CS6421: Deep Neural Networks

Gregory Provan

Spring 2020

Lecture 1: Introduction

Based on notes from John Canny, Ismini Lourentzou,
Stanford courses CS224NLP/CS231

Overview

- Course Overview
  - Focus
  - Grading
  - Topics to be covered
- Neural Networks: Brief Survey
- Introduction to Deep Learning

Focus of Course

- Introduce key aspects of deep learning
  - *Scientific basis*
  - *Methodology* to implement deep networks
    - Use Keras/TensorFlow for implementations
- NOT tool-focused
  - Tools change over time, but the science is the key to using deep learning
  - If you want to learn about tools, take a different course

Course Schedule

	Week	Date	Number	Main Topic	Details
BACKGROUND	1	**15/01/2020**	1	Overview	Introduction to deep learning
		17/01/2020	2	Classification and Learning	Using deep networks for classification and learning tasks
	2	22/01/2020	3	Modularity	Architectures for Deep networks
		24/01/2020	4	Mathematical Background	Matrices and differential calculus
BASIC INFERENCE	3	29/01/2020	5	DL Inference	Designing a Learning problem
		31/01/2020	6	Inference: classification	Computation graph - Theory - Tensor Flow
	4	05/02/2020	7	TensorFlow: practical aspects	TensorFlow implementation details
		07/02/2020	8	Deep Learning: Training	Backpropagation; Optimisation
PRACTICAL ASPECTS	5	12/02/2020	9	Practical Aspects	Network-Initialisation: Architectures; Activation and Loss functions
		14/02/2020	10	Practical Aspects	Weight initialisation; Data pre-processing
	6	19/02/2020	11	Practical Aspects	Data augmentation; transfer learning; Deep-Network practical issues; CPU/GPU practical issues
		21/02/2020	12	Practical Aspects: Training Deep Networks	Optimization gradient flow batch optimization
	7			**MIDTERM**	
CNN	8	04/03/2020	13	Convolution Neural Networks (CNNs)	Applications of CNNs
		06/03/2020	14	Convolution Neural Networks (CNNs)	Applications of CNNs
	9	11/03/2020	15		
TEMPORAL NETWORKS		13/03/2020	16	Temporal Deep Networks	RNN
	10	18/03/2020	17	Temporal Deep Networks	LSTM
		20/03/2020	18	Temporal Deep Networks	RNN/LSTM applications
REINFORCEMENT LEARNING	11	25/03/2020	19	Deep Reinforcement Learning	Deep Reinforcement Learning introduction; critic-based methods
		27/03/2020	20	Deep Reinforcement Learning	Deep Reinforcement Learning: actor-based methods
	12	01/04/2020	21	Deep-RL-applications	
		03/04/2020	22	Deep Probabilistic Networks	Restricted Boltzmann Machines

Technical Topics Covered

- Machine Learning basics
- Introduction to Deep Learning
  - what is Deep Learning
  - why is it useful
- Main components/hyper-parameters:
  - activation functions
  - optimizers, cost functions and training
  - regularization methods
  - tuning
  - classification vs. regression tasks
- Methodology of deep learning
  - How to train deep networks on real-world data
- DNN basic architectures
  - convolutional
  - recurrent
  - attention mechanism

Some material from CS224 NLP with DL course at Stanford

Pre-Requisites

- Mathematics
  - Multi-variable calculus
  - Linear algebra
    - Matrix and tensor algebras
  - Basic probability theory
- Programming
  - Python (intermediate level)

Lecture: Jan. 24th

Grading

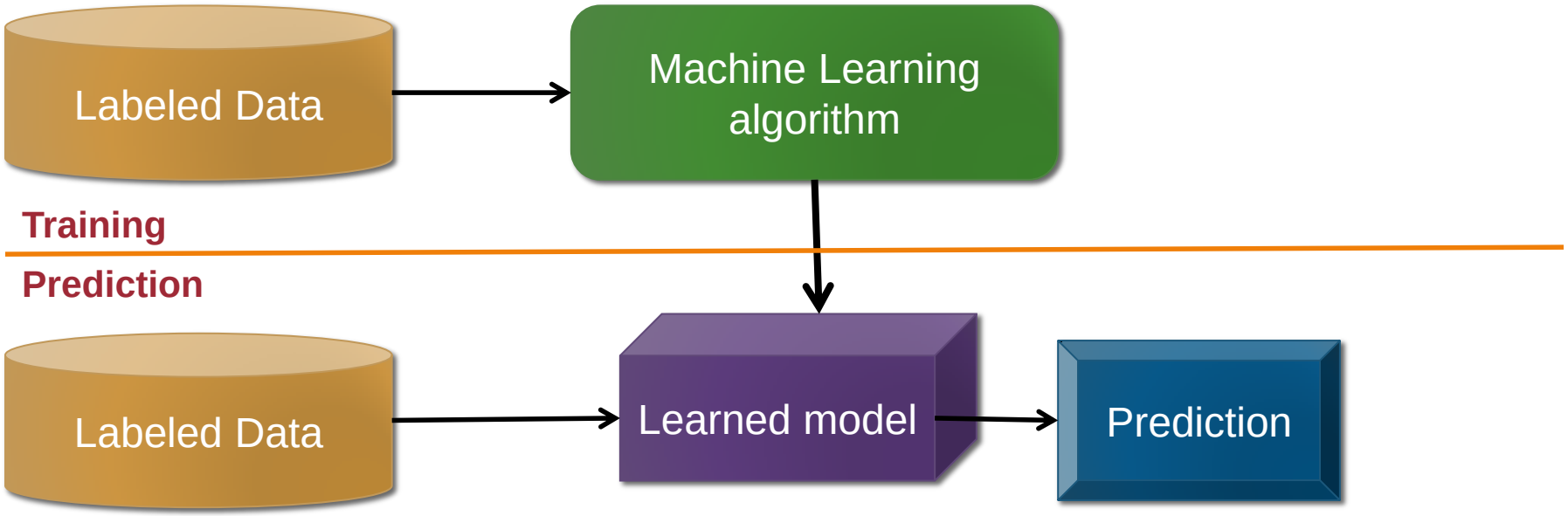
- Mid-term: 35%
- Final: 35%
- Programming assignments: 30%

