Capstone Project for Data Science

Yuan YAO

HKUST

Course Infomation

- Course web:
  - https://yao-lab.github.io/capstone/2020.fall/
- Time:
  - TuTh, 4:30-5:50pm
- Venue:
  - Zoom Meetings, join from CANVAS
- Instructor:
  - Yuan Yao < yuany@ust.hk > (https://yao-lab.github.io/)
- Teaching Assistant:
  - Weizhi Zhu <wzhuai@connect.ust.hk>

Course Requirement

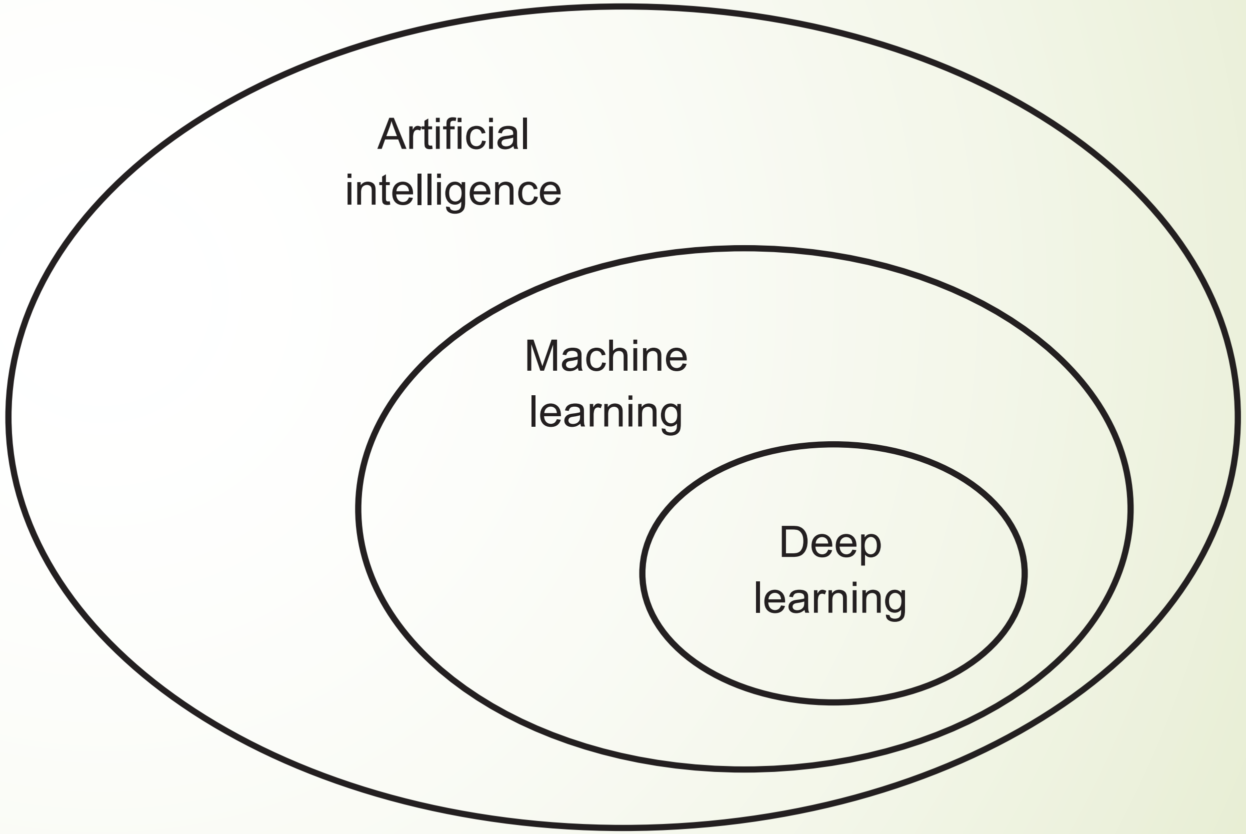
- **Prerequisite**: (Statistical) Machine Learning, e.g. working knowledge about linear regression, classification, logistic regression, decision trees (CART), boosting, random forests, support vector machines, neural networks
- With these, you can pursue basic projects in this class
- Advanced project needs convolutional Neural Networks

- **Projects**:
  - Basic level:
    - Kaggle Contest: Home Credit Default Risk
    - https://www.kaggle.com/c/home-credit-default-risk/overview
  - Advanced level:
    - Kaggle contest: Nexperia Image Classification
    - https://www.kaggle.com/c/semi-conductor-image-classification-first

A Brief History of AI, Machine Learning, and Deep Learning

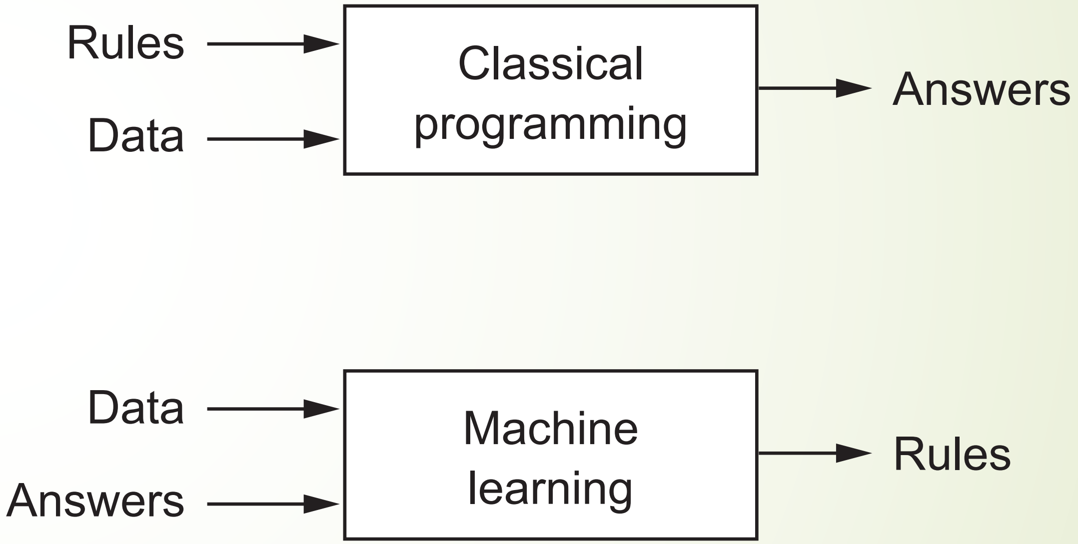
Artificial Intelligence, Machine Learning, and Deep Learning

- AI is born in 1950s, when a handful of pioneers from the nascent field of computer science started asking **whether computers could be made to "think"**—a question whose ramifications we're still exploring today.



