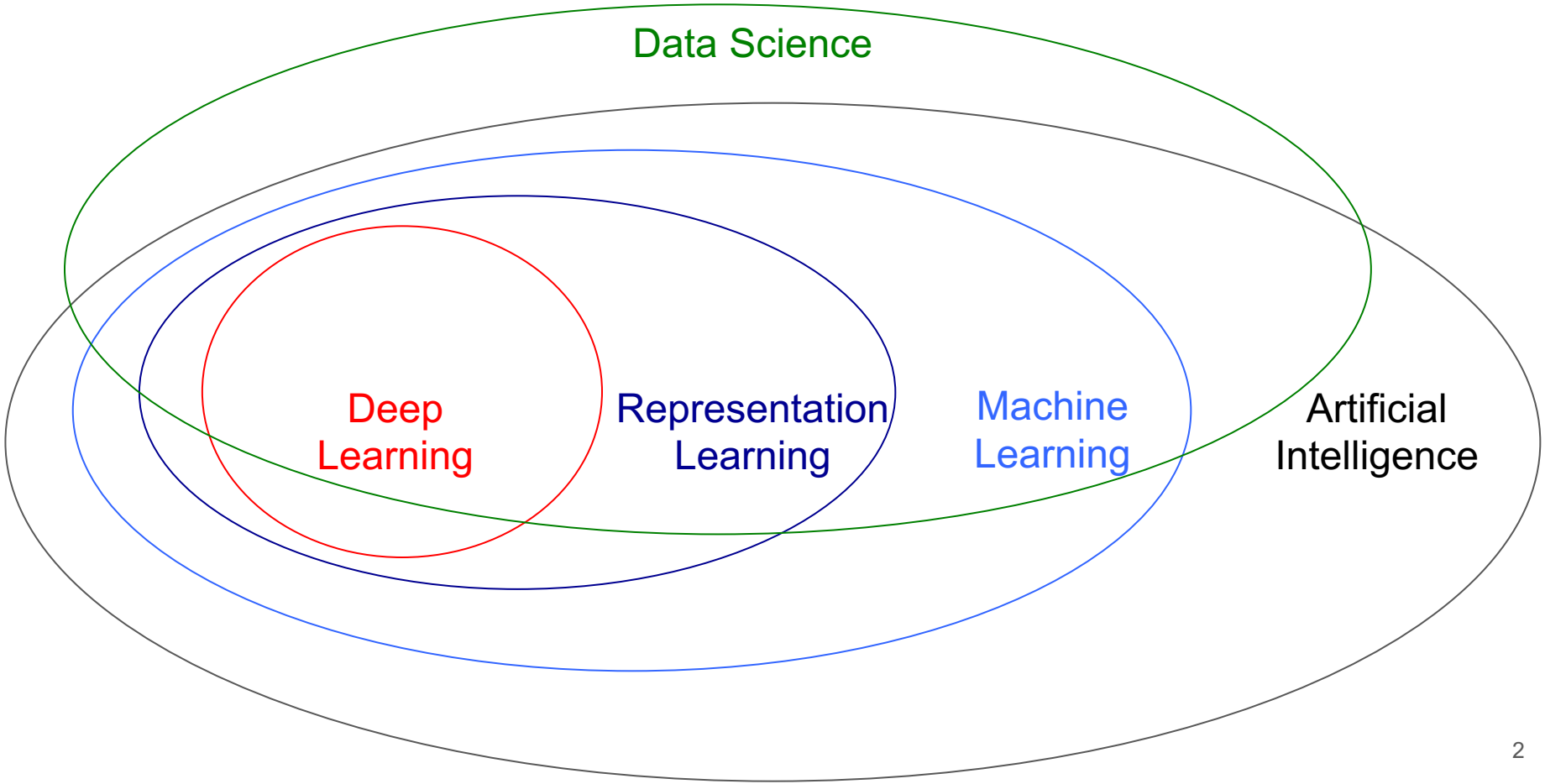
Biomedical Data Science: Mining and Modeling

