

# ML Week

## Artificial Neural Networks

Jeff Abrahamson

23–24 novembre 2016

# Introducing Perceptron

- Supervised learning
- Binary classifier
- Linear classifier
- Permits online learning

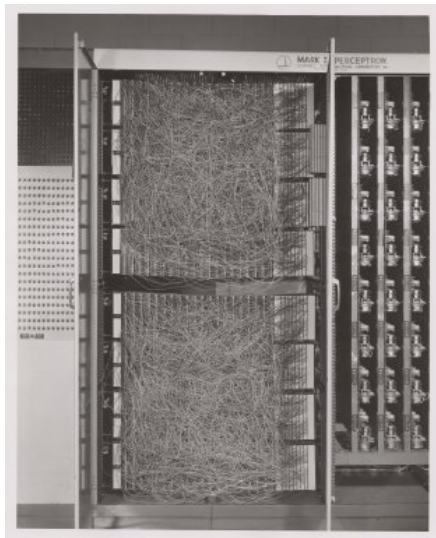
# History

- Invented in 1957 by Frank Rosenblatt
- Cornell University Aeronautical Laboratory
- Paid for by ONR
- First implementation in software (IBM 704)
- Intended to be a machine
- Designed for image recognition

# Hardware

- Inputs = 400 CdS photocells
- Weights = potentiometers
- Tuning = electric motors

# Patch panel and potentiometers



# Controversy

- 1958, press conference, NYT
- Rosenblatt too optimistic
- 1969, Minsky and Papert

We want to learn a linear separator

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

where  $w \in \mathbb{R}^{n+1}$ .

# Algorithm

- 1 Initialise weights
- 2 From input, compute output
- 3 If correct, add  $(\alpha \cdot \text{input})$  to weight



# Convergence

- If linearly separable, yes
- If not linearly separable, learning fails

# feedforward neural network

# Deep Learning?

- Machine learning
- Model high-level abstraction by using multiple non-linear transformations.

# Deep Learning?

- Machine learning
- Model high-level abstraction by using multiple non-linear transformations.
- Example: **Image:**

pixels  $\Rightarrow$  edges  $\Rightarrow$  shapes  $\Rightarrow$  faces.

# Review

Broadly, ML comes in three flavors:

- **Supervised learning:** Predict output given input
- **Reinforcement learning:** Select action to maximize payoff
- **Unsupervised learning:** Discover a good internal representation of input

# Review

Supervised learning comes in two flavors:

- **Regression:** real-valued output
- **Classification:** labeled output

# Review

The idea behind supervised learning is often written thus:

$$y = f(x, W)$$

# Review

The idea behind supervised learning is often written thus:

$$y = f(x, W)$$

where

$y$  = predicted output

$x$  = input

$W$  = parameters

and our goal is to adjust parameters to minimize loss (error)



**Labels are expensive**

# Review

## Unsupervised learning

- Historically, it's clustering
- Now we can do more

## Unsupervised learning

- Historically, it's clustering
- Now we can do more
- Create an internal representation of the input that is useful for later supervised or reinforcement learning
- Find a compact, low-dimensional representation of the input

# Some architectures

- Deep neural networks
- Convolutional deep neural networks
- Deep belief networks

# Some successful applications

- Computer vision (CV)
- Speech recognition (ASR)
- Natural language processing (NLP)
- Music and audio recognition