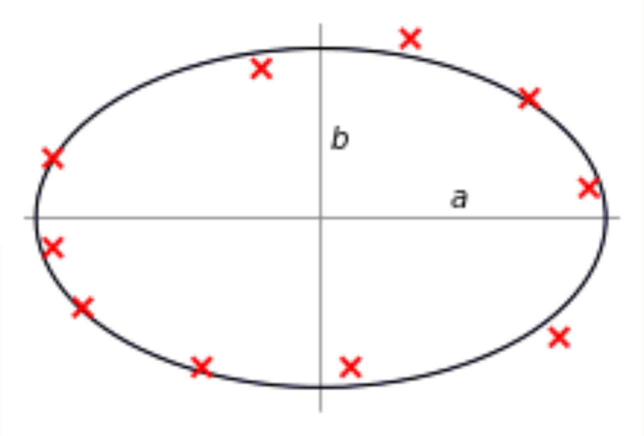
Machine Learning is a new paradigm of computer programming

- During 1950s-1980s, two competitive ideas of realizing AI exist
  - Rule based inference, or called **Expert System**
  - Statistics based inference, or called **Machine Learning**
- 1990s- Machine Learning becomes dominant



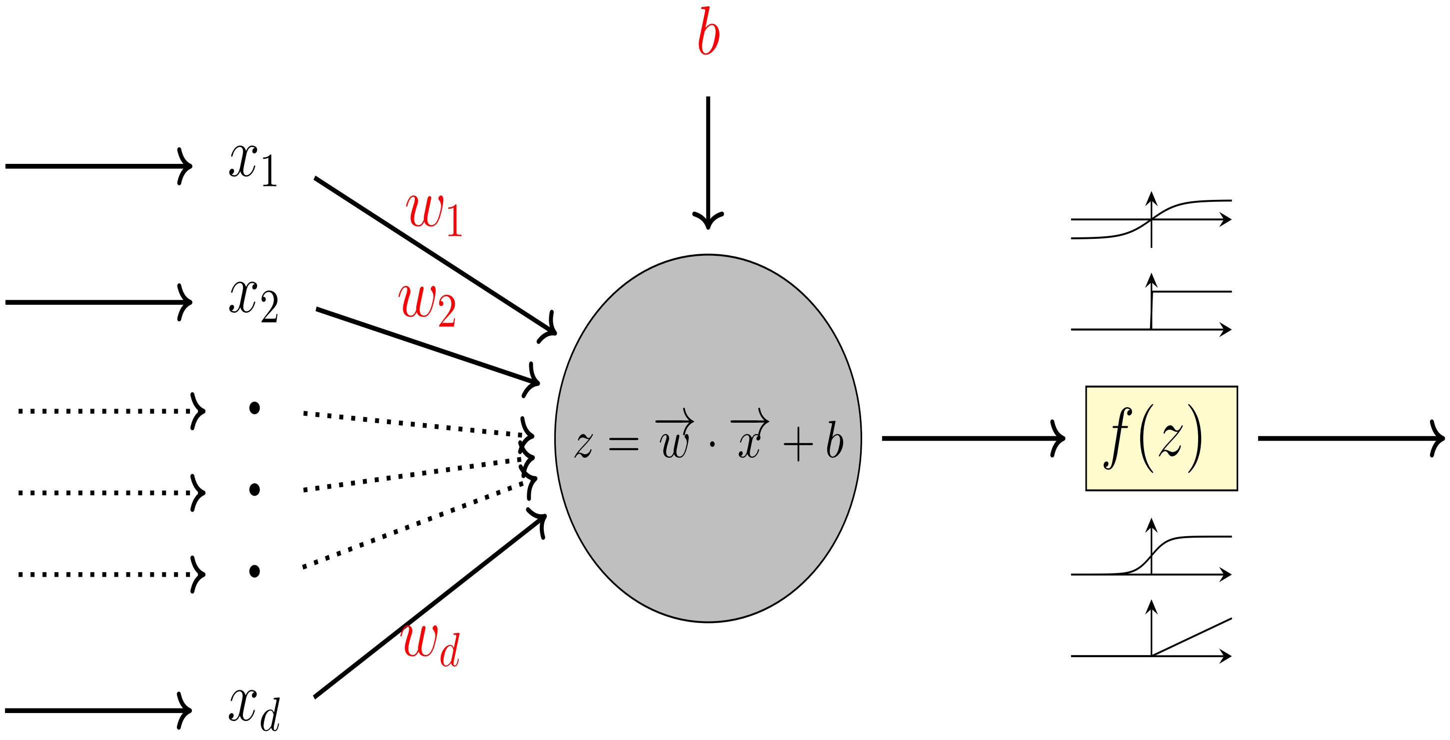
The 1st machine learning method: Least Squares

- Invention:
  - **Carl Friederich Gauss** (~1795/1809/1810),
  - Adrien-Marie Legendre (1805)
  - Robert Adrain (1808)
- Application:
  - Prediction of the location of asteroid Ceres after it emerged from behind the sun (Franz Xaver von Zach 1801)
  - Orbits of planets, Newton Laws
  - Statistics,
  - ...



The 1st neural network: Perceptron

- Invented by Frank Rosenblatt (1957)



The Perceptron Algorithm for classification

$$\ell(w) = -\sum_{i \in \mathcal{M}_w} y_i \langle w, \mathbf{x}_i \rangle, \quad \mathcal{M}_w = \{i : y_i \langle \mathbf{x}_i, w \rangle < 0, y_i \in \{-1, 1\}\}.$$

The Perceptron Algorithm is a *Stochastic Gradient Descent* method (**Robbins-Monro 1951**):

$$\begin{aligned} w_{t+1} &= w_t - \eta_t \nabla_i \ell(w) \\ &= \begin{cases} w_t - \eta_t y_i \mathbf{x}_i, & \text{if } y_i w_t^T \mathbf{x}_i < 0, \\ w_t, & \text{otherwise.} \end{cases} \end{aligned}$$

Finiteness of Stopping Time and Margin

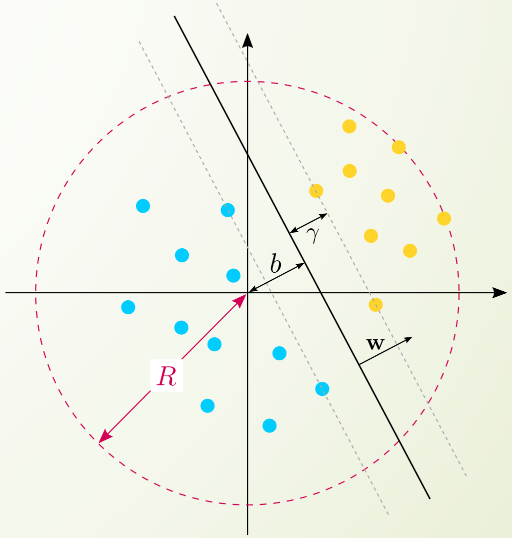
The perceptron convergence theorem was proved by Block (1962) and Novikoff (1962). The following version is based on that in Cristianini and Shawe-Taylor (2000).

Theorem 1 (Block, Novikoff). *Let the training set $S = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$ be contained in a sphere of radius R about the origin. Assume the dataset to be linearly separable, and let* $\mathbf{w}_{\text{opt}}$, $\|\mathbf{w}_{\text{opt}}\| = 1$, *define the hyperplane separating the samples, having functional margin* $\gamma > 0$. *We initialise the normal vector as* $\mathbf{w}_0 = \mathbf{0}$. *The number of updates, k, of the perceptron algorithms is then bounded by*

$$k \leq \left(\frac{2R}{\gamma}\right)^2. \quad (10)$$

Input ball: $R = \max_i \|\mathbf{x}_i\|.$

Margin: $\gamma := \min_i y_i f(x_i)$



Hilbert's 13th Problem

Algebraic equations (under a suitable transformation) of degree up to 6 can be solved by functions of two variables. What about

$$x^7 + ax^3 + bx^2 + cx + 1 = 0?$$

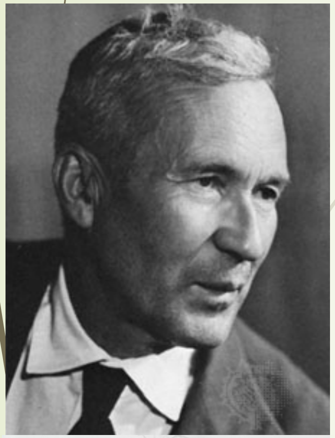
Hilbert's conjecture: $x(a, b, c)$ cannot be expressed by a superposition (sums and compositions) of bivariate functions.

