\partial o_j}{\partial net_j}$

For hidden unit,

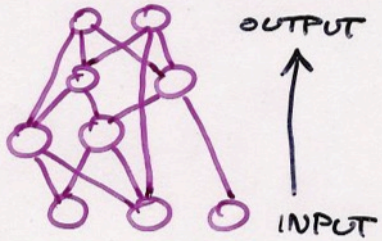
$$\delta_j = \left[\sum_k \delta_k w_{kj}\right] \cdot o_j(1-o_j)$$



- Δw_{ji} for output unit is the same as LMS with a nonlinear output (LMS ≡ "delta rule"; back prop ≡ "generalized delta rule")
- Two phase process: Forward (activation propagation) phase
  Backward (error propagation) phase

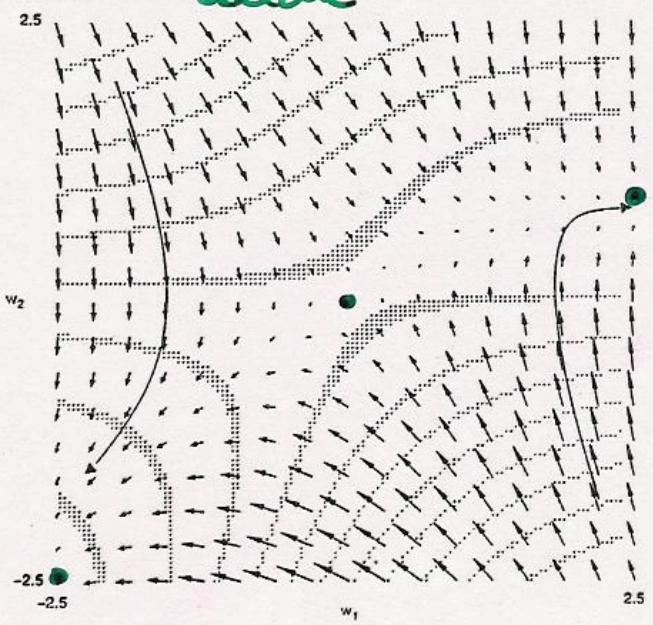


- As with LMS rule, back prop performs gradient descent in error space, i.e., finding best set of weights that minimize error. weights = all weights (biases) in network
- Back propagation works for arbitrarily deep feedforward networks

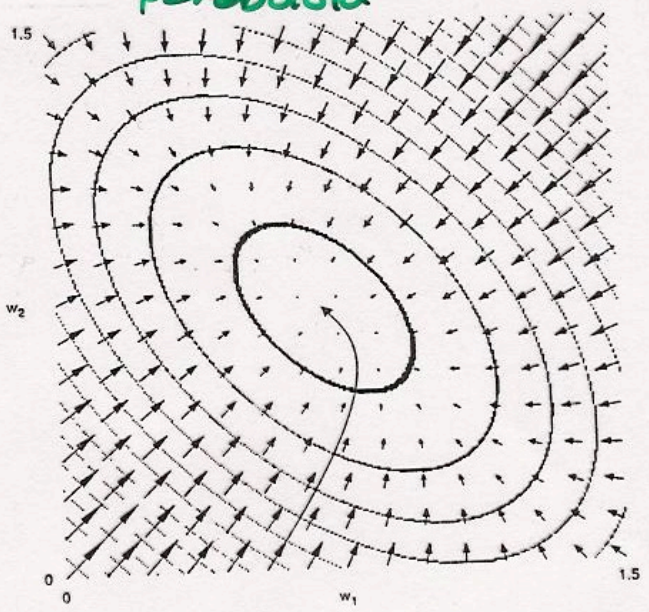


– Error surface is no longer hyperparabolic (i.e., no longer one global minimum)

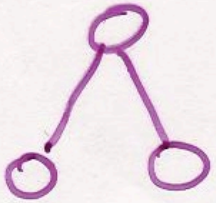
saddle



paraboloid



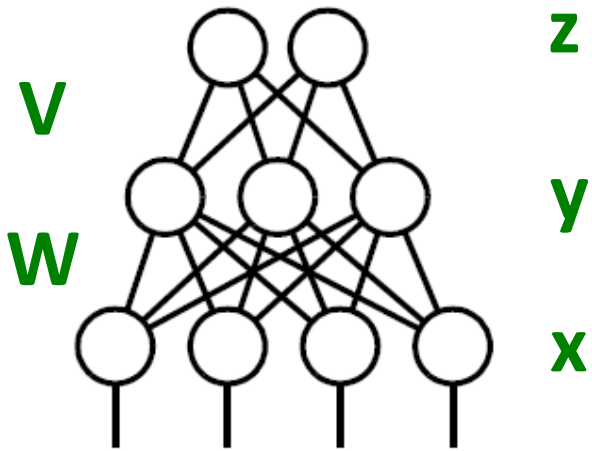
• = local minimum



Why Are Nonlinearities Necessary?