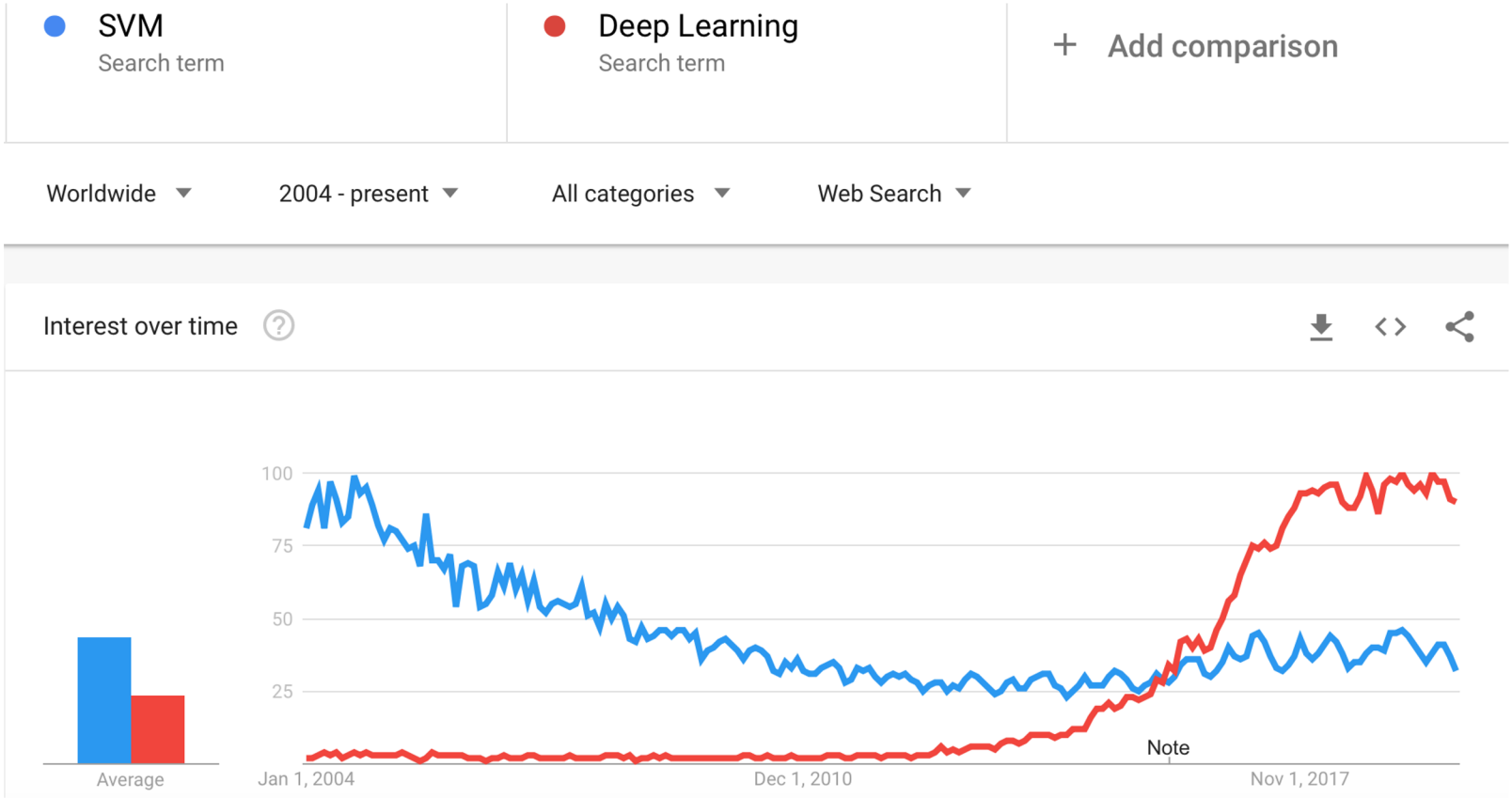
Deep Learning I: Introduction

Dr. Martin Renqiang Min
Department of Machine Learning
NEC Laboratories America

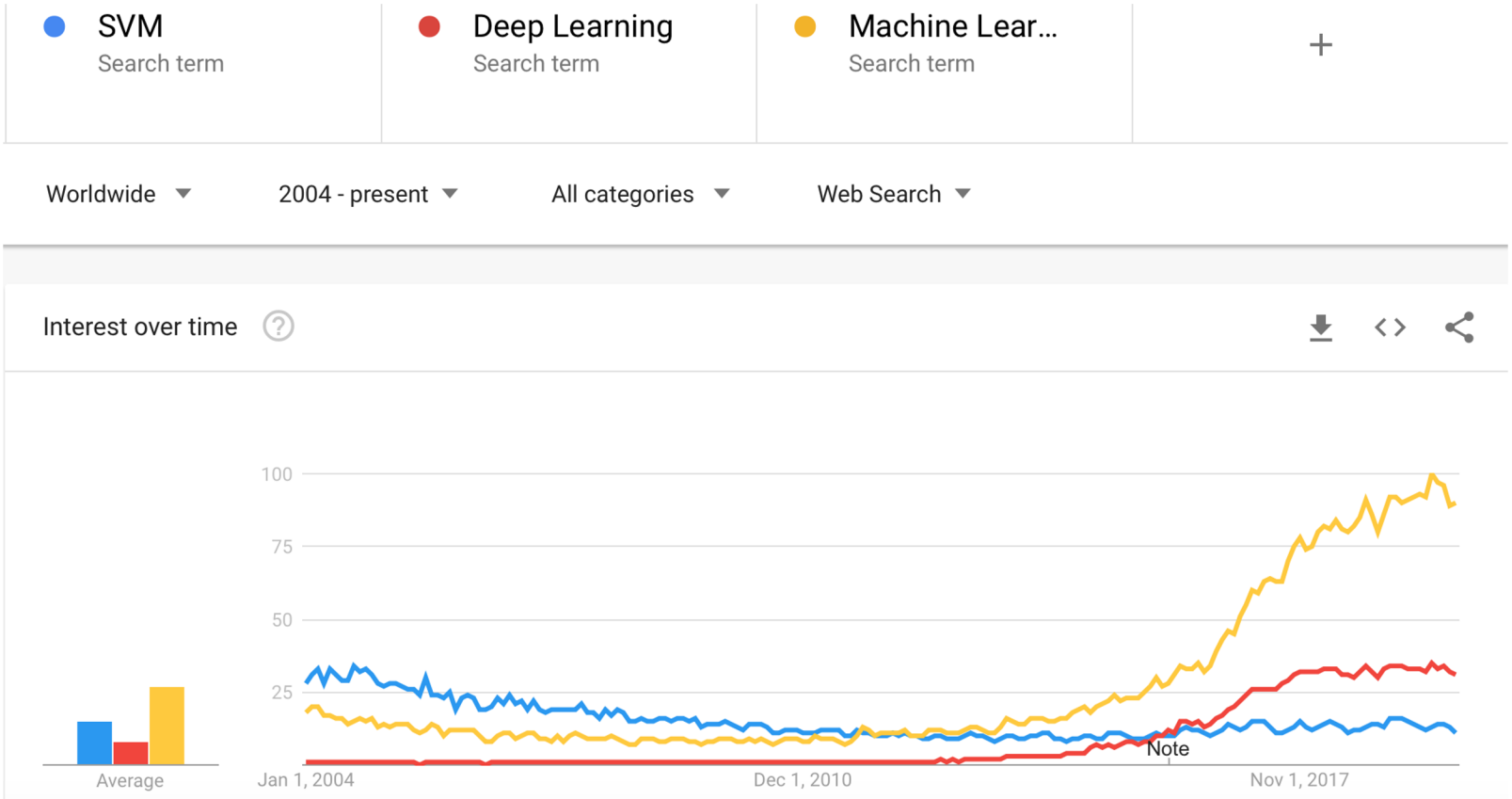
What is Deep Learning



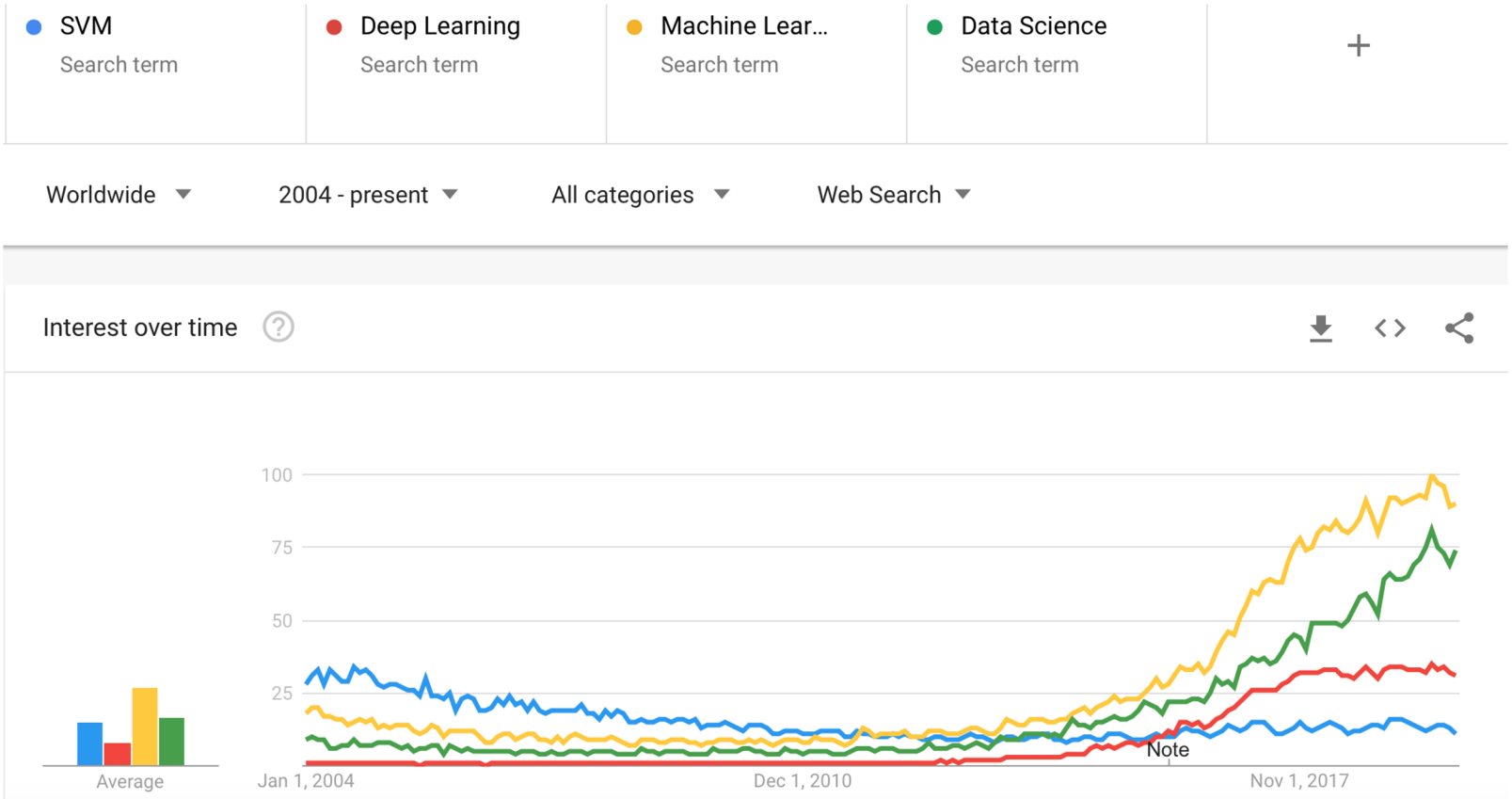
Google Trends: Deep Learning



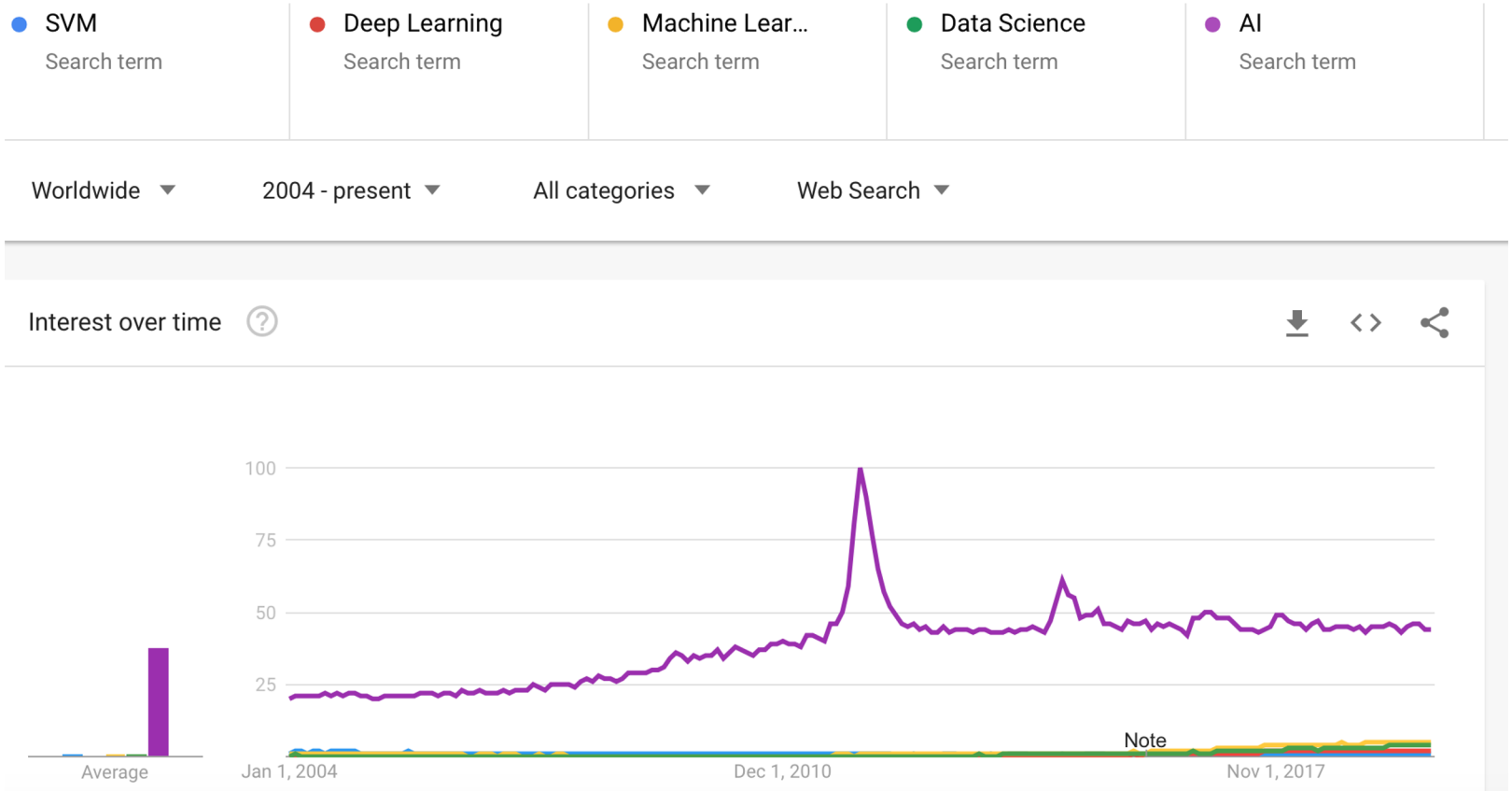
Google Trends: Deep Learning



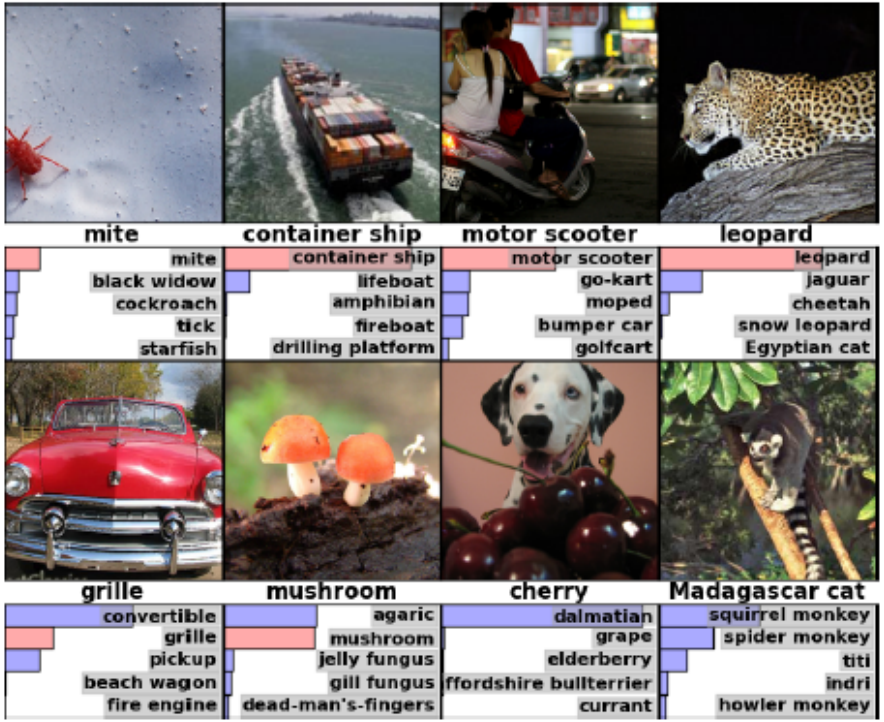
Google Trends: Deep Learning



Google Trends: Deep Learning and AI



The AI Revolution is Driven by Deep Learning: The ImageNet Challenge in Computer Vision



The AI Revolution is Driven by Deep Learning: The ImageNet Challenge in Computer Vision

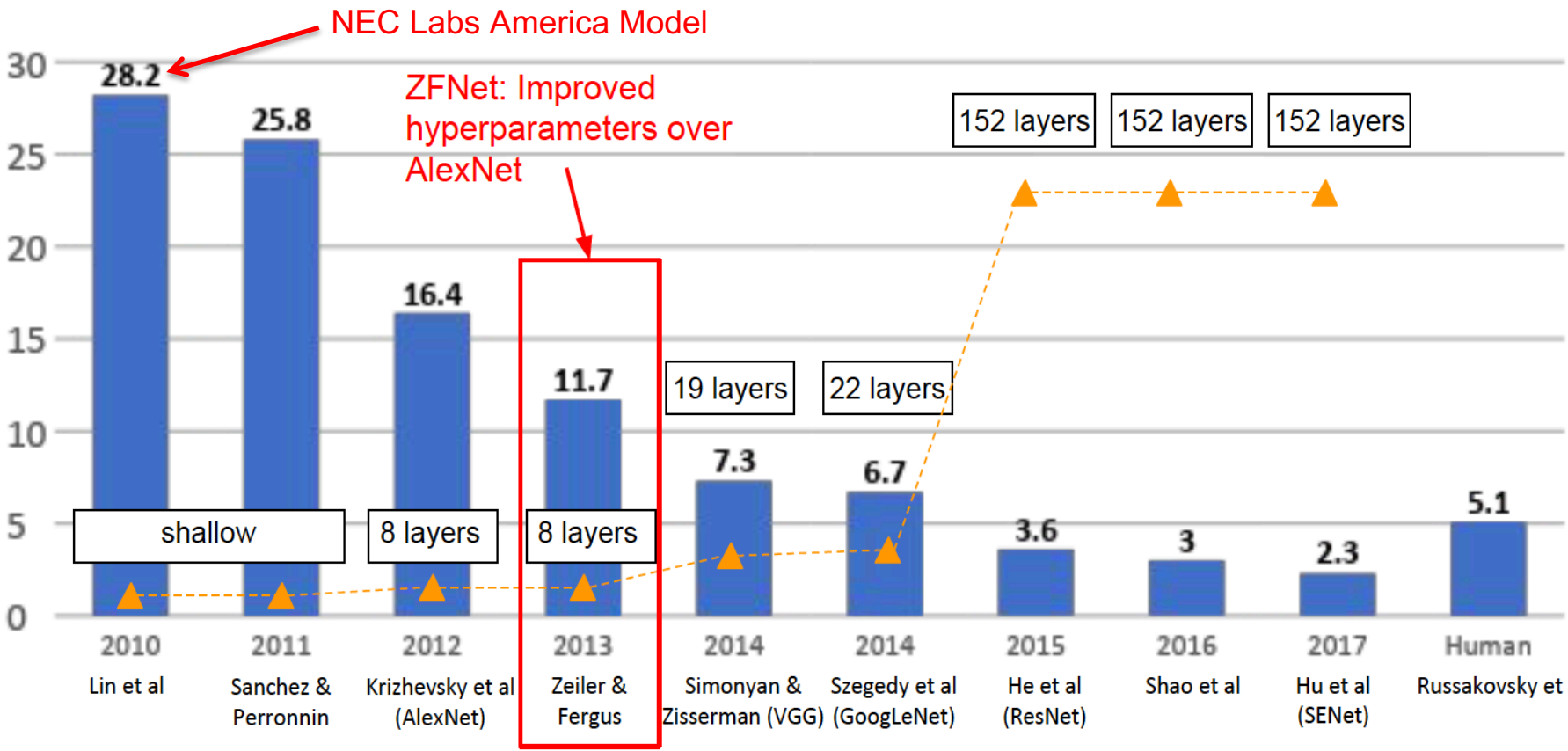
[PDF] ImageNet Classification with Deep Convolutional Neural ...

https://papers.nips.cc › paper › 4824-imagenet-classification-with-deep-co...

by A Krizhevsky - 2012 - Cited by 54415 - Related articles

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 dif-.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

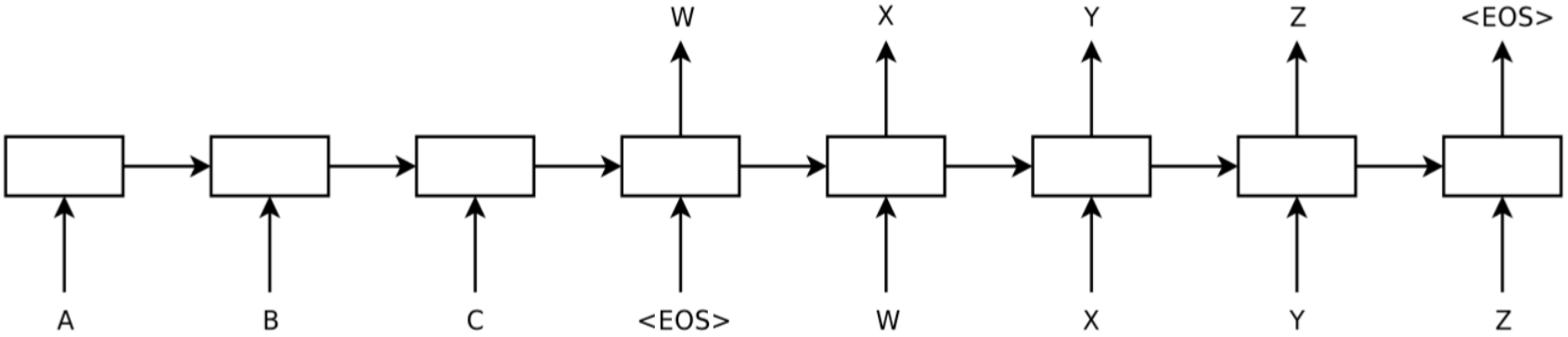


The AI Revolution is Driven by Deep Learning: Speech Recognition

Deep neural networks for acoustic modeling in speech recognition

Authors	Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, Tara Sainath
Publication date	2012/11/1
Journal	IEEE Signal processing magazine
Volume	29
Description	Gaussian mixture models (GMMs) to determine how well each state of each HMM fits a frame or a short window of frames of coefficients that represents the acoustic input. An alternative way to evaluate the fit is to use a feed-forward neural network that takes several frames of coefficients as input and produces posterior probabilities over HMM states as output. Deep neural networks (DNNs) that have many hidden layers and are trained using new methods have been shown to outperform GMMs on a variety of speech recognition benchmarks, sometimes by a large margin. This article provides an overview of this progress and represents the shared views of four research groups that have had recent successes in using DNNs for acoustic modeling in speech recognition.