# Some famous data sets

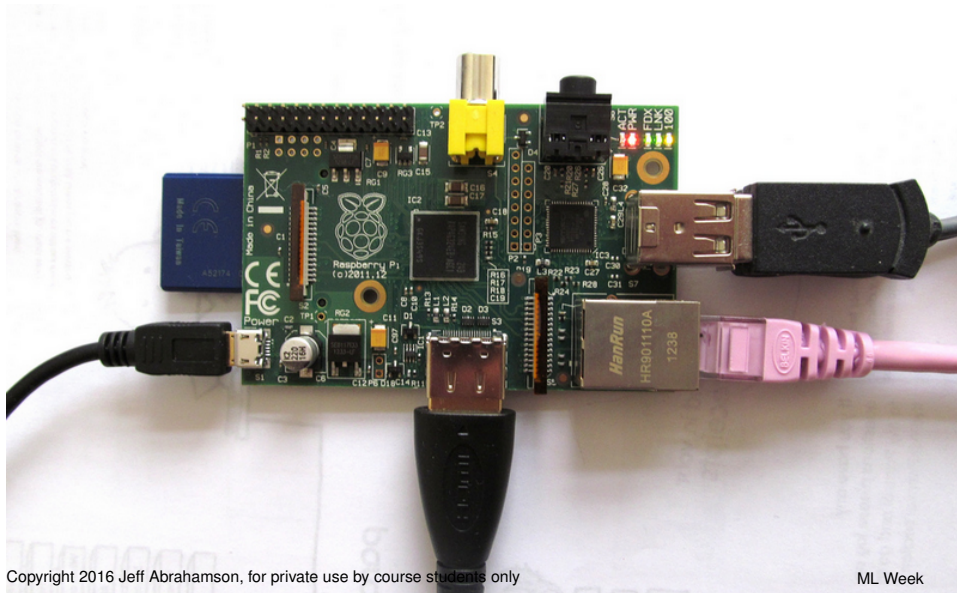
- TIMIT (ASR)
- MNIST (image classification)
- ImageNet

# Some successful hardware

- GPU's
- Data centers

*Luiz André Barroso and Urs Hölzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 2009.*

# GPU's



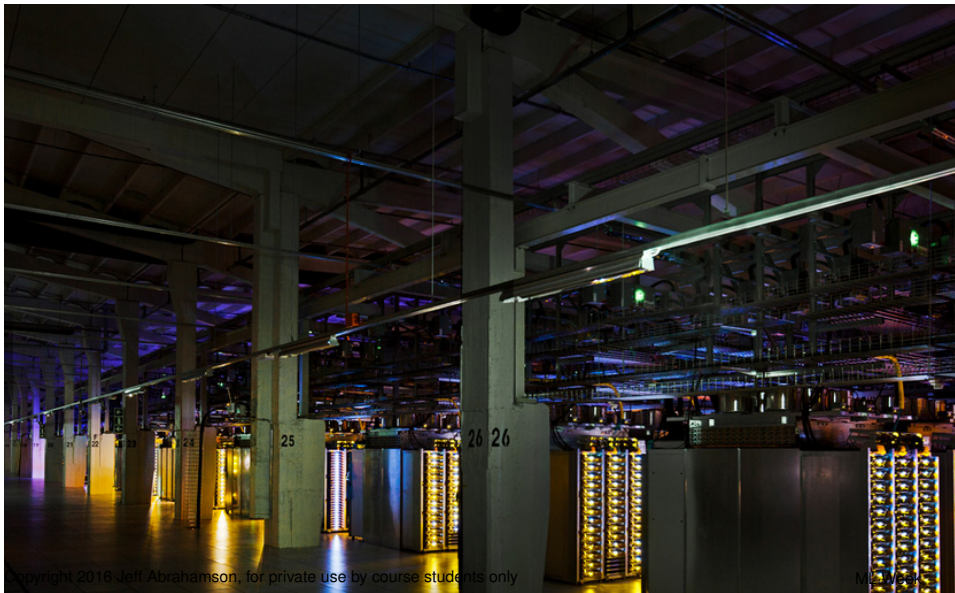


# What is Hello World for GPU's?

Some things to look at. (Disclaimer: I haven't.)

- CUDA (but only nvidia)  
`www.nvidia.com/object/cuda_home_new.html`
- OpenCL (originally Apple) `https://en.wikipedia.org/wiki/OpenCL`
- GPU++ / GPGPU `http://gpgpu.org/`
- libSh `http://libsh.org/`
- OpenACC `http://www.openacc-standard.org/`

# Data Centers



# Why ML at all?

- We don't know how *we* do it
- Write programs to write programs

# Brains (yum!)

- Neurons, synapses, chemistry
- Special kind of parallelism
- Power (but we're not there yet)