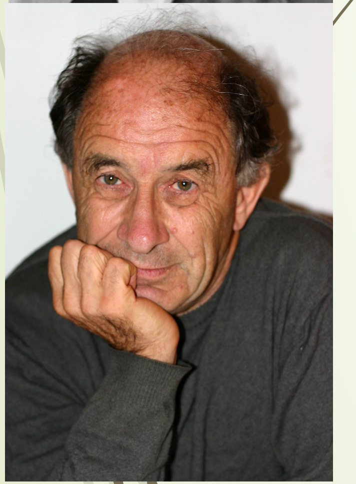
Question: can every continuous (analytic, C^∞, etc) function of n variables be represented as a superposition of continuous (analytic, C^∞, etc) functions of $n-1$ variables?

Theorem (D. Hilbert)

There is an analytic function of three variables that cannot be expressed as a superposition of bivariate ones.

Kolmogorov's Superposition Theorem





Theorem (A. Kolmogorov, 1956; V. Arnold, 1957)

Given $n \in \mathbb{Z}^+$, every $f_0 \in C([0,1]^n)$ can be reprensented as

$$f_0(x_1, x_2, \cdots, x_n) = \sum_{q=1}^{2n+1} g_q \left(\sum_{p=1}^{n} \phi_{pq}(x_p) \right),$$

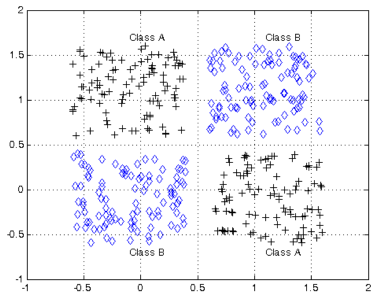
where $\phi_{pq} \in C[0,1]$ are increasing functions independent of f_0 and $g_q \in C[0,1]$ depend on f_0.

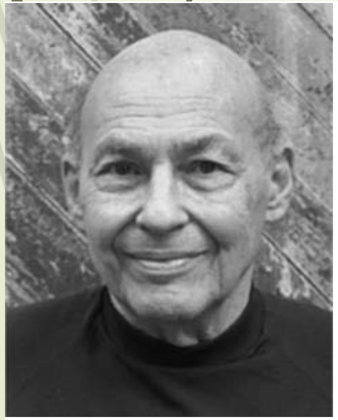
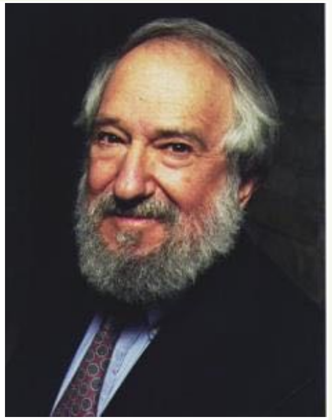
- Can choose g_q to be all the same $g_q \equiv g$ (Lorentz, 1966).
- Can choose ϕ_{pq} to be Hölder or Lipschitz continuous, but not C^1 (Fridman, 1967).
- Can choose $\phi_{pq} = \lambda_p \phi_q$ where $\lambda_1, \cdots, \lambda_n > 0$ and $\sum_p \lambda_p = 1$ (Sprecher, 1972).

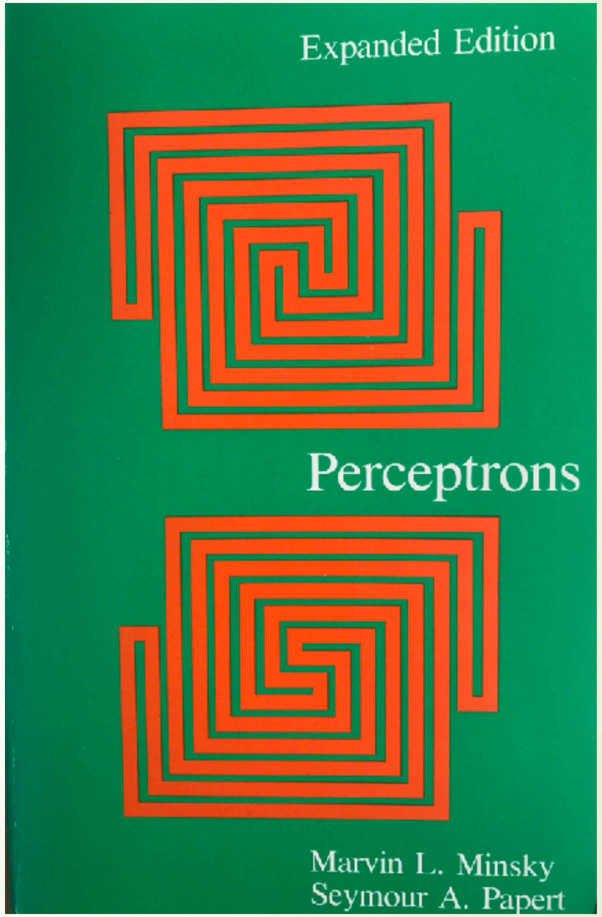
If *f* is a multivariate continuous function, then *f* can be written as a superposition of composite functions of mixtures of continuous functions of single variables:
finite **composition** of continuous functions of a **single variable** and the **addition**.

Locality or Sparsity of Computation

Minsky and Papert, 1969
Perceptron can't do **XOR** classification
Perceptron needs infinite global information to compute **connectivity**





Marvin Minsky   Seymour Papert



Locality or **Sparsity** is important:
Locality in time?
Locality in space?

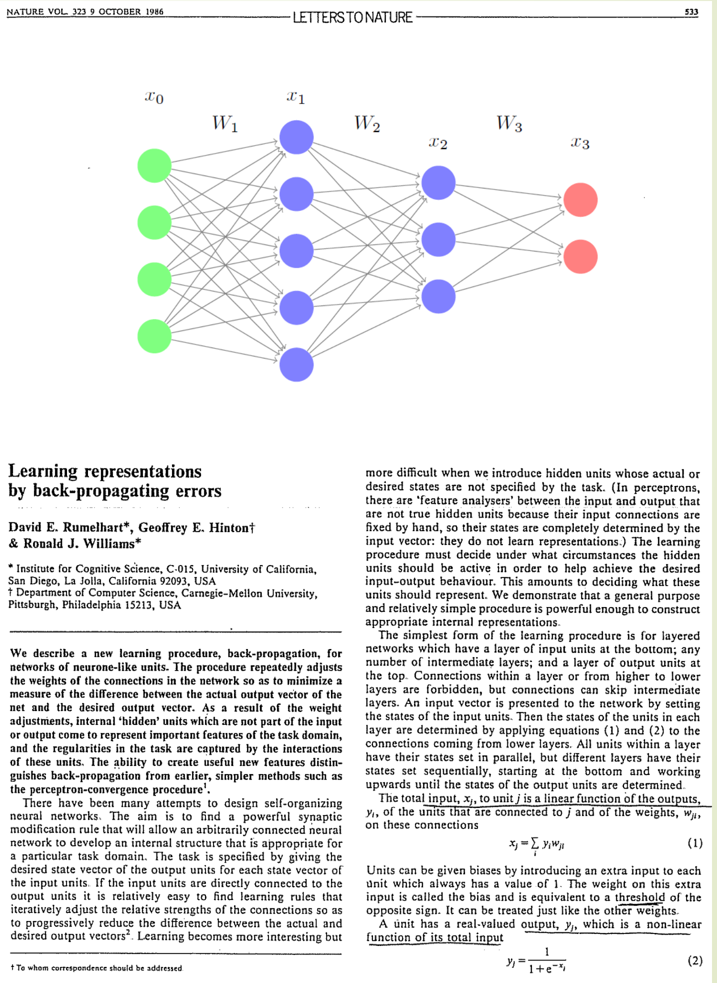
Multilayer Perceptrons (MLP) and Back-Propagation (BP) Algorithms

D.E. Rumelhart, G. Hinton, R.J. Williams (1986)
Learning representations by back-propagating errors, Nature, 323(9): 533-536

BP algorithms as **stochastic gradient descent** algorithms (**Robbins–Monro 1950; Kiefer-Wolfowitz 1951**) with Chain rules of Gradient maps

Deep network may classify **XOR**. Yet **topology**?