Total citations	Cited by 7057

The AI Revolution is Driven by Deep Learning: Machine Translation



Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Sutskever, Vinyals, and Le, NIPS 2014

The AI Revolution is Driven by Deep Learning: Machine Translation

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

(Submitted on 26 Sep 2016 (v1), last revised 8 Oct 2016 (this version, v2))

Neural Machine Translation (NMT) is an end-to-end learning approach for automated translation, with the potential to overcome many of the weaknesses of conventional phrase-based translation systems. Unfortunately, NMT systems are known to be computationally expensive both in training and in translation inference. Also, most NMT systems have difficulty with rare words. These issues have hindered NMT's use in practical deployments and services, where both accuracy and speed are essential. In this work, we present GNMT, Google's Neural Machine Translation system, which attempts to address many of these issues. Our model consists of a deep LSTM network with 8 encoder and 8 decoder layers using attention and residual connections. To improve parallelism and therefore decrease training time, our attention mechanism connects the bottom layer of the decoder to the top layer of the encoder. To accelerate the final translation speed, we employ low-precision arithmetic during inference computations. To improve handling of rare words, we divide words into a limited set of common sub-word units ("wordpieces") for both input and output. This method provides a good balance between the flexibility of "character"-delimited models and the efficiency of "word"-delimited models, naturally handles translation of rare words, and ultimately improves the overall accuracy of the system. Our beam search technique employs a length-normalization procedure and uses a coverage