## A brief tour of some neurons

**video time**

# Linear neuron

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

# Linear neuron

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

where

$y$  = output

$b$  = bias

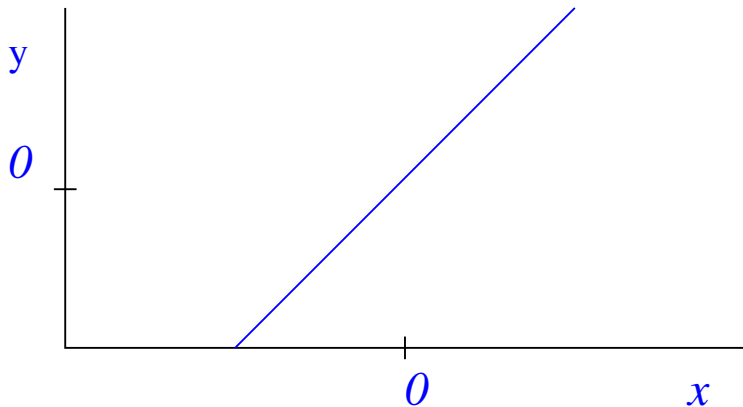
$x_i$  =  $i^{\text{th}}$  input

$w_i$  = weight on  $i^{\text{th}}$  input



# Linear neuron

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



# Binary threshold neuron

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

# Binary threshold neuron

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where

$z$  = total input

$y$  = output

$x_i$  =  $i^{\text{th}}$  input

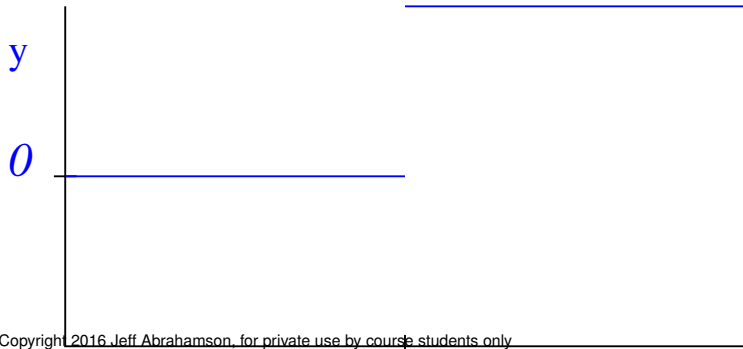
$w_i$  = weight on  $i^{\text{th}}$  input

*W. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115–133, 1943.*

# Binary threshold neuron

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



# Rectified linear neuron

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

$$y = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

# Rectified linear neuron

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

$$y = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where

$z$  = total input

$y$  = output

$b$  = bias

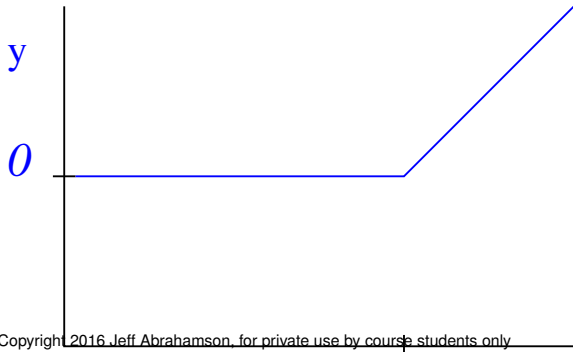
$x_i$  =  $i^{\text{th}}$  input

$w_i$  = weight on  $i^{\text{th}}$  input

# Rectified linear neuron

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

$$y = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



# Sigmoid neuron

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

$$y = \frac{1}{1 + e^{-z}}$$



# Sigmoid neuron

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

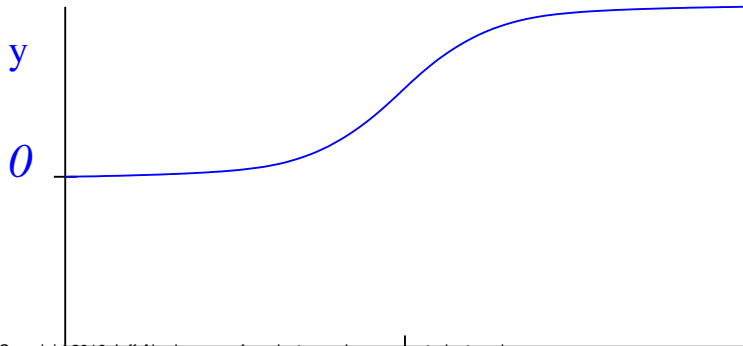
$$y = \frac{1}{1 + e^{-z}}$$

(It's differentiable!)