We address complexity and geometric invariant properties first.



NATURE VOL. 323 9 OCTOBER 1986 — LETTERS TO NATURE — 533

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \quad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \quad (2)$$

† To whom correspondence should be addressed

Parallel Distributed Processing

by Rumelhart and McClelland, 1986

Minsky and Papert set out to show which functions can and cannot be computed by this class of machines. They demonstrated, in particular, that such perceptrons are unable to calculate such mathematical functions as parity (whether an odd or even number of points are on in the retina) or the topological function of connectedness (whether all points that are on are connected to all other points that are on either directly or via other points that are also on) without making use of absurdly large numbers of predicates. The analysis is extremely elegant and demonstrates the importance of a mathematical approach to analyz-

of multilayer networks that compute parity). Similarly, it is not difficult to develop networks capable of solving the connectedness or inside/outside problem. Hinton and Sejnowski have analyzed a version of such a network (see Chapter 7).

Essentially, then, although Minsky and Papert were exactly correct in their analysis of the *one-layer perceptron*, the theorems don't apply to systems which are even a little more complex. In particular, it doesn't apply to multilayer systems nor to systems that allow feedback loops.

BP algorithm = Gradient Descent Method

- Training examples $\{x_0^i\}_{i=1}^n$ and labels $\{y^i\}_{i=1}^n$
- Output of the network $\{x_L^i\}_{i=1}^m$
- Objective   Square loss, cross-entropy loss, etc.

$$J(\{W_l\}, \{b_l\}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|y^i - x_L^i\|_2^2 \qquad (1)$$

- Gradient descent

$$W_l = W_l - \eta\frac{\partial J}{\partial W_l}$$

$$b_l = b_l - \eta\frac{\partial J}{\partial b_l}$$

In practice: use Stochastic Gradient Descent (SGD)

Derivation of BP: Lagrangian Multiplier

LeCun et al. 1988

Given n training examples $(I_i, y_i) \equiv$ (input,target) and L layers

- Constrained optimization

$$\min_{W,x} \quad \sum_{i=1}^{n} \|x_i(L) - y_i\|_2$$

subject to $x_i(\ell) = f_\ell\Big[W_\ell x_i(\ell - 1)\Big],$

$$i = 1, \ldots, n, \quad \ell = 1, \ldots, L, \; x_i(0) = I_i$$

- Lagrangian formulation (Unconstrained)

$$\min_{W,x,B} \mathcal{L}(W, x, B)$$

$$\mathcal{L}(W, x, B) = \sum_{i=1}^{n} \Bigg\{ \|x_i(L) - y_i\|_2^2 + \sum_{\ell=1}^{L} B_i(\ell)^T \Big(x_i(\ell) - f_\ell\Big[W_\ell x_i(\ell - 1)\Big]\Big)\Bigg\}$$

http://yann.lecun.com/exdb/publis/pdf/lecun-88.pdf

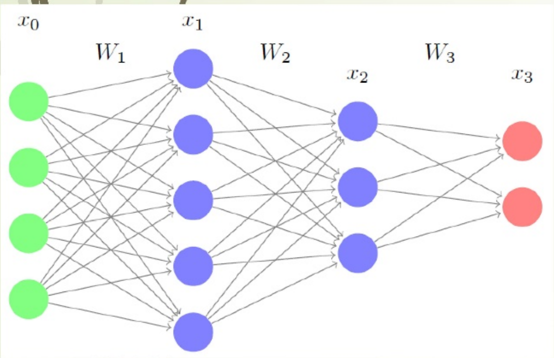
BP Algorithm: Forward Pass

- Cascade of repeated [linear operation followed by coordinatewise nonlinearity]'s
- Nonlinearities: sigmoid, hyperbolic tangent, (recently) ReLU.

Algorithm 1 Forward pass

Input: x_0
Output: x_L

for $\ell = 1$ to L **do**
    $x_\ell = f_\ell(W_\ell x_{\ell-1} + b_\ell)$
end for



back-propagation – derivation

- $\frac{\partial \mathcal{L}}{\partial B}$

Forward pass

$$x_i(\ell) = f_\ell\Big[\underbrace{W_\ell x_i\,(\ell-1)}_{A_i(\ell)}\Big] \quad \ell = 1,\ldots,L, \quad i = 1,\ldots,n$$

- $\frac{\partial \mathcal{L}}{\partial x}, z_\ell = [\nabla f_\ell]B(\ell)$

Backward (adjoint) pass

$$z(L) = 2\nabla f_L\Big[A_i(L)\Big]\,(y_i - x_i(L))$$

$$z_i(\ell) = \nabla f_\ell\Big[A_i(\ell)\Big]W_{\ell+1}^T z_i(\ell+1) \quad \ell = 0,\ldots,L-1$$

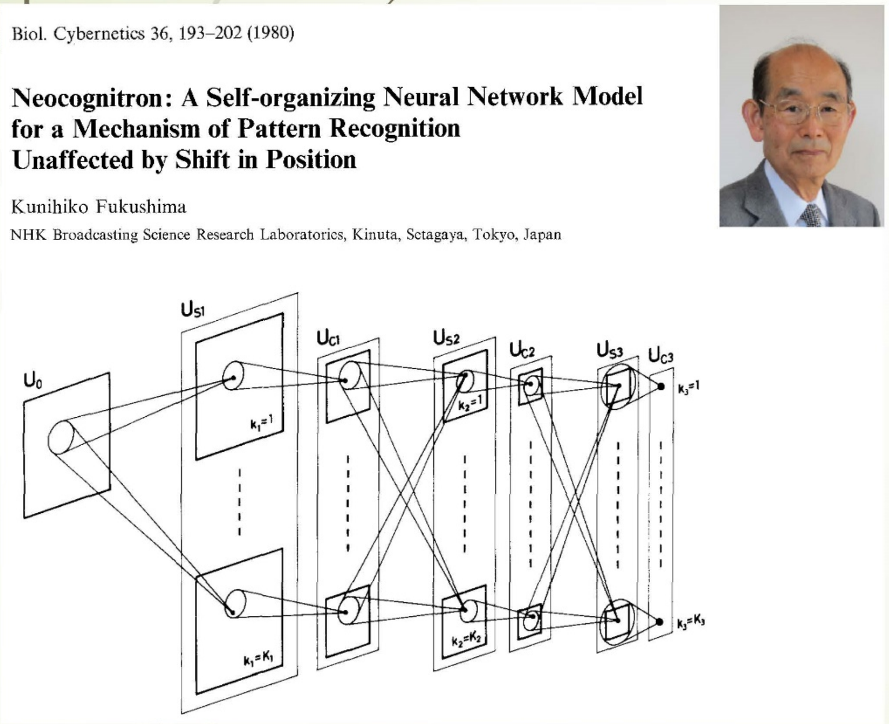
- $W \leftarrow W + \lambda \frac{\partial \mathcal{L}}{\partial W}$