The AI Revolution is Driven by Deep Learning: Video Game

MENU

nature

Letter | Published: 25 February 2015

Human-level control through deep reinforcement learning

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis

Nature **518**, 529–533(2015) | Cite this article

74k Accesses | **2533** Citations | **1517** Altmetric | Metrics

The AI Revolution is Driven by Deep Learning: Video Game

https://deepmind.com/blog/article/deep-reinforcement-learning

The AI Revolution is Driven by Deep Learning: Go Game

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol

theguardian.com
Mar 2016

DeepMind's artificial intelligence astonishes fans to defeat human opponent and offers evidence computer software has mastered a major challenge

▲ The world's top Go player, Lee Sedol, lost the final game of the Google DeepMind challenge match. Photograph: Yonhap/Reuters

Learn How to Play Chess without Human Knowledge

ARTICLE

doi:10.1038/nature2427

Mastering the game of Go without human knowledge

David Silver[1]*, Julian Schrittwieser[1]*, Karen Simonyan[1]*, Ioannis Antonoglou[1], Aja Huang[1], Arthur Guez[1], Thomas Hubert[1], Lucas Baker[1], Matthew Lai[1], Adrian Bolton[1], Yutian Chen[1], Timothy Lillicrap[1], Fan Hui[1], Laurent Sifre[1], George van den Driessche[1], Thore Graepel[1] & Demis Hassabis[1]

A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting *tabula rasa*, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play

David Silver[1,2,*,†], Thomas Hubert[1,*], Julian Schrittwieser[1,*], Ioannis Antonoglou[1], Matthew Lai[1], Arthur Guez[1], Marc Lanct…
+ See all authors and affiliations

Science 07 Dec 2018:
Vol. 362, Issue 6419, pp. 1140-1144
DOI: 10.1126/science.aar6404

One program to rule them all

Computers can beat humans at increasingly complex games, including chess and Go. However, these programs are typically constructed for a particular game, exploiting its properties, such as the symmetries of the board on which it is played. Silver *et al.* developed a program called AlphaZero, which taught itself to play Go, chess, and shogi (a Japanese version of chess) (see the Editorial, and the Perspective by Campbell). AlphaZero managed to beat state-of-the-art programs specializing in these three games. The ability of AlphaZero to

The AI Revolution is Driven by Deep Learning: Molecule (Drug) Synthesis

ARTICLE

doi:10.1038/nature25978

Planning chemical syntheses with deep neural networks and symbolic AI

Marwin H. S. Segler[1,2], Mike Preuss[3] & Mark P. Waller[4]

To plan the syntheses of small organic molecules, chemists use retrosynthesis, a problem-solving technique in which target molecules are recursively transformed into increasingly simpler precursors. Computer-aided retrosynthesis would be a valuable tool but at present it is slow and provides results of unsatisfactory quality. Here we use Monte Carlo tree search and symbolic artificial intelligence (AI) to discover retrosynthetic routes. We combined Monte Carlo tree search with an expansion policy network that guides the search, and a filter network to pre-select the most promising retrosynthetic steps. These deep neural networks were trained on essentially all reactions ever published in organic chemistry. Our system solves for almost twice as many molecules, thirty times faster than the traditional computer-aided search method, which is based on extracted rules and hand-designed heuristics. In a double-blind AB test, chemists on average considered our computer-generated routes to be equivalent to reported literature routes.

Nature, Mar 2018

The AI Revolution is Driven by Deep Learning: Poker Game

NATURE | NEWS

How rival bots battled their way to poker supremacy

Artificial-intelligence programs harness game-theory strategies and deep learning to defeat human professionals in two-player hold 'em.

Elizabeth Gibney

02 March 2017

The AI Revolution is Driven by Deep Learning: Poker Game

MENU

nature

Subscribe

NEWS · 11 JULY 2019

No limit: AI poker bot is first to beat professionals at multiplayer game

Triumph over five human opponents at Texas hold'em brings bots closer to solving complicated real-world problems.

Douglas Heaven

The AI Revolution is Driven by Deep Learning: Strategic Game

nature

Article | Published: 30 October 2019

Grandmaster level in StarCraft II using multi-agent reinforcement learning

Oriol Vinyals ✉, Igor Babuschkin, [...] David Silver ✉

Nature **575**, 350–354(2019) | Cite this article

44k Accesses | **4** Citations | **961** Altmetric | Metrics

Abstract

Many real-world applications require artificial agents to compete and coordinate with other agents in complex environments. As a stepping stone to this goal, the domain of StarCraft has emerged as an important challenge for artificial intelligence research, owing to its iconic and enduring status among the most difficult professional esports and its relevance to the real world in terms of its raw complexity and multi-agent challenges. Over the course of a decade and numerous competitions[1,2,3],

The AI Revolution is Driven by Deep Learning: Strategic Game

https://www.nature.com/articles/d41586-019-03343-4

The AI Revolution is Driven by Deep Learning: A Powerful Language Model

Legolas and Gimli advanced on the orcs, raising their weapons and shouting in unison. The orcs swung their weapons at them, but the two heroes were ready. They leapt from their horses and charged the orcs, slashing and slashing until their blades were dull and their weapons were broken. They were the first to fall, but the orcs were not done yet. They rushed the two heroes, and they fell in a hail of arrows.

The orcs were not finished yet, however. They charged the two heroes again, and they fell in a hail of arrows. The orcs were not finished yet. They charged the two heroes again, and they fell in a hail of arrows.

The orcs were not finished yet, however. They charged the two heroes again, and they fell in a hail of arrows. The orcs were not finished yet.. The orcs were not far off, and the two of them were not far from the edge of the cliff.

Gimli turned to face the orcs, who were now close enough to see him. He raised his hands and shouted, "I'm Gimli, son of Thráin, and I'm here to help you!"

The orcs were confused. They had seen Gimli before, but he was not the same man who had saved them from the orcs in the cave. Gimli was a tall and powerful man, and he had a beard and a moustache. He was also a dwarf, and he had a strong build, and he was covered in tattoos. He was not a man who looked like a hobbit.

Artificial intelligence is moving into movie production. SHAREGRID/UNSPLASH

New algorithm can create movies from just a few snippets of text

By Matthew Hutson | Feb. 23, 2018 , 4:35 PM


thispersondoesnotexist.com
Imagined by a GAN (generative adversarial network)
StyleGAN2 (Dec 2019) - Karras et al. and Nvidia
Don't panic. Learn how it works [1] [2] [3] [4]
Help this AI continue to dream | Contact me
Code for training your own [original] [simple]
Another | Save • Cats | Articles | Friends | Office

Deep Learning Methods Can Be Easily Fooled

Computer Science > Computer Vision and Pattern Recognition

Intriguing properties of neural networks

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus

(Submitted on 21 Dec 2013 (v1), last revised 19 Feb 2014 (this version, v4))

Deep neural networks are highly expressive models that have recently achieved state of the art performance on speech and visual recognition tasks. While their expressiveness is the reason they succeed, it also causes them to learn uninterpretable solutions that could have counter-intuitive properties. In this paper we report two such properties.

First, we find that there is no distinction between individual high level units and random linear combinations of high level units, according to various methods of unit analysis. It suggests that it is the space, rather than the individual units, that contains of the semantic information in the high layers of neural networks.

Second, we find that deep neural networks learn input-output mappings that are fairly discontinuous to a significant extend. We can cause the network to misclassify an image by applying a certain imperceptible perturbation, which is found by maximizing the network's prediction error. In addition, the specific nature of these perturbations is not a random artifact of learning: the same perturbation can cause a different network, that was trained on a different subset of the dataset, to misclassify the same input.

Deep Learning Methods Can Be Easily Fooled

(a) (b)

Figure 5: Adversarial examples generated for AlexNet [9].(Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an *"ostrich, Struthio camelus"*. Average distortion based on 64 examples is 0.006508. Plase refer to `http://goo.gl/huaGPb` for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved.

Deep Learning Methods Can Be Easily Fooled

TOM SIMONITE SECURITY 08.25.2017 11:00 AM

Even Artificial Neural Networks Can Have Exploitable 'Backdoors'

Malicious machine learning can hide nasty surprises.

GETTY IMAGES

14,677 views | Nov 30, 2019, 07:20am

AI Is Not Similar To Human Intelligence. Thinking So Could Be Dangerous

Elizabeth Fernandez Contributor ⓘ
Science
I write about the philosophy and ethics of science and technology.

TECH TRANSFORMERS

Stephen Hawking says A.I. could be ‘worst event in the history of our civilization’

PUBLISHED MON, NOV 6 2017•2:10 PM EST , NOV 6 2017•3:39 PM EST

Arjun Kharpal
@ARJUNKHARPAL

Face with Cold Sweat Emoji (U…
iemoji.com

SHARE

Behind the Scene: Deep Learning

- Deep Learning:
  - The driving force of all these technological advancements
  - The root cause of all these controversial debates and potential dangers

Deep Learning is a Subfield of Machine Learning

- What is Machine Learning
  - Machine learning is about teaching computers to perform tasks by only showing them data/examples without explicitly programming instructions
  - Machine learning, data science, and computer science are more and more related to each other
  - Machine Learning: Instead of programming computers, let computers learn to program by themselves by showing them some examples
    - Feature Representation + Objective Function + Optimization (training data)
      → Generalization (test data)

Three Types of Machine Learning Tasks

- Supervised Learning:
  - Given training examples of input feature vectors and corresponding target outputs, predict outputs on future inputs. For e.g., classification, regression, time series prediction.
- Unsupervised Learning:
  - Given only input feature vectors, automatically discover representations, structures, etc. For e.g., clustering, data compression, outlier detection
- Reinforcement Learning:
  - Given sequences of [inputs from an environment, actions from a fixed set, and occasional scalar rewards], learn to select action sequences in a way that maximizes the expected sum of (discounted) future reward