# Sigmoid neuron

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

$$y = \frac{1}{1 + e^{-z}}$$



# Stochastic binary neuron

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

$$p = \frac{1}{1 + e^{-z}}$$

$$y = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$$

# Stochastic binary neuron

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

$$p = \frac{1}{1 + e^{-z}}$$

$$y = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$$

(a probability distribution)

# Stochastic binary neuron

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

$$p = \frac{1}{1 + e^{-z}}$$

$$y = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$$

Can also do something similar with rectified linear neurons, produce spikes with probability  $p$  with a Poisson distribution.

A (too) quick example

## Example: handwriting recognition of digits

- Input neurons: pixels
- Output neurons: classes (digits)
- Connect them all! (*bipartite*)

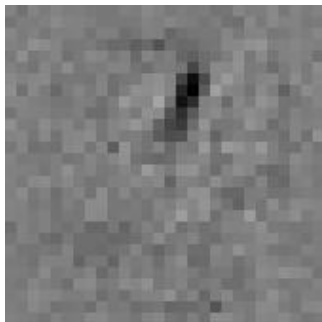
## Example: handwriting recognition of digits

- Input neurons: pixels
- Output neurons: classes (digits)
- Connect them all! (*bipartite*)
- Initialize input weights to random



# Example: handwriting recognition of digits

- Input neurons: pixels
- Output neurons: classes (digits)
- Connect them all! (*bipartite*)
- Initialize input weights to random



# Example: handwriting recognition of digits

To train this ANN:

- Increment weights from active pixels going to correct class
- Decrement weights from active pixels going to predicted class

# Example: handwriting recognition of digits

To train this ANN:

- Increment weights from active pixels going to correct class
- Decrement weights from active pixels going to predicted class

When it's right, nothing happens. This is good.

## Some notes on architectures

# Architectures

It's about how we connect the neurons.

# Feedforward neural networks

- Input comes into input neurons
- Flow is unidirectional
- No loops
- Output at output neurons

# Recurrent neural networks

- Cycles
- Memory
- Oscillations
- More powerful
- Hard to train (research interest)
- More biologically realistic

# Recurrent neural networks

- Cycles
- Memory
- Oscillations
- More powerful
- Hard to train (research interest)
- More biologically realistic

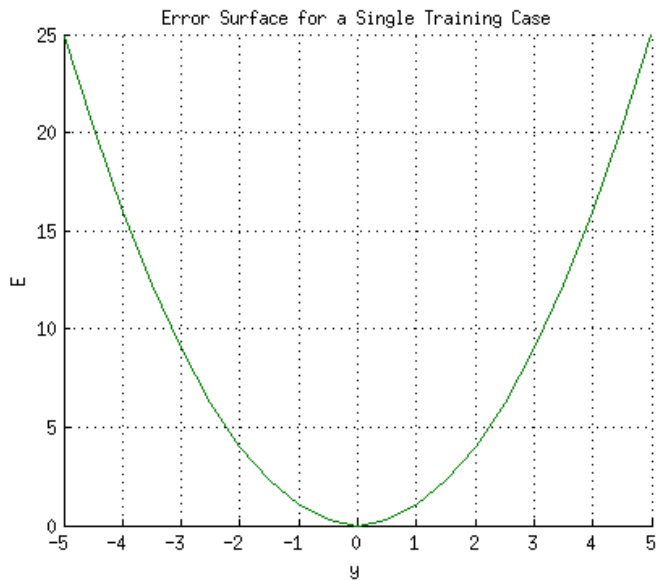
Deep RNN is just a special case of a general recurrent NN with some hidden links missing.