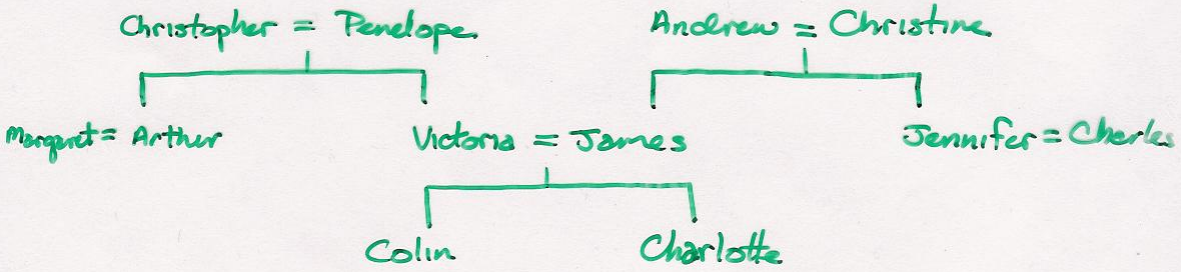
Prove

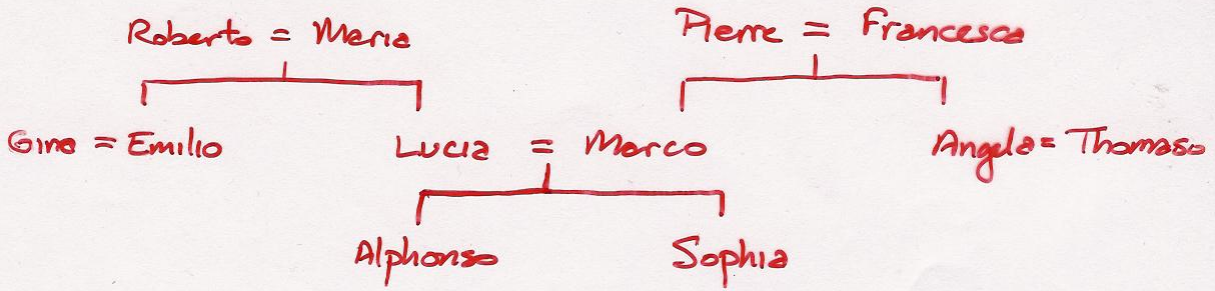
- **A network with a linear hidden layer has no more functionality than a network with no hidden layer (i.e., direct connections from input to output)**
- **For example, a network with a linear hidden layer cannot learn XOR**



Family Trees Task (HINTON, Proc. 8th Annual Conf. Cog. Sci Society, Amherst, MA, 1986, Erlbaum Assoc.)

Can we get a network to learn the information in these two family trees?





Knowledge can be represented as triples:

(has-mother Alphonso Lucia)
(has-wife James Victoria)
(has-son Penelope Arthur)

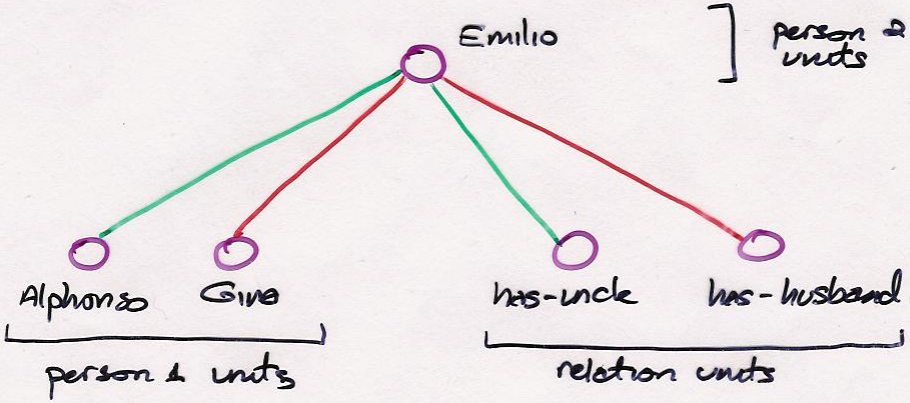
12 relationships: brother, sister, nephew, niece, uncle, aunt, husband, wife, son, daughter, father, mother

Can system learn to produce the third term of a triple when given the first two?