Machine Learning Models

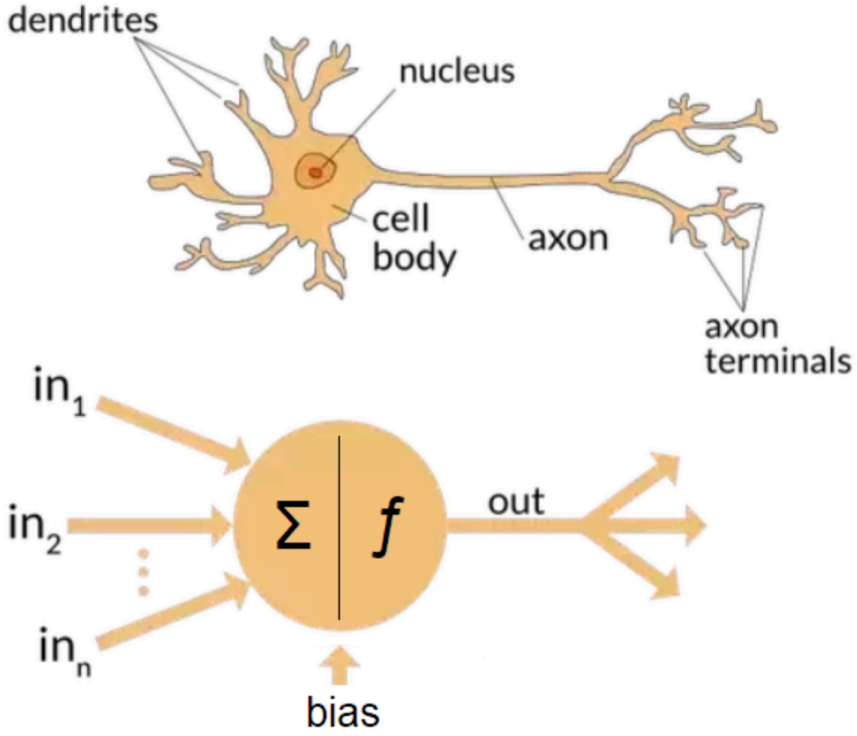
- Supervised Learning:
  - Linear Regression, Logistic Regression, Naïve Bayes, Linear Discriminant Analysis, k Nearest Neighbor, (kernel) SVM, Decision Tree, Random Forest, Multilayer Perceptron (Neural Network for classification/regression), Conditional Random Field
- Unsupervised Learning:
  - K-means, Gaussian Mixture Model, Hierarchical Clustering Methods, Principal Component Analysis, Independent Component analysis, Vector Quantification (Data Compression), Latent Dirichlet Allocation, (Denoising) Autoencoder, (Restricted) Boltzmann Machines
- Reinforcement Learning:
  - Q Learning, Policy Gradient Methods, REINFORCE

What is Deep Learning

- Deep learning
  - One of many research areas in Machine Learning
  - Deep Learning focuses on learning data representations (adaptive features) using deep neural networks
  - Deep neural networks are standard neural networks with many hidden layers
  - Deep learning can be applied to any type of learning task of machine learning
- Organization of deep learning topics in this course:
  - Deep Supervised Learning: Deep CNN/RNN for image classification/sequence classification
  - Deep Unsupervised Learning: Deep Autoencoder, Deep Generative Models
  - Deep Reinforcement Learning: AlphaGO, AlphaZero

Since deep learning is a modern name of deep neural network invented decades ago, I will review the history of (deep) neural network and introduce backpropagation in the remaining part of this lecture.

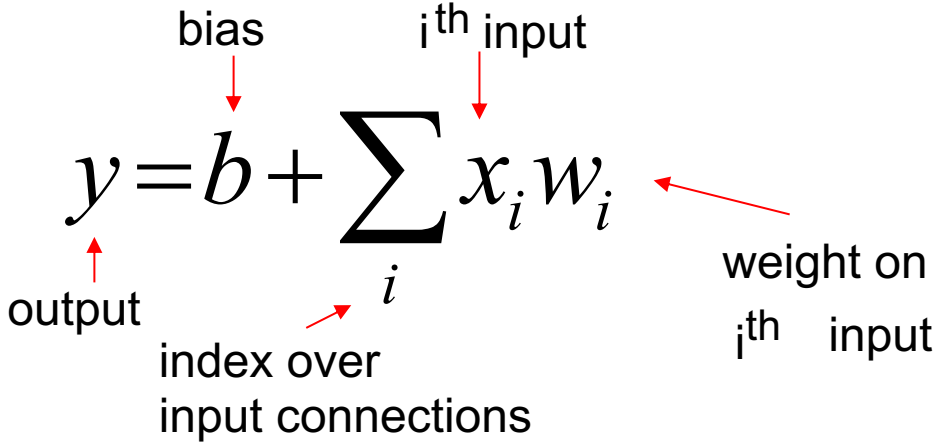
Modeling a Biological Neuron



A biological and an artificial neuron (via https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network)

https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

A Linear Neuron

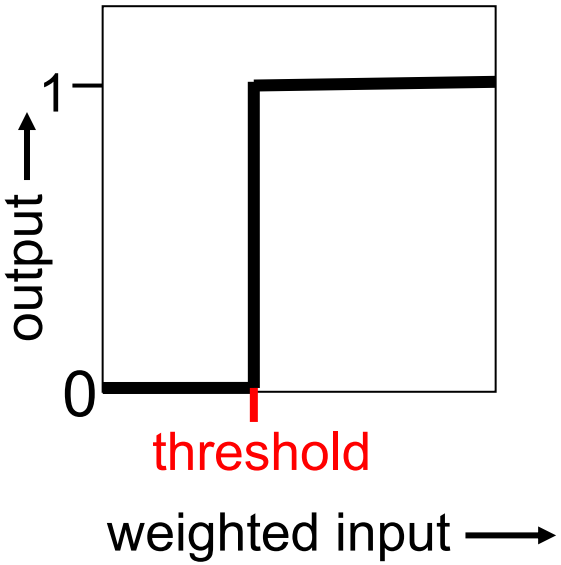


Binary Threshold Neuron

- McCulloch-Pitts (1943): influenced Von Neumann.
  - First compute a weighted sum of the inputs.
  - Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.

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

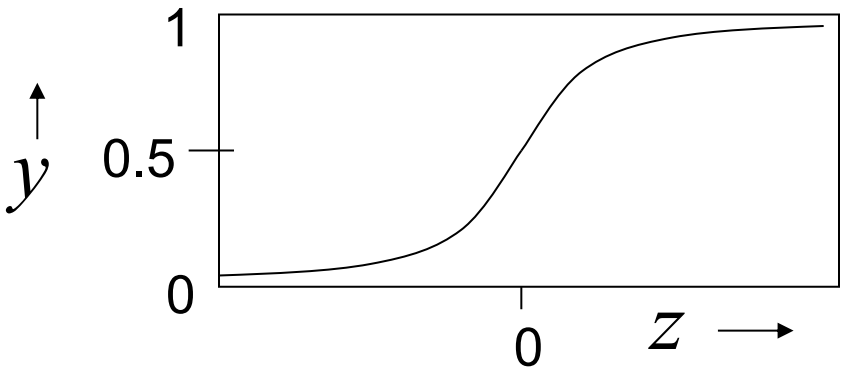
$$y = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ otherwise} \end{cases}$$



A Sigmoid Neuron

- It gives a real-valued output that is a smooth and bounded function of its total input.
  - It has a nice probabilistic interpretation ([0, 1]).
  - It has nice derivatives that make learning easy (discuss it later).

$$z = b + \sum_i x_i w_i \qquad y = \frac{1}{1 + e^{-z}}$$



Rosenblatt's Perceptron (1958)

- A linear neuron

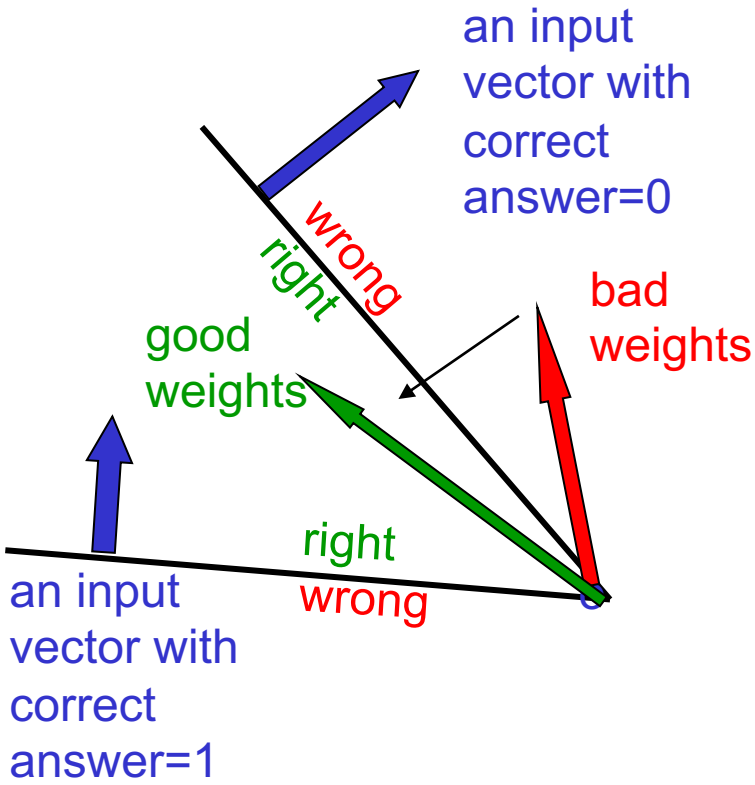
Learning algorithm for binary classification with a binary output neuron (1 or 0):

1. Initialize the weights and threshold to 0 or small random values
2. For each example i in the training set, perform the following steps:
    - Calculate the model output for input i using the current model based on the linear neuron
    - If the output is correct, do nothing
    - If the model incorrectly outputs 0, add the feature vector of i to the weight vector
    - If the model incorrectly outputs 1, minus the feature vector of i to the weight vector

    [(ground-truth label – predicted label) * input feature vector x]
3. Repeat step 2 until some convergence criteria is met

Intuitive explanations why the algorithm convergences

- Each training case can be seen as a hyperplane. To get all training cases right we need to find a weight vector on the right side of all the planes.
  - There may not be any such vector!
- If there are any weight vectors that get the right answer for all cases, they lie in a hyper-cone with its apex at the origin.
  - Because the average of two good weight vectors is a good weight vector.



x’ = [**x**; 1]

Intuitive explanations why the algorithm convergences

- Each time the perceptron makes a mistake, the current weight vector moves to decrease its squared distance from every weight vector in the “generously feasible” region.
- The squared distance decreases by at least the squared length of the input vector.
- So after a finite number of mistakes, the weight vector must lie in the feasible region if this region exists.