Weight update

$$W_\ell \leftarrow W_\ell + \lambda \sum_{i=1}^n z_i(\ell) x_i^T(\ell-1)$$

Convolutional Neural Networks: shift invariances and locality

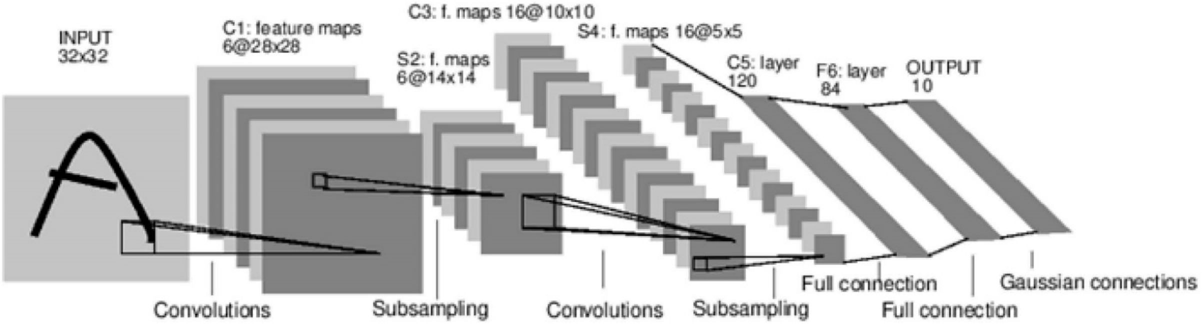
- Can be traced to *Neocognitron* of Kunihiko Fukushima (1979)
- Yann LeCun combined convolutional neural networks with back propagation (1989)
- Imposes **shift invariance** and **locality** on the weights
- Forward pass remains similar
- Backpropagation slightly changes – need to sum over the gradients from all spatial positions


Biol. Cybernetics 36, 193–202 (1980)

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position

Kunihiko Fukushima

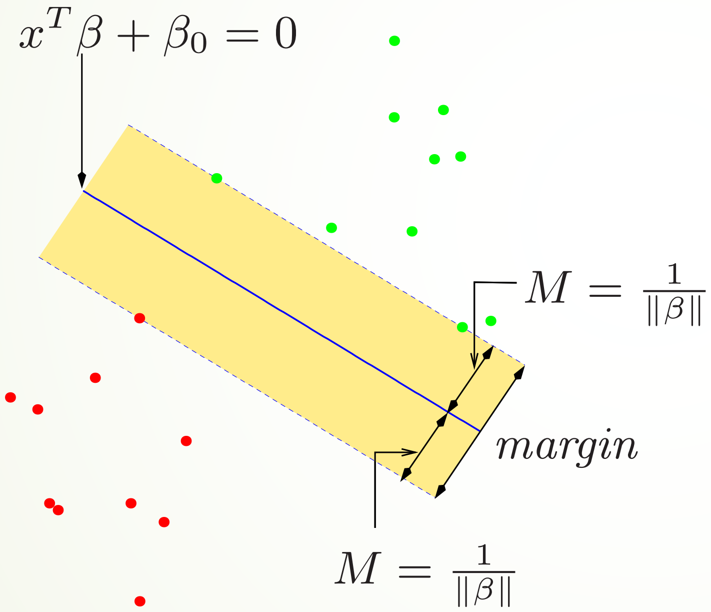
NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan

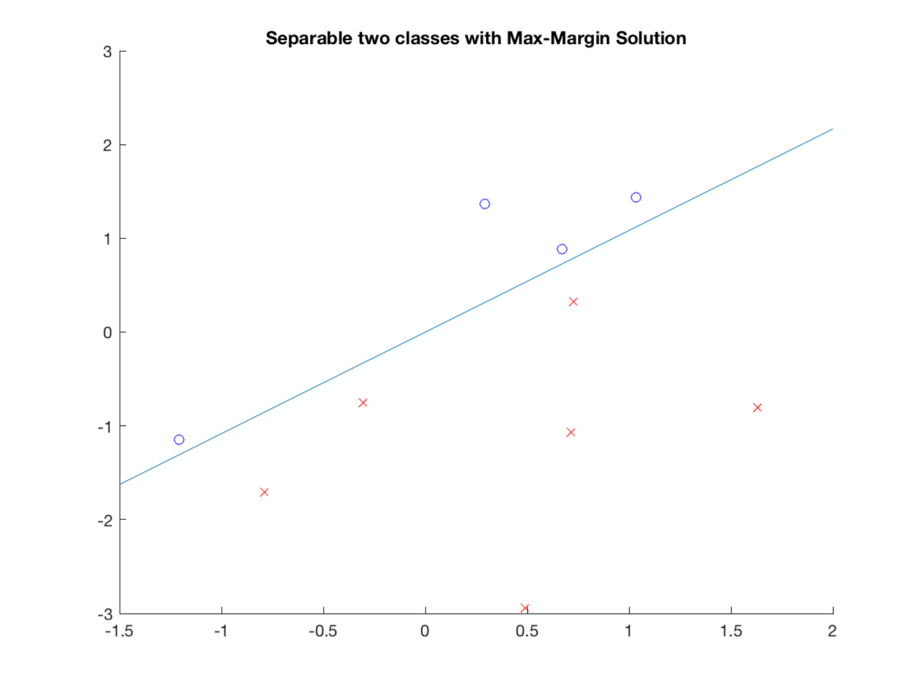


Max-Margin Classifier (SVM)

$$\text{minimize}_{\beta_0,\beta_1,\ldots,\beta_p} \|\beta\|^2 := \sum_j \beta_j^2$$

subject to $y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq 1$ for all i





Vladmir Vapnik, 1994

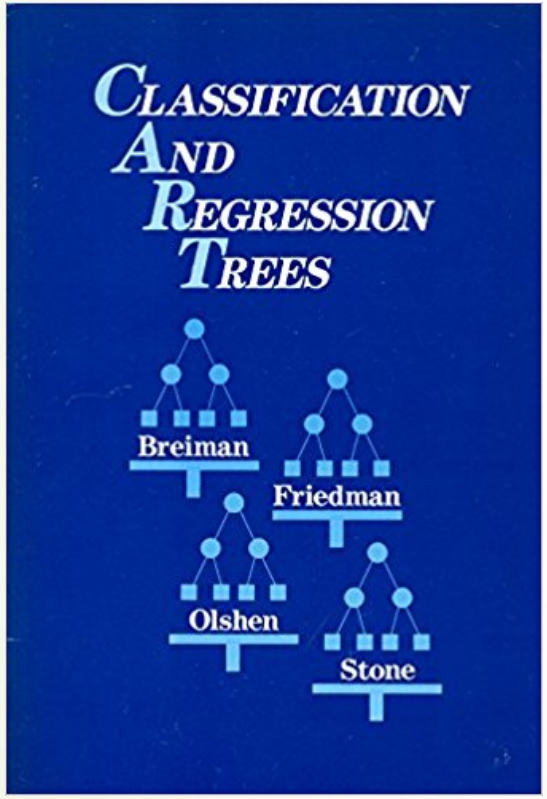
MNIST Dataset Test Error
LeCun et al. 1998

Simple SVM performs as well as Multilayer Convolutional Neural Networks which need careful tuning (LeNets)