E.g., (has-aunt Alphonso ?) Angela!

A two layer net won't do:

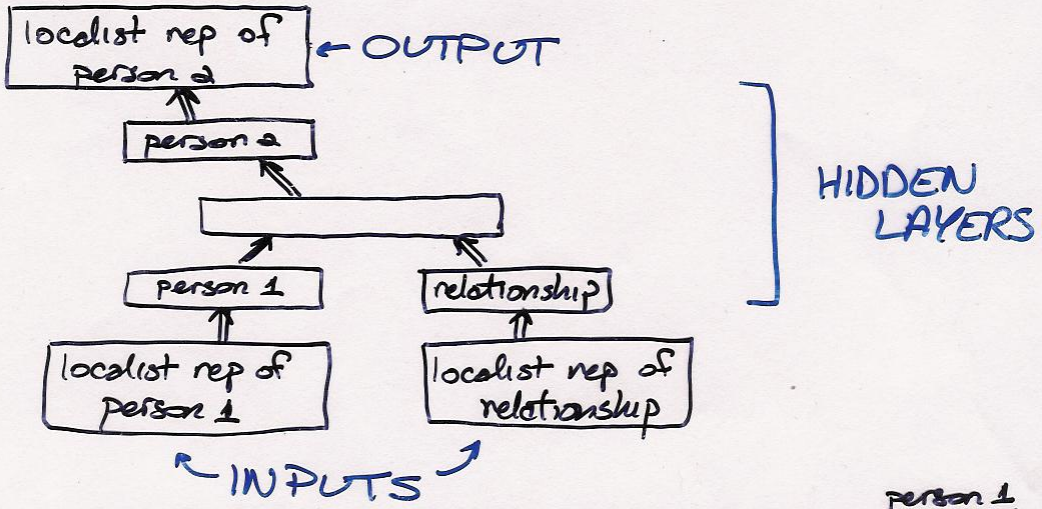
E.g., (has-uncle Alphonso Emilio)
(has-husband Gina Emilio)



Problem:

(has-uncle Gina ?)
(has-husband Alphonso ?)