The New York Times

Electronic 'Brain' Teaches Itself

July 13, 1958

The New York Times

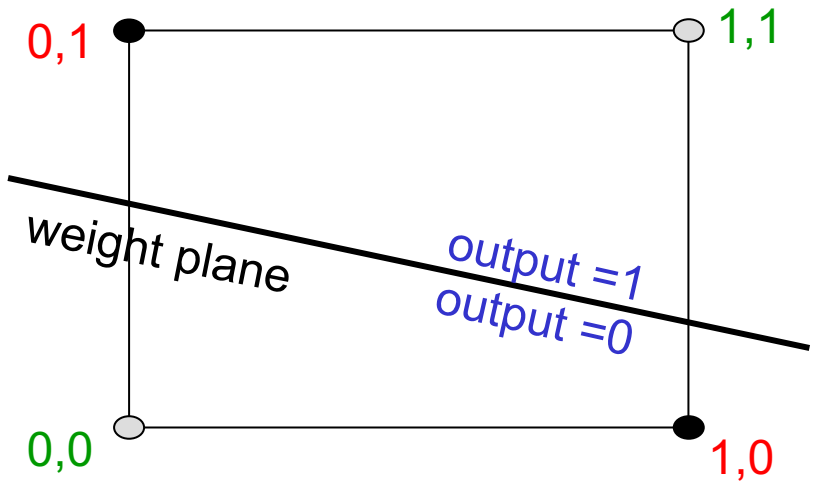
The Navy last week demonstrated the embryo of an electronic computer named the Perceptron which, when completed in about a year, is expected to be the first non-living mechanism able to "perceive, recognize and identify its surroundings without human training or control." VIEW FULL ARTICLE IN TIMESMACHINE »

Perceptron can’t even solve XOR!

Input		Output
x_1	x_2	
0	0	0
0	1	1
1	0	1
1	1	0

A geometric view of what binary threshold neurons cannot do

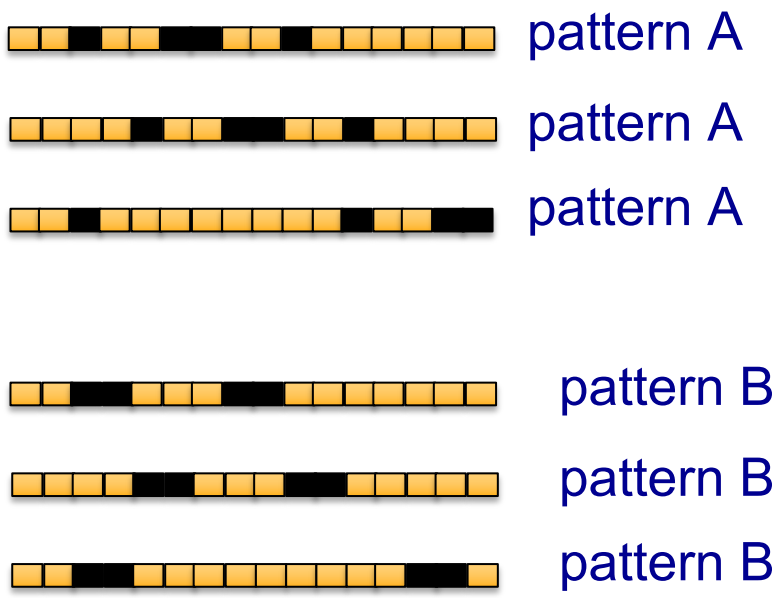
- Imagine “data-space” in which the axes correspond to components of an input vector.
  - Each input vector is a point in this space.
  - A weight vector defines a plane in data-space.
  - The weight plane is perpendicular to the weight vector and misses the origin by a distance equal to the threshold.



The positive and negative cases cannot be separated by a plane

Discriminating simple patterns under translation with wrap-around

- Suppose we just use binary pixels as the features.
- Can a binary threshold unit discriminate between different patterns that have the same number of on pixels?
    - The patterns can translate with wrap-around!



The answer is No.

- For pattern A, use training cases in all possible translations.
  - Each pixel will be activated by 4 different translations of pattern A.
  - So the total input received by the decision unit over all these patterns will be four times the sum of all the weights.
- For pattern B, use training cases in all possible translations.
  - Each pixel will be activated by 4 different translations of pattern B.
  - So the total input received by the decision unit over all these patterns will be four times the sum of all the weights.
- But to discriminate correctly, every single case of pattern A must provide more input to the decision unit than every single case of pattern B.
  - This is impossible if the sums over cases are the same.

Why this result is devastating for Perceptron (Hinton)

- The whole point of pattern recognition is to recognize patterns despite transformations like translation.

- Minsky and Papert's "Group Invariance Theorem" says that the part of a Perceptron that learns cannot learn to do this if the transformations form a group.
    - Translations with wrap-around form a group.

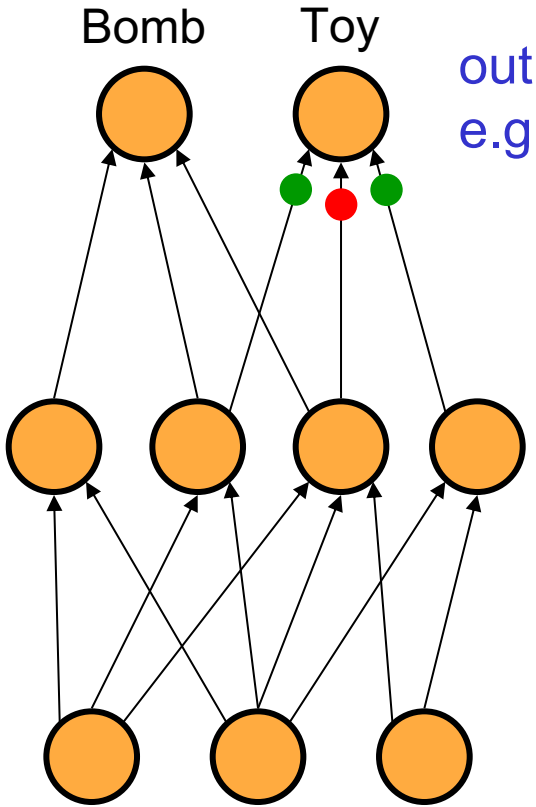
- To deal with such transformations, a Perceptron needs to use multiple feature units to recognize transformations of informative sub-patterns.
    - So the tricky part of pattern recognition must be solved by the hand-coded feature detectors, not the learning procedure.

What should we do? Handcraft Features (~1960s).

$$\phi_1(\mathbf{x}) = x_1$$
$$\phi_2(\mathbf{x}) = x_2$$
$$\phi_3(\mathbf{x}) = x_1 x_2$$

In this representation, our training set becomes

$\phi_1(\mathbf{x})$	$\phi_2(\mathbf{x})$	$\phi_3(\mathbf{x})$	t
0	*0*	*0*	*0*
0	*1*	*0*	*1*
1	*0*	*0*	*1*
1	*1*	*1*	*0*



output units
e.g. class labels

non-adaptive
hand-coded
features

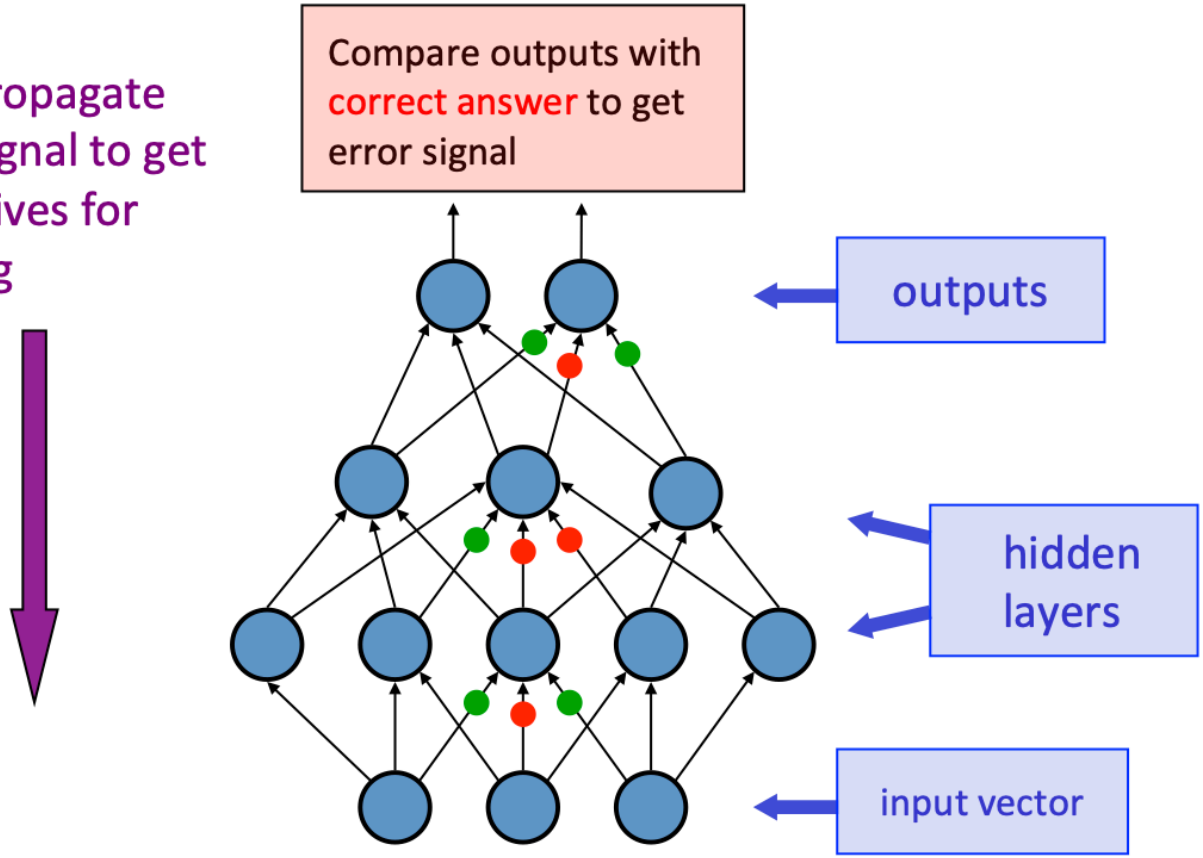
input units
e.g. pixels

Learning with Hidden Units (Hinton)

- Networks without hidden units are very limited in the input-output mappings they can learn to model.
    - More layers of linear units do not help. Its still linear.
    - Fixed output non-linearities are not enough.
- We need multiple layers of adaptive, non-linear hidden units. But how can we train such nets?
    - We need an efficient way of adapting all the weights, not just the last layer. This is hard.
    - Learning the weights going into hidden units is equivalent to learning features.
    - This is difficult because nobody is telling us directly what the hidden units should do.