Dark era for NN: 1998-2012

2000-2010: The Era of SVM, Boosting, … as nights of Neural Networks

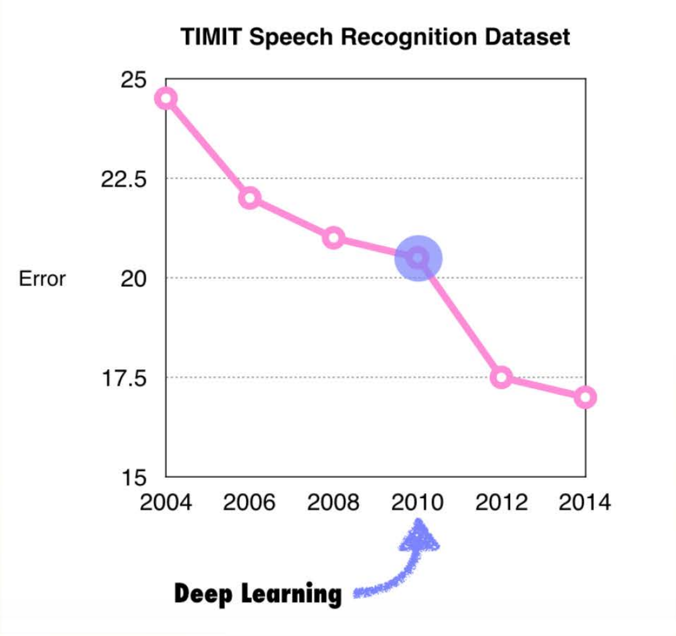
Decision Trees and Boosting



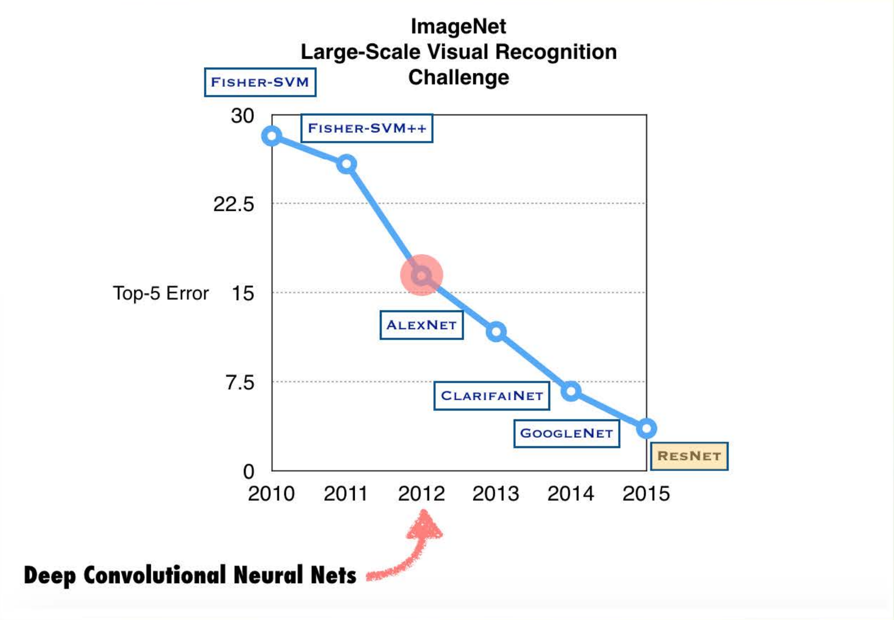
- Breiman, Friedman, Olshen, Stone, (1983): CART
- ``The Boosting problem'' (M. Kearns & L. Valiant): **Can a set of weak learners create a single strong learner**? (三个臭皮匠顶个诸葛亮？)
- Breiman (1996): Bagging
- Freund, Schapire (1997): **AdaBoost** ("the best off-the-shelf algorithm" by Breiman)
- Breiman (2001): **Random Forests**

Around the year of 2012: return of NN as `deep learning’

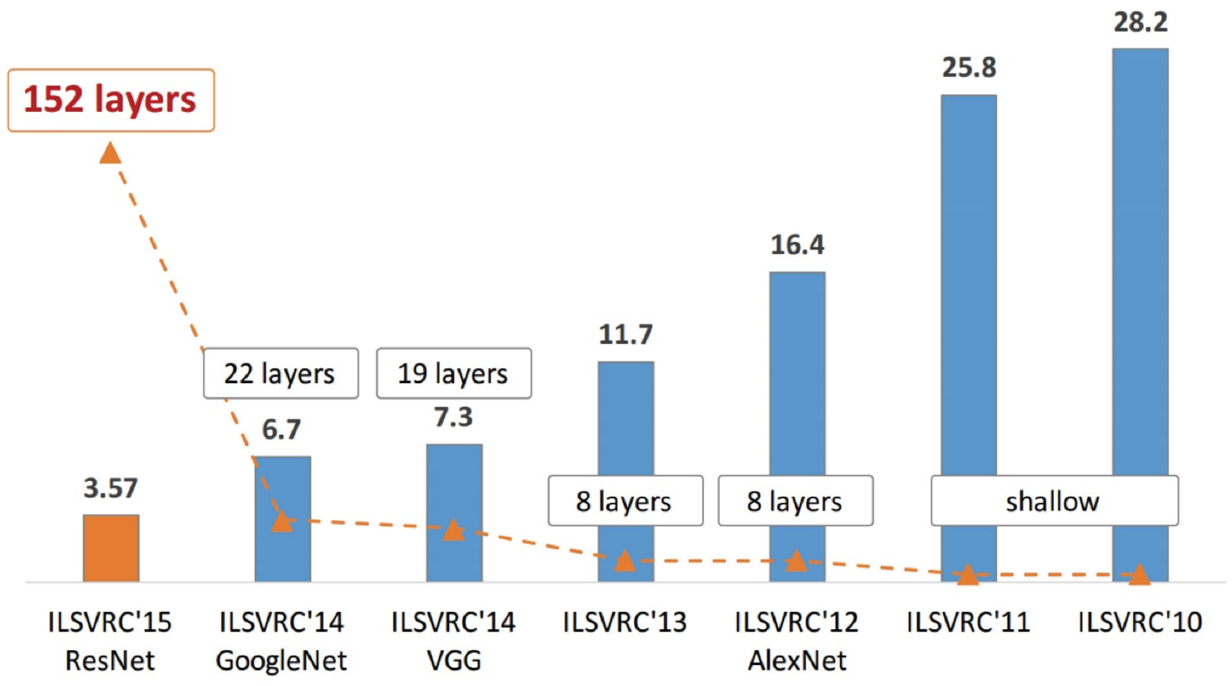
Speech Recognition: TIMIT



Computer Vision: ImageNet



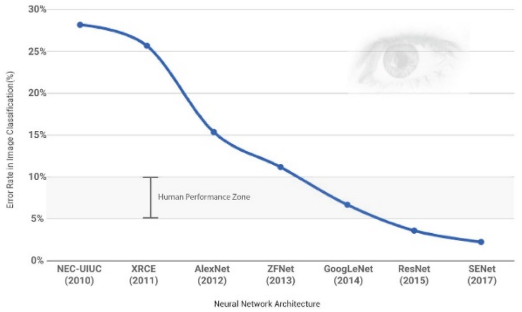
Depth as function of year



[He et al., 2016]

ILSVRC ImageNet Top 5 errors

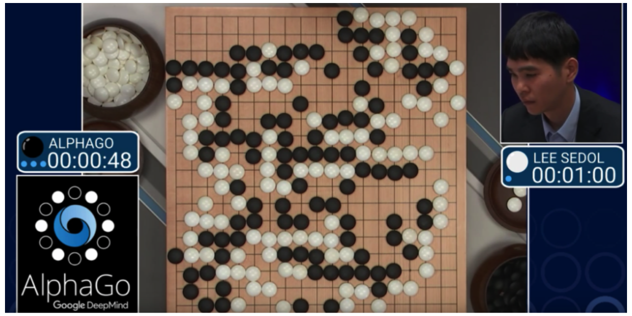
- ImageNet (subset):
  - 1.2 million training images
  - 100,000 test images
  - 1000 classes
- ImageNet large-scale visual recognition Challenge