Hinton's architecture



System must learn an "internal representation" of person 1 person 2 relationship

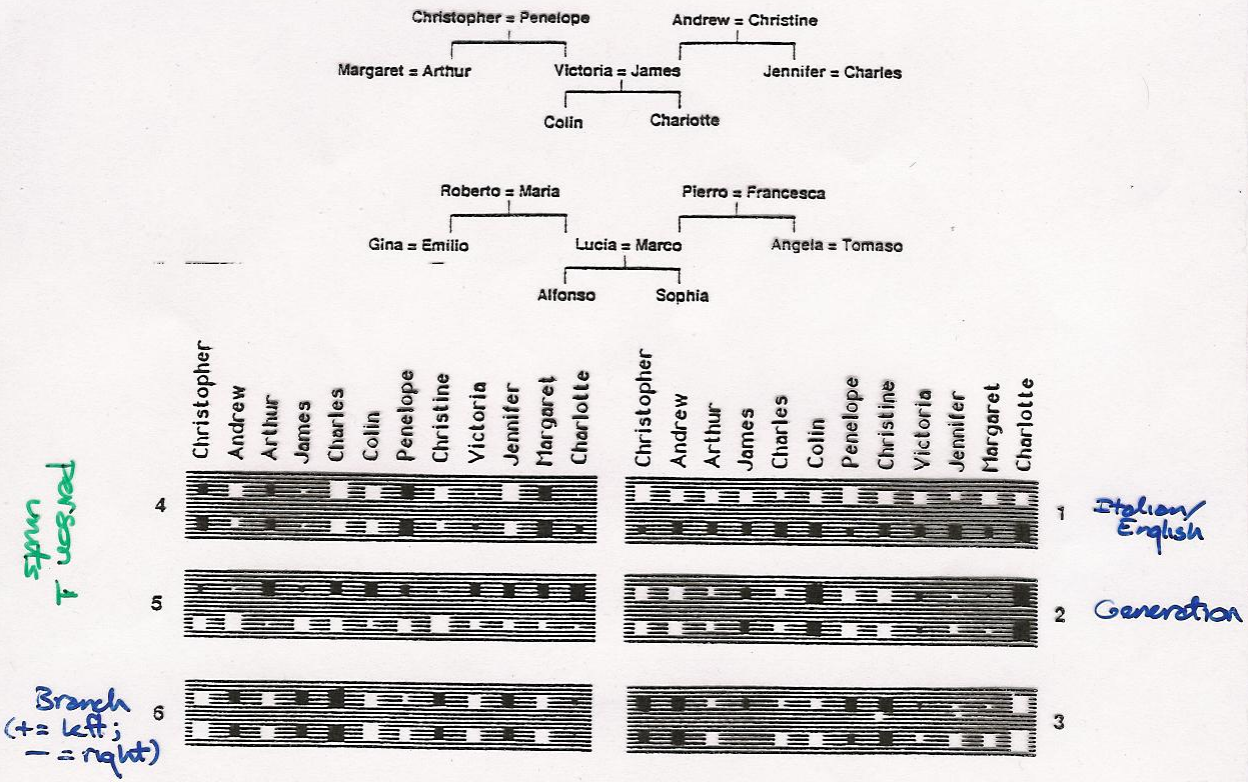


Figure 6: The weights from the 24 input units that represent people to the 6 units in the second layer that learn distributed representations of people. White rectangles stand for excitatory weights, black for inhibitory weights, and the area of the rectangle encodes the magnitude of the weight. The weights from the 12 English people are in the top row of each unit. Beneath each of these weights is the weight from the isomorphic Italian.

Figure 7: The weights from the 12 input units that represent relationships to the 6 units in the second layer that learn distributed representations of the relationships.

Changing Loss Function

squared error

$$E = \frac{1}{2}\sum_j \left(d_j - y_j\right)^2$$

$$\frac{\partial E}{\partial y_j} = d_j - y_j$$

cross entropy (= max likelihood)

$$E = -\sum_j d_j \log y_j + (1 - d_j)\log(1 - y_j)$$

$$\frac{\partial E}{\partial y_j} = \frac{d_j - y_j}{y_j(1 - y_j)}$$

Changing Loss Function

Back propagation

- **logistic activation function**

$$z_j = \sum_i w_{ji} x_i \qquad y_j = \frac{1}{1+\exp(-z_j)}$$

- **weight update**

$$\Delta w_{ji} = \varepsilon \delta_j x_i$$

$$\delta_j = \begin{cases} \dfrac{\partial E}{\partial y_j} y_j (1 - y_j) & \text{for output unit} \\ \left(\sum_k w_{kj} \delta_k \right) y_j (1 - y_j) & \text{for hidden unit} \end{cases}$$

Changing Activation Function 1

Back propagation

- **softmax activation function for 1-of-N classification**

$$z_j = \sum_i w_{ji} x_i \qquad y_j = \frac{\exp(z_j)}{\sum_k \exp(z_k)}$$

- **weight update**

$$\Delta w_{ji} = \varepsilon \delta_j x_i \qquad \delta_j = \begin{cases} \dfrac{\partial E}{\partial y_j} y_j (1 - y_j) & \text{for output unit} \\ \left(\sum_k w_{kj} \delta_k \right) y_j (1 - y_j) & \text{for hidden unit} \end{cases}$$

- **gradient is the same when expressed in terms of y_j**

Changing Activation Function 2

Back propagation

- **tanh activation function**

$$z_j = \sum_i w_{ji} x_i \qquad y_j = \tanh(z_j) = 2\,\text{logistic}(z_j)$$

- **weight update**

$$\Delta w_{ji} = \varepsilon \delta_j x_i \qquad \delta_j = \begin{cases} \dfrac{\partial E}{\partial y_j}(1+y_j)(1-y_j) & \text{for output unit} \\ \left(\sum_k w_{kj}\delta_k\right)(1+y_j)(1-y_j) & \text{for hidden unit} \end{cases}$$

- **incompatible with cross entropy loss**