# Backwards propagation of errors

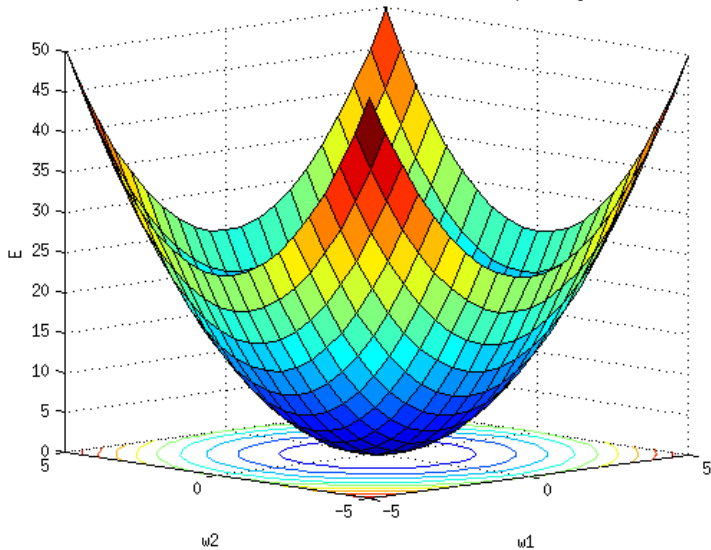
# The problem

- Multi-layer ANN's are powerful
- Training them is (was) hellish



source: wikipedia

Error Surface of a Linear Neuron with Two Input Weights



source: wikipedia

# Backpropagation

- Backward propagation of errors
- To calculate loss function, need a known, desired output for each input
- Gradient descent
- Calculate gradient of loss function w.r.t. all weights and minimize loss function

# Layers

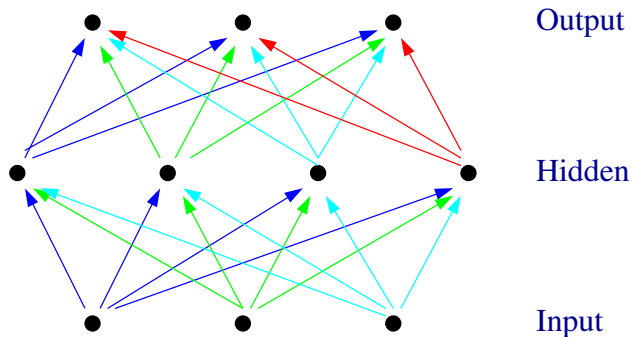
- Each layer computes a representation of its input
- Can change similarity

# Layers

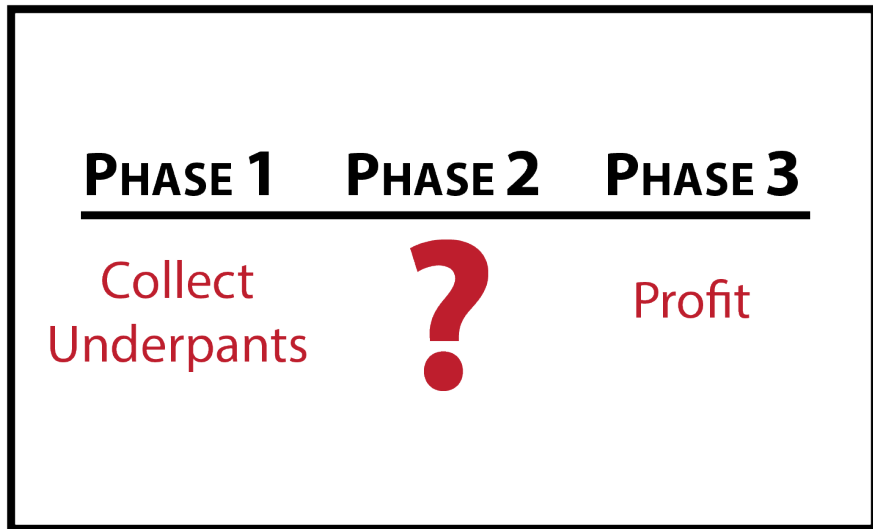
- Each layer computes a representation of its input
- Can change similarity
- Example:
  - (different speakers, same word) should become more similar
  - (same speaker, different words) should become more dissimilar

# Layers

- If more than two hidden layers, then we call it deep
- Neuron activity at each layer must be a non-linear function of previous layer







# Some inspirations

- Biology: David H. Hubel and Torsten Wiesel (1959) found two types of cells in the visual primary cortex: simple and complex.
- Cascading models

*M Riesenhuber, T Poggio. Hierarchical models of object recognition in cortex. Nature neuroscience, 1999(11) 1019–1025.*

# History

- ANN's exist pre-1980. Backpropagation since 1974.