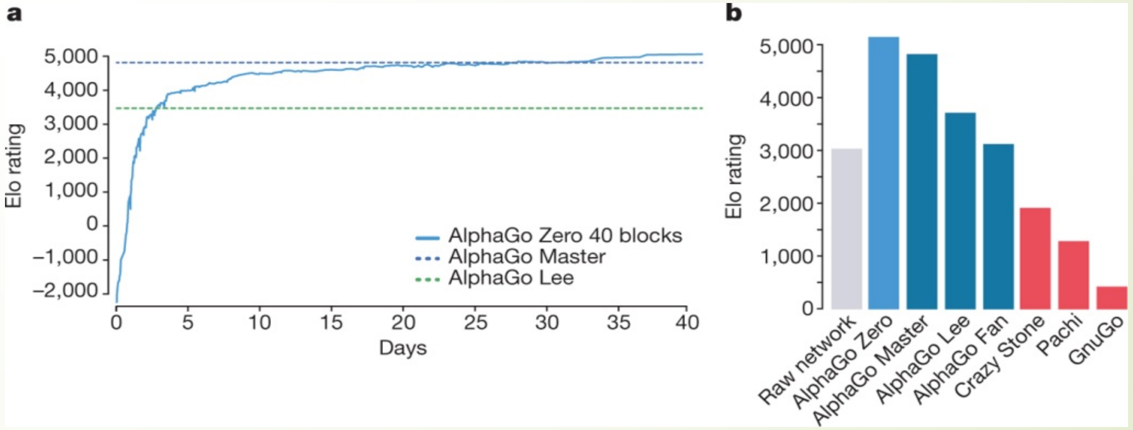
source: https://www.linkedin.com/pulse/must-read-path-breaking-papers-image-classification-muktabh-mayank

Reaching Human Performance Level in Games

Deep Blue in 1997



AlphaGo "LEE" 2016



AlphaGo "ZERO" D Silver *et al.* *Nature* **550,** 354–359 (2017) doi:10.1038/nature24270

Number of AI papers on arXiv, 2010-2019

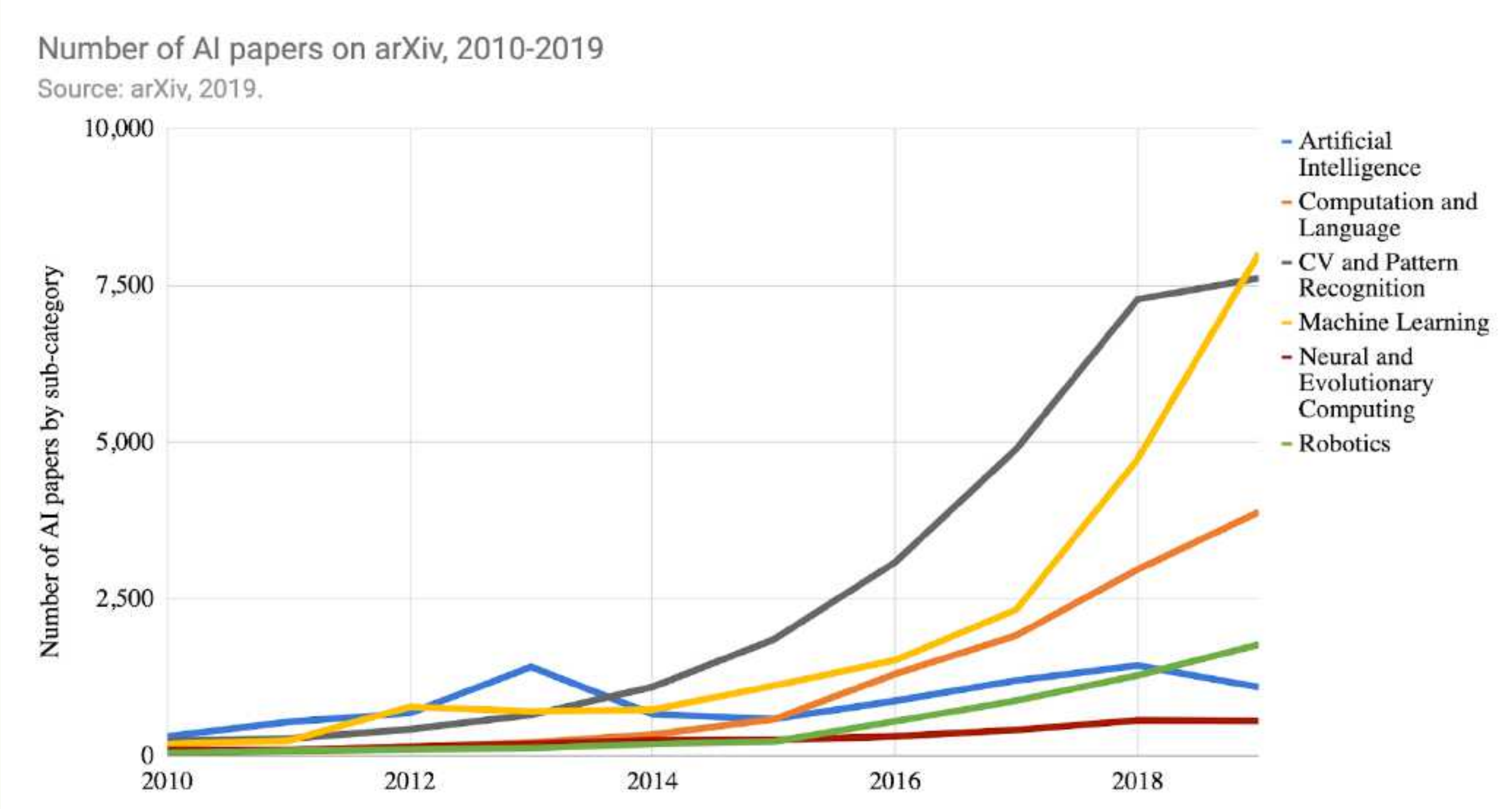
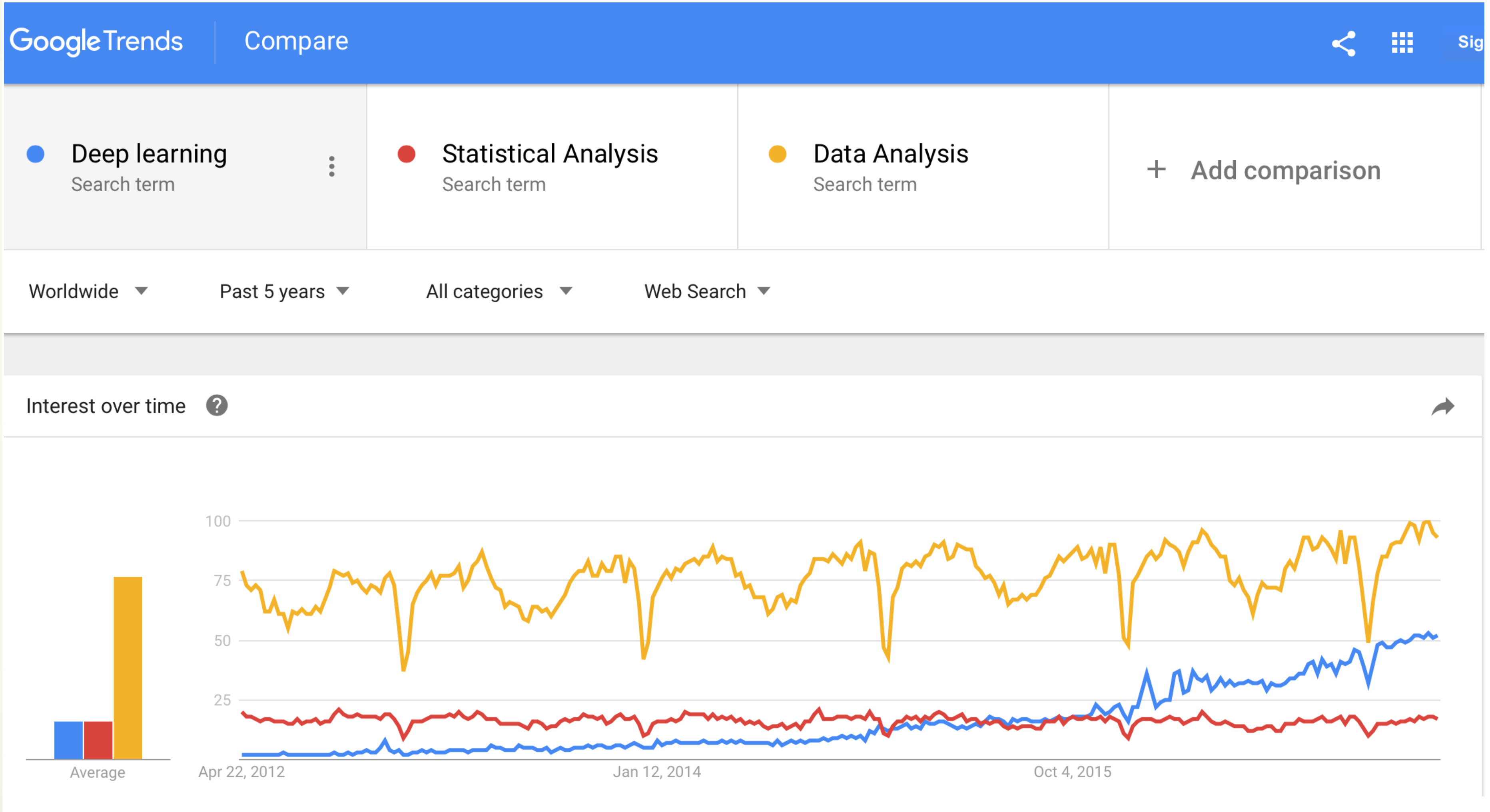