Second generation neural networks (~1985, multilayer perceptron with backpropagation)

Back-propagate error signal to get derivatives for learning



Rumelhart, Hinton, and Williams, Nature 1986

Benefits of MLP with Backpropagation

- Backpropagation allows neural networks to design their own features and multiple layers of features adaptively
- No domain knowledge is needed. You just need to show the computer some training examples.
- Backpropagation is an efficient algorithm for computing how weight vectors should be updated to effect the output error
    - Instead of naively updating one weight at a time, BP computes the gradient of the weights in parallel and updates all weights through an efficient forward pass and backward pass

A Linear Neuron in Matrix-Vector Form

weight vector

$$y = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}$$

neuron's estimate of the desired output

input vector

The Delta Rule For a Linear Neuron

- Define the error as the squared residuals summed over all training cases:

$$E = \frac{1}{2} \sum_{n \in training} (t^n - y^n)^2$$

- Now differentiate to get error derivatives for weights

$$\frac{\partial E}{\partial w_i} = \frac{1}{2} \sum_n \frac{\partial y^n}{\partial w_i} \frac{dE^n}{dy^n}$$

$$= -\sum_n x_i^n (t^n - y^n)$$

- The batch delta rule changes the weights in proportion to their error derivatives summed over all training cases

$$\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i} = \sum_n \varepsilon \, x_i^n (t^n - y^n)$$

The derivatives of a logistic neuron

- The derivatives of the logit, z, with respect to the inputs and the weights are very simple:

$$z = b + \sum_i x_i w_i \qquad y = \frac{1}{1 + e^{-z}}$$

$$\frac{\partial z}{\partial w_i} = x_i \qquad \frac{\partial z}{\partial x_i} = w_i \qquad \frac{dy}{dz} = y\,(1 - y)$$

The derivatives of a logistic neuron

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

$$\frac{dy}{dz} = \frac{-1(-e^{-z})}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) = y(1-y)$$

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

Using the chain rule to get the derivatives needed for learning the weights of a logistic unit

- To learn the weights we need the derivative of the output with respect to each weight:

$$\frac{\partial y}{\partial w_i} = \frac{\partial z}{\partial w_i}\frac{dy}{dz} = x_i\, y\,(1-y)$$

$$\frac{\partial E}{\partial w_i} = \sum_n \frac{\partial y^n}{\partial w_i}\frac{\partial E}{\partial y^n} = -\sum_n x_i^n\, y^n\,(1-y^n)\,(t^n - y^n)$$

delta-rule

extra term = slope of logistic

The idea behind backpropagation (Hinton)

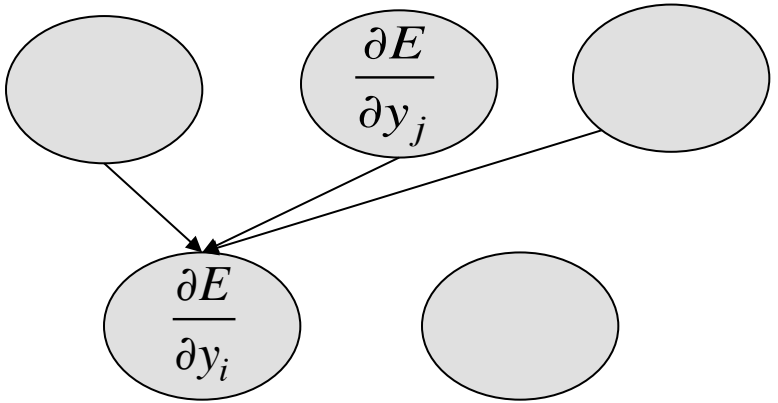
- We don't know what the hidden units ought to do, but we can compute how fast the error changes as we change a hidden activity.
  - Instead of using desired activities to train the hidden units, use error derivatives w.r.t. hidden activities.
  - Each hidden activity can affect many output units and can therefore have many separate effects on the error. These effects must be combined.
- We can compute error derivatives for all the hidden units efficiently at the same time.
  - Once we have the error derivatives for the hidden activities, its easy to get the error derivatives for the weights going into a hidden unit.

Sketch of the backpropagation algorithm on a single case (Hinton)

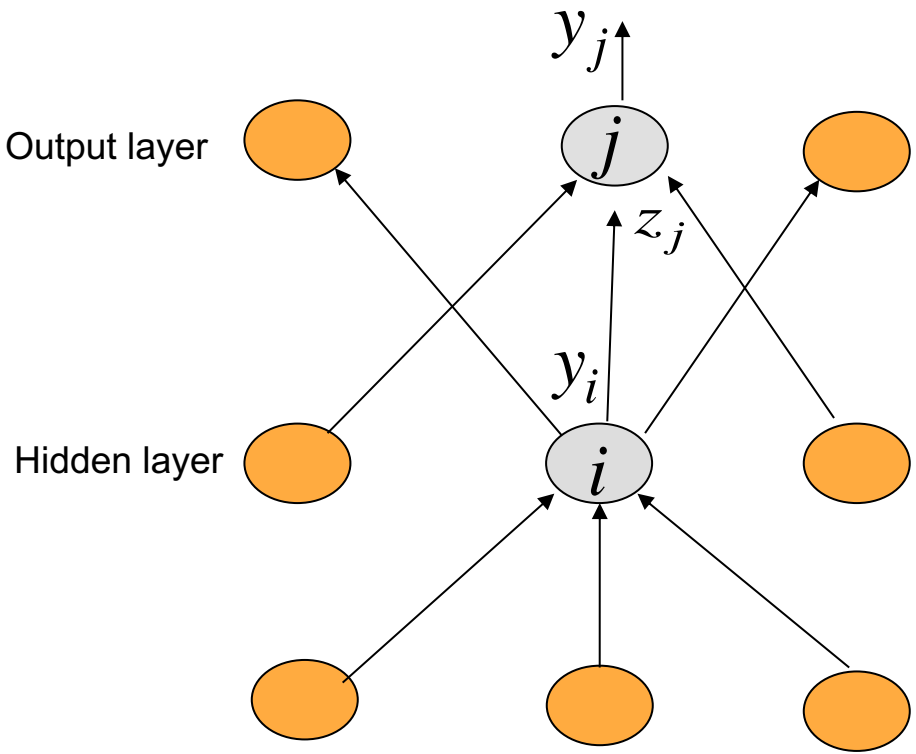
- First convert the discrepancy between each output and its target value into an error derivative.
- Then compute error derivatives in each hidden layer from error derivatives in the layer above.
- Then use error derivatives *w.r.t.* activities to get error derivatives *w.r.t.* the incoming weights.

$$E = \frac{1}{2} \sum_{j \in output} (t_j - y_j)^2$$

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



Backpropagating dE/dy (Hinton)