What will the exams be like?

- Technical understanding of deep learning
  - *Key concept*: how to design a deep network to perform task X.
  - Questions will explore science of deep learning
- Practice exams will be provided
- Examples
  - What architecture is best suited to image processing?
  - What activation functions work best for temporal models?
  - Draw a computation graph to solve task Y
  - How does the differentiability of an activation function affect the performance of a deep network?

Machine Learning Basics

Machine learning is a field of computer science that gives computers the ability to **learn without being explicitly programmed**



Methods that can learn from and make predictions on data

Types of Learning

Supervised: Learning labels ***y*** with a **labeled training** set ***x***
Example: email *classification* with already labeled emails

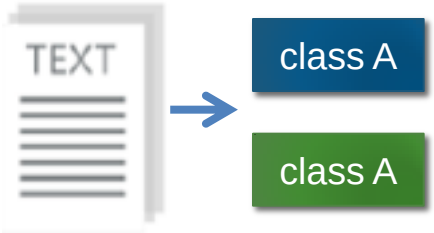
$y=f(x)$

Unsupervised: Discover **patterns** in **unlabeled** data ***x***
Example: *cluster* similar documents based on text

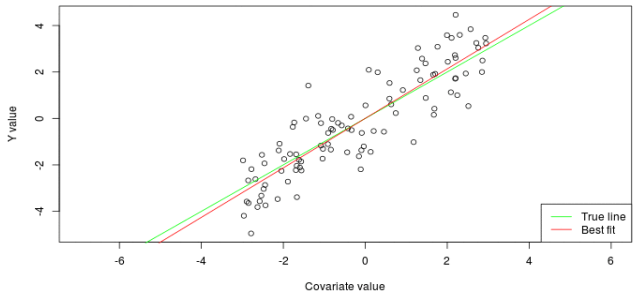
$f(x)$

Reinforcement learning: learn to **act** based on **feedback/reward *z***
Example: learn to play Go, reward: *win or lose*