Fig. 1.6.

Growth of Deep Learning

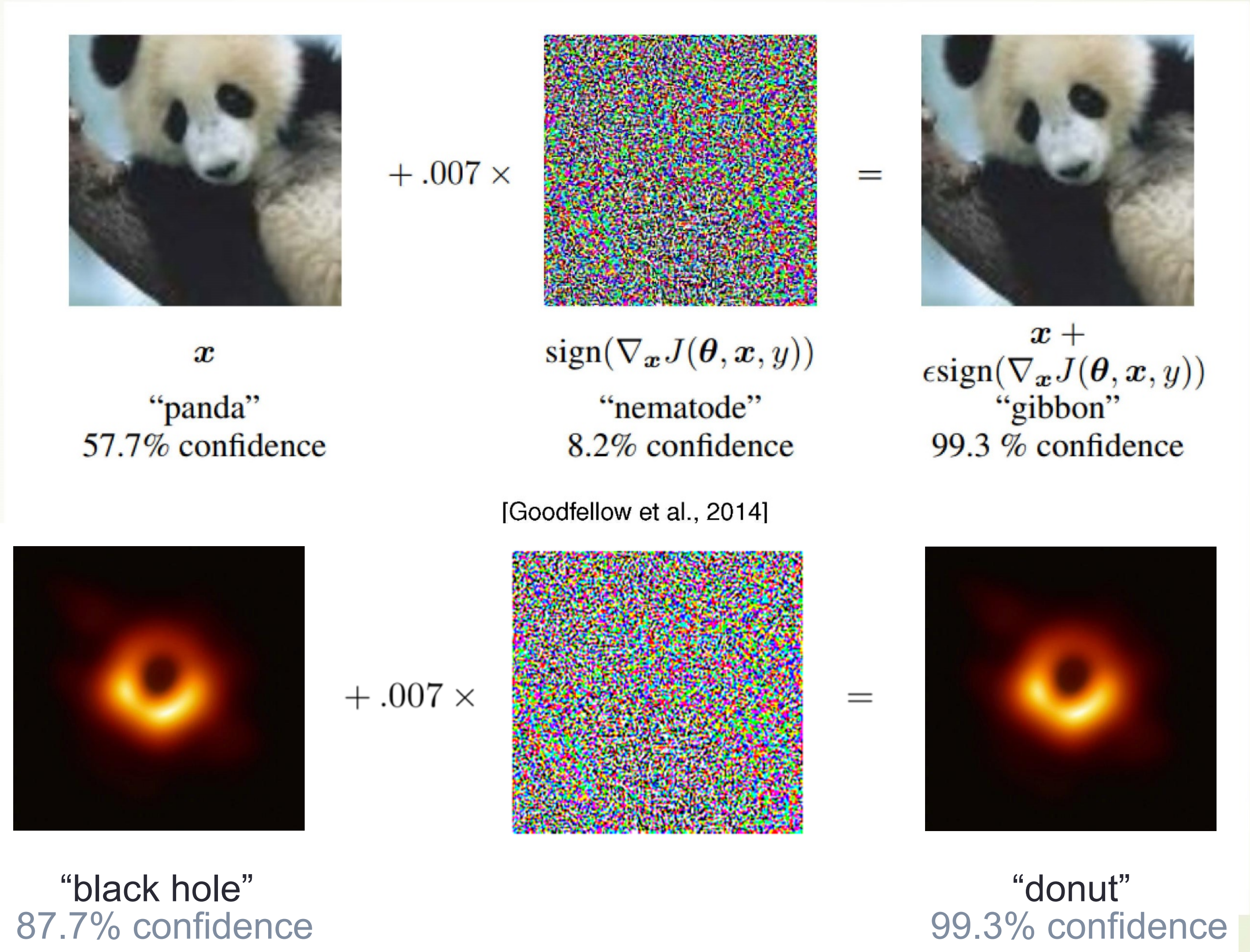
'Deep Learning' is coined by Hinton et al. in their Restricted Boltzman Machine paper, *Science* 2006, not yet popular until championing ImageNet competitions.



Some Cold Water: Tesla Autopilot Misclassifies Truck as Billboard

Problem: Why? How can you trust a blackbox?

Deep Learning may be fragile in generalization against noise!



CNN learns texture features, not shapes

(a) Texture image

81.4%	**Indian elephant**
10.3%	indri
8.2%	black swan

(b) Content image

71.1%	**tabby cat**
17.3%	grey fox
3.3%	Siamese cat

(c) Texture-shape cue conflict

63.9%	**Indian elephant**
26.4%	indri
9.6%	black swan

Geirhos et al. ICLR 2019

https://videoken.com/embed/W2HvLBMhCJQ?tocitem=46

1:16:47

Overfitting causes privacy leakage

- Model inversion attack leaks privacy

Figure: Recovered (Left), Original (Right)

Fredrikson et al. Proc. CCS, 2016

What's wrong with deep learning?

Ali Rahimi NIPS'17: Machine (deep) Learning has become **alchemy**.
https://www.youtube.com/watch?v=ORHFOnaEzPc

Yann LeCun CVPR'15, invited talk: **What's wrong with deep learning?** One important piece: **missing some theory (clarity in understanding)**!

http://techtalks.tv/talks/whats-wrong-with-deep-learning/61639/

Being alchemy is certainly not a shame, not wanting to work on advancing to chemistry is a shame! -- **by Eric Xing**

“Shall we see soon an emergence from Alchemy to Science in deep leaning?”

How can we teach our students in the next generation science rather than alchemy?

In this class

- Understand its principles: statistics, optimization
- Analyze the real world data with the methods
- Team-work (no more than 3 persons per team)!

Thank you!