*P. Werbos., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," PhD thesis, Harvard University, 1974.*

# History

- ANN's exist pre-1980. Backpropagation since 1974.
- Neocognitron (Kunihiko Fukushima, 1980), partially unsupervised

*K. Fukushima., "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biol. Cybern., 36, 193–202, 1980.*

# History

- ANN's exist pre-1980. Backpropagation since 1974.
- Neocognitron (Kunihiko Fukushima, 1980), partially unsupervised
- Yann LeCun et al. recognize handwritten postal codes (backpropagation)

*LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, 1, pp. 541–551, 1989.*

Aside: statistical pattern recognition looks like this:

- 1 Convert raw input vector into a vector of feature activations (hand-written)
- 2 Learn weights on feature activation to get single scalar quantity
- 3 If scalar quantity exceeds some threshold, then decide that input vector is an example of the target

# Perceptron

- Perceptron is an example of SPR for image recognition
- Initially very promising

*Frank Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386—408, 1958.  
doi:10.1037/h0042519*

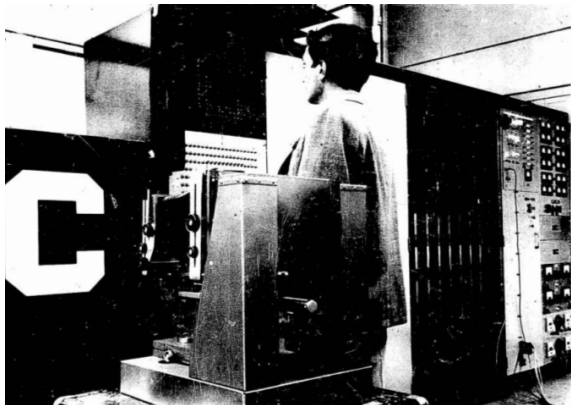
# Perceptron

- Perceptron is an example of SPR for image recognition
- Initially very promising
- IBM 704 (software implementation of algorithm)
- Mark 1 Perceptron at the Smithsonian Institution
- 400 photocells randomly connected to neurons.
- Weights encoded in potentiometers, updated during learning by electric motors

*Frank Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386—408, 1958.  
doi:10.1037/h0042519*



# Mark 1 Perceptron



*Frank Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Report No. 1196-G-8, 15 March 1961, Cornell Aeronautical Laboratory*

# Perceptron

- Minsky and Papert showed perceptrons are incapable of recognizing certain classes of images
- AI community mistakenly over-generalized to all NN's
- So NN research stagnated for some time

*M. L. Minsky and S. A. Papert, Perceptrons. Cambridge, MA: MIT Press. 1969.*

# Perceptron

- Minsky and Papert showed perceptrons are incapable of recognizing certain classes of images
- AI community mistakenly over-generalized to all NN's
- So NN research stagnated for some time
- Single layer perceptrons only recognize linearly separable input
- Hidden layers overcome this problem

*M. L. Minsky and S. A. Papert, Perceptrons. Cambridge, MA: MIT Press. 1969.*

# Paradise glimpsed, paradise lost

- ANN's were slow.
- Vanishing gradient problem (Sepp Hochreiter)
- Support vector machines (SVN) were faster

*S. Hochreiter., "Untersuchungen zu dynamischen neuronalen Netzen," Diploma thesis. Institut f. Informatik, Technische Univ. Munich. Advisor: J. Schmidhuber, 1991.*

*S. Hochreiter et al., "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.*

# Some progress

Multi-level hierarchy of networks (pre-train by level, unsupervised, backpropagation) (1992)

*J. Schmidhuber., "Learning complex, extended sequences using the principle of history compression," Neural Computation, 4, pp. 234–242, 1992.*

# Some progress

## Long short term memory network (LSTM) (1997)

*Hochreiter, Sepp; and Schmidhuber, Jürgen; Long Short-Term Memory, Neural Computation, 9(8):1735–1780, 1997.*

# Some progress

Deep multidimensional LSTM networks win three ICDAR competitions in handwriting recognition without prior language knowledge (2009)

*Graves, Alex; and Schmidhuber, Jürgen; Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), Advances in Neural Information Processing Systems 22 (NIPS'22), December 7th–10th, 2009, Vancouver, BC, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552.*

*A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.*

# Some progress

Use sign of gradient (Rprop) for image reconstruction and face localization (2003)