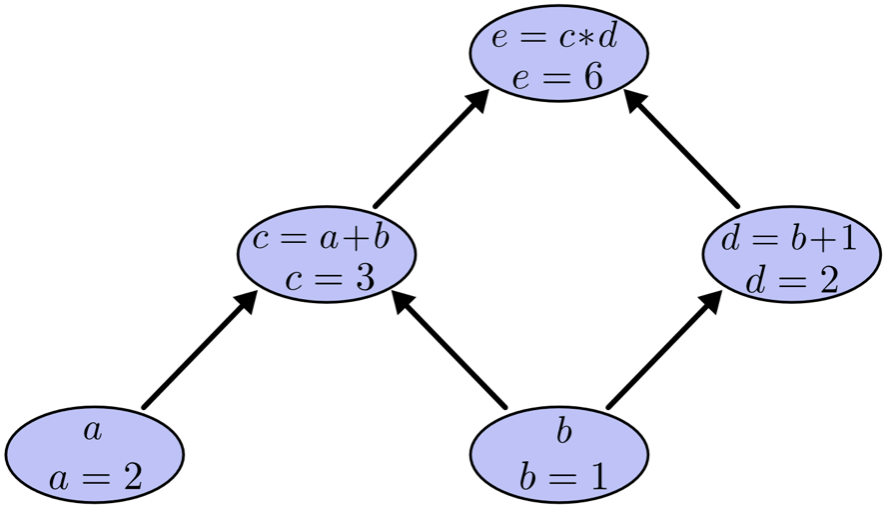


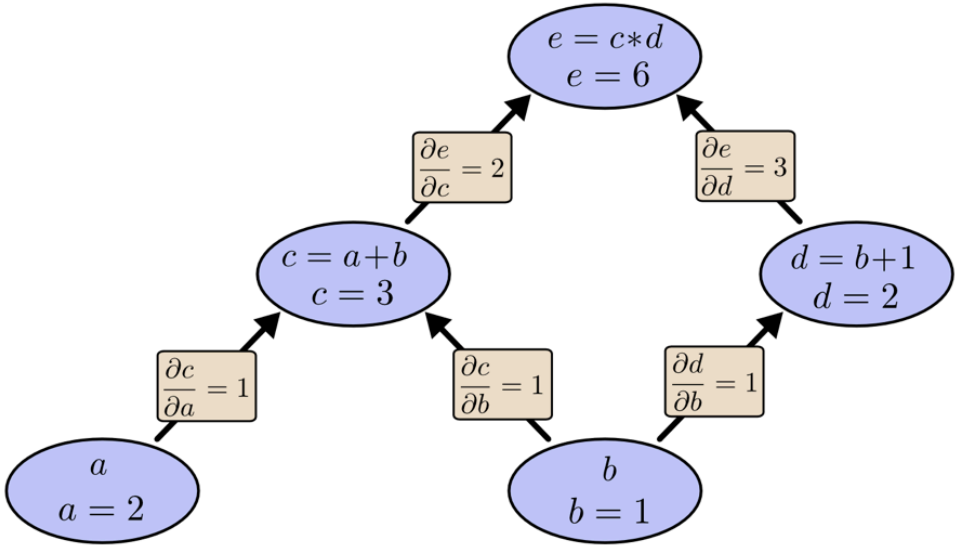
$$\frac{\partial E}{\partial z_j} = \frac{dy_j}{dz_j}\frac{\partial E}{\partial y_j} = y_j(1-y_j)\frac{\partial E}{\partial y_j}$$

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{dz_j}{dy_i}\frac{\partial E}{\partial z_j} = \sum_j w_{ij}\frac{\partial E}{\partial z_j}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}}\frac{\partial E}{\partial z_j} = y_i\frac{\partial E}{\partial z_j}$$

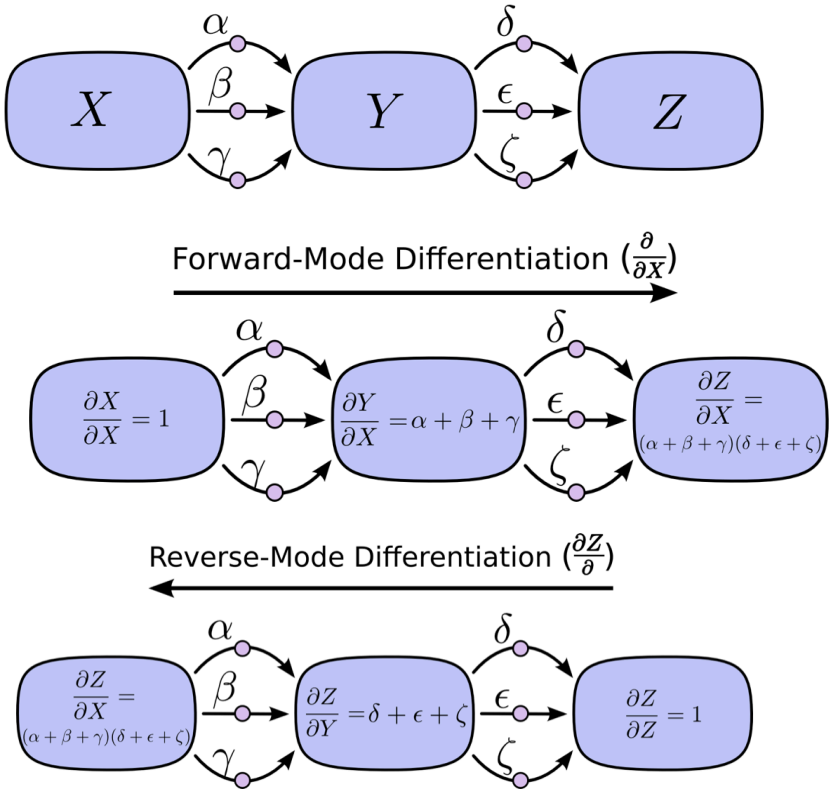
BP in a Computational Graph





http://colah.github.io/posts/2015-08-Backprop/

BP in a Computational Graph



http://colah.github.io/posts/2015-08-Backprop/

Old Criticisms about Backpropagation

- It requires labeled training data.
  - Almost all data is unlabeled.
- The brain needs to fit about 10^14 connection weights in only about 10^9 seconds.
  - Unless the weights are highly redundant, labels cannot possibly provide enough information.
- The learning time does not scale well ?
  - It is very slow in networks with multiple hidden layers.
- The neurons need to send two different types of signal ?
  - Forward pass: signal = activity = y
  - Backward pass: signal = dE/dy

Regularized Deep Autoencoder in 2005

Chapter 2

Regularized Autoencoder

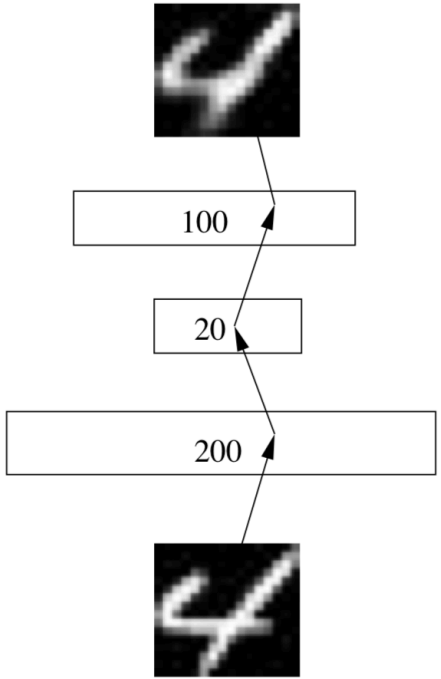
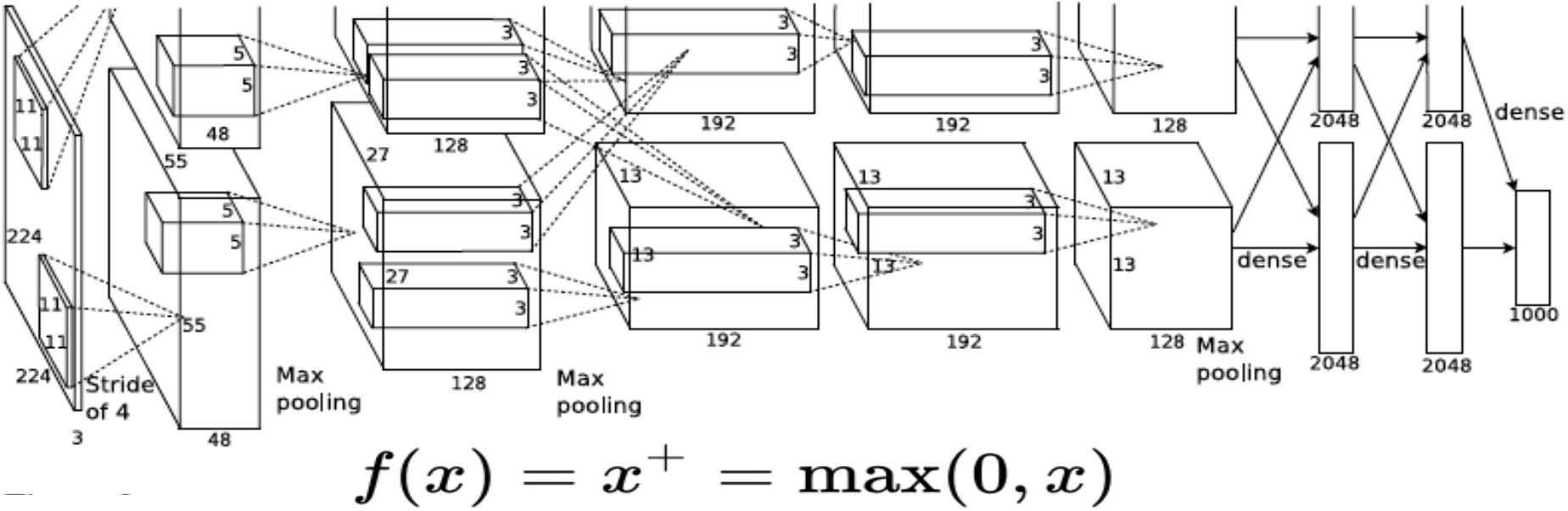


Figure 4.3: The configuration of the autoencoder and RAN for digit data.

Modern Deep Convolutional Neural Networks (2012-now, big data + **deep CNN + rectified linear + data augmentation** +dropout + GPUs)

Deep learning on GPU (Krizhevsky et al. NIPS12)

Revolutionary success on the ImageNet Challenge



$$f(x) = x^+ = \max(0, x)$$

60 million parameters and 650,000 neurons

What makes deep learning revive?

- Deep learning is a modern name for deep neural networks
- Why deep neural networks revive
  - Good model: has the intrinsic advantage of learning hierarchical distributed feature representations
  - Powerful computers and big data enable us to build models with huge capacity
  - Data augmentation (using heuristics to create more data to fill the input data distribution space)
  - Parameter learning without vanishing gradient or saturated activation functions (rectified linear hidden units) and regularizations (dropout and batch normalization)
  - Good open-source tools: cuda-convnet2, Torch, Caffe, TensorFlow, and PyTorch

Summary

- The ongoing AI revolution is driven by deep learning. Hidden units and backpropagation are great inventions in neural networks.

- A deep model + big data + powerful computers -> Success

- A quote about machine learning by Prof. Pedro Domingos from University of Washington, “Learning is more like farming, which lets nature do most of the work. Farmers combine seeds with nutrients to grow crops. Learners combine knowledge with data to grow programs.” This is also true for deep learning.

- Deep learners should combine their knowledge with large-scale data to grow programs, encode essential knowledge into network structures, and let backpropagation and stochastic gradient descent do the heavy lifting.

The End

Next lecture:

Deep Learning II: Supervised Deep Learning with Convolutional Neural Networks