*Sven Behnke (2003). Hierarchical Neural Networks for Image Interpretation.. Lecture Notes in Computer Science 2766. Springer.*



# And then there was Hinton

- Geoffrey Hinton and Ruslan Salakhutdinov
- Train many-layered feedforward NN's one layer at a time
- Treat layers as unsupervised restricted Boltzmann machines (Smolensky, 1986)
- Use supervised backpropagation for label classification
- Also: Schmidhuber and recurrent NN's

# And then there was Hinton (bibliography)

*G. E. Hinton., "Learning multiple layers of representation," Trends in Cognitive Sciences, 11, pp. 428–434, 2007.*

*J. Schmidhuber., "Learning complex, extended sequences using the principle of history compression," Neural Computation, 4, pp. 234–242, 1992.*

*J. Schmidhuber., "My First Deep Learning System of 1991 + Deep Learning Timeline 1962–2013."*

*Smolensky, P. (1986). "Information processing in dynamical systems: Foundations of harmony theory." In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. 1. pp. 194–281.*

## And then there was Hinton (bibliography)

*Hinton, G. E.; Osindero, S.; Teh, Y. (2006). "A fast learning algorithm for deep belief nets". Neural Computation 18 (7): 1527–1554. doi:10.1162/neco.2006.18.7.1527. PMID 16764513.*

*Hinton, G. (2009). "Deep belief networks". Scholarpedia 4 (5): 5947. doi:10.4249/scholarpedia.5947. edit*

## Yet more progress

Google Brain project (Andrew Ng, Jeff Dean) recognized cats in youtube videos.

*Ng, Andrew; Dean, Jeff (2012). "Building High-level Features Using Large Scale Unsupervised Learning".*

*John Markoff (25 June 2012). "How Many Computers to Identify a Cat? 16,000.", New York Times.*

# More progress

Brute force!

Dan Ciresan et al. (IDSIA, 2010) use lots of GPU's to bulldoze the vanishing gradient problem and outperform LeCun (and everyone else) on MNIST.

*D. C. Ciresan et al., "Deep Big Simple Neural Nets for Handwritten Digit Recognition," Neural Computation, 22, pp. 3207–3220, 2010.*

# State of the art, 2011

## Deep learning feedforward networks

- Convolutional layers
- Max-pooling layers
- Plus pure classification layers

*D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber. Flexible, High Performance Convolutional Neural Networks for Image Classification. International Joint Conference on Artificial Intelligence (IJCAI-2011, Barcelona), 2011.*

*Martines, H., Bengio, Y., and Yannakakis, G. N. (2013). Learning Deep Physiological Models of Affect. I IEEE Computational Intelligence, 8(2), 20.*

# State of the art, post-2011

Lots of GPU's. Sometimes human-competitive performance!

- IJCNN 2011 Traffic Sign Recognition Competition
- ISBI 2012 Segmentation of neuronal structures in EM stacks challenge
- and more

# State of the art, post-2011

*D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber. Flexible, High Performance Convolutional Neural Networks for Image Classification. International Joint Conference on Artificial Intelligence (IJCAI-2011, Barcelona), 2011*

*D. C. Ciresan, U. Meier, J. Masci, J. Schmidhuber. Multi-Column Deep Neural Network for Traffic Sign Classification. Neural Networks, 2012.*

*D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In Advances in Neural Information Processing Systems (NIPS 2012), Lake Tahoe, 2012.*

*D. C. Ciresan, U. Meier, J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. IEEE Conf. on Computer Vision and Pattern Recognition CVPR 2012.*



# Basic ideas

- Distributed representations: observed data is organized at multiple levels of abstraction or composition
- Higher level concepts learned from lower level concepts (hierarchical explanatory factors)
- Often can frame problems as unsupervised. (Labeling is expensive.)

*Y. Bengio, A. Courville, and P. Vincent., "Representation Learning: A Review and New Perspectives," IEEE Trans. PAMI, special issue Learning Deep Architectures, 2013.*

# Basic ideas

- Unsupervised  $\Rightarrow$  unlabeled data is ok
- Often greedy between layers

# Basic ideas

- Science advances in fits and starts
- Sometimes dead-ends just take time
- We still can't recognize cats at 100W powered by bananas

# Questions?

[ml-week.com/1](http://ml-week.com/1)