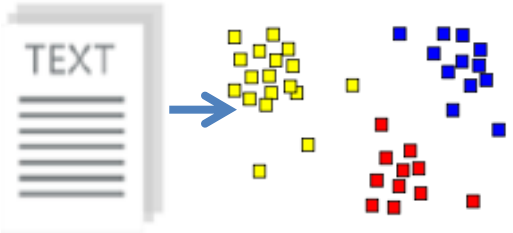
$y=f(x)$
z



Classification



Regression



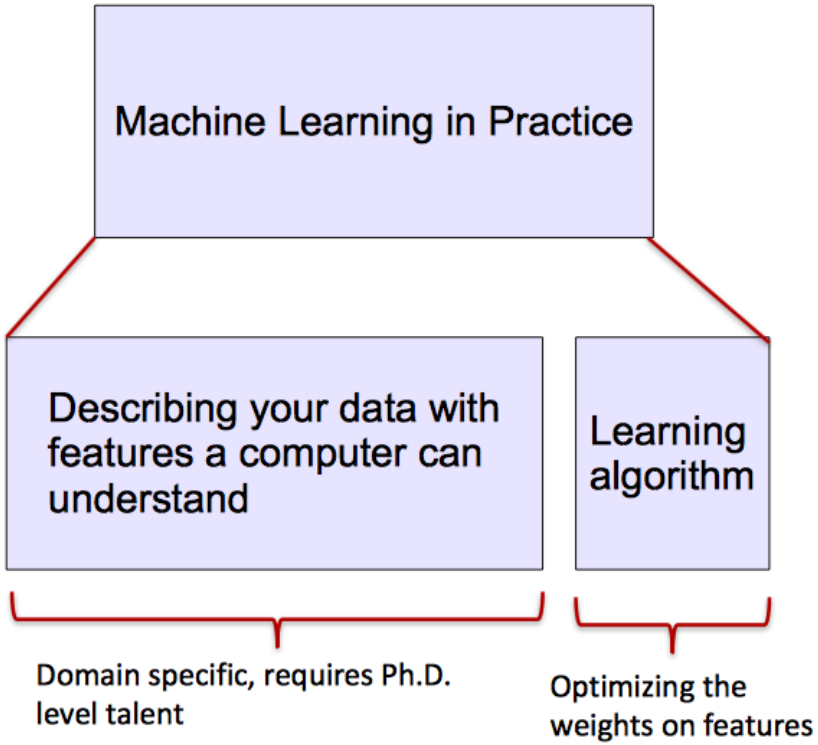
Clustering

Anomaly Detection
Sequence labeling
…

http://mbjoseph.github.io/2013/11/27/measure.html

ML vs. Deep Learning

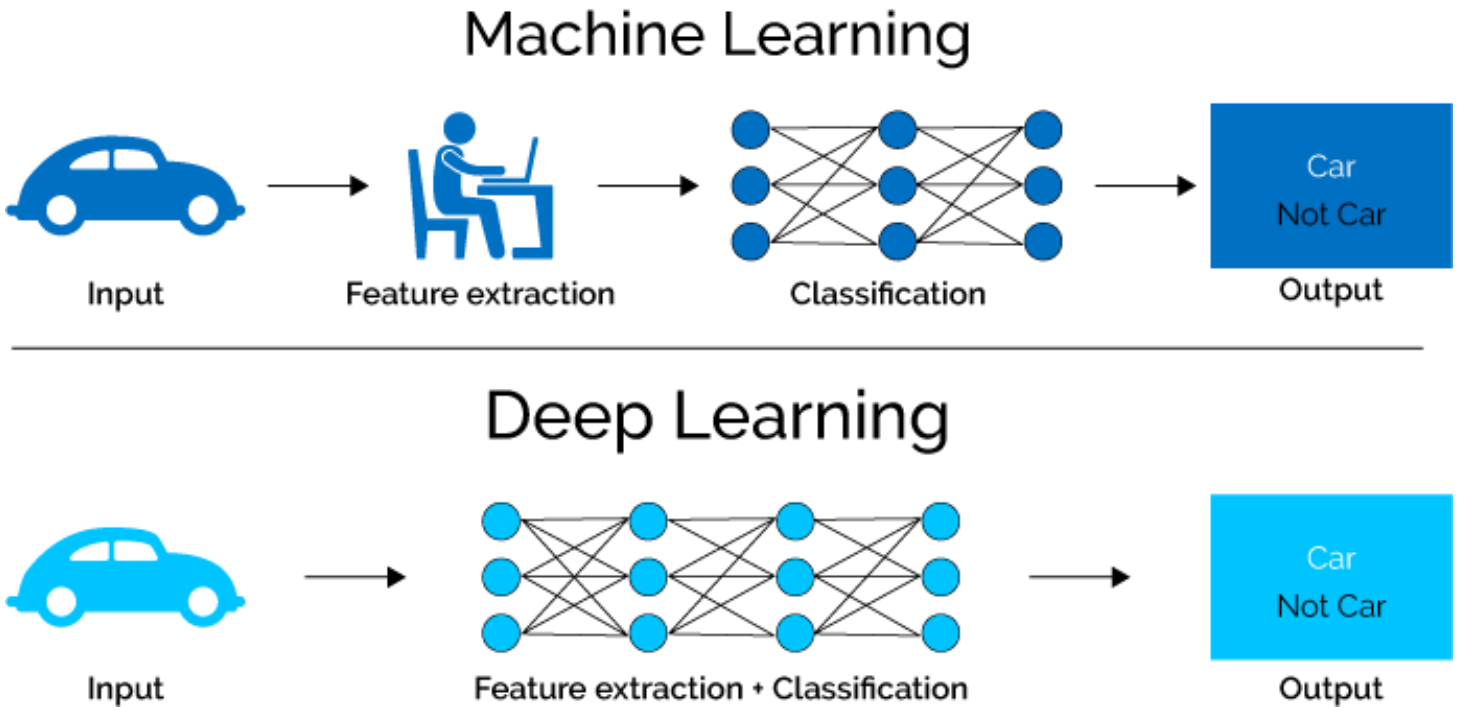
Most machine learning methods work well because of **human-designed representations** and **input features**
ML becomes just **optimizing weights** to best make a final prediction



Feature	NER
Current Word	✓
Previous Word	✓
Next Word	✓
Current Word Character n-gram	all
Current POS Tag	✓
Surrounding POS Tag Sequence	✓
Current Word Shape	✓
Surrounding Word Shape Sequence	✓
Presence of Word in Left Window	size 4
Presence of Word in Right Window	size 4

What is Deep Learning (DL) ?

A machine learning subfield of learning **representations** of data. Very effective at **learning patterns**.
Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**
If you provide the system **tons of information**, it begins to understand it and respond in useful ways.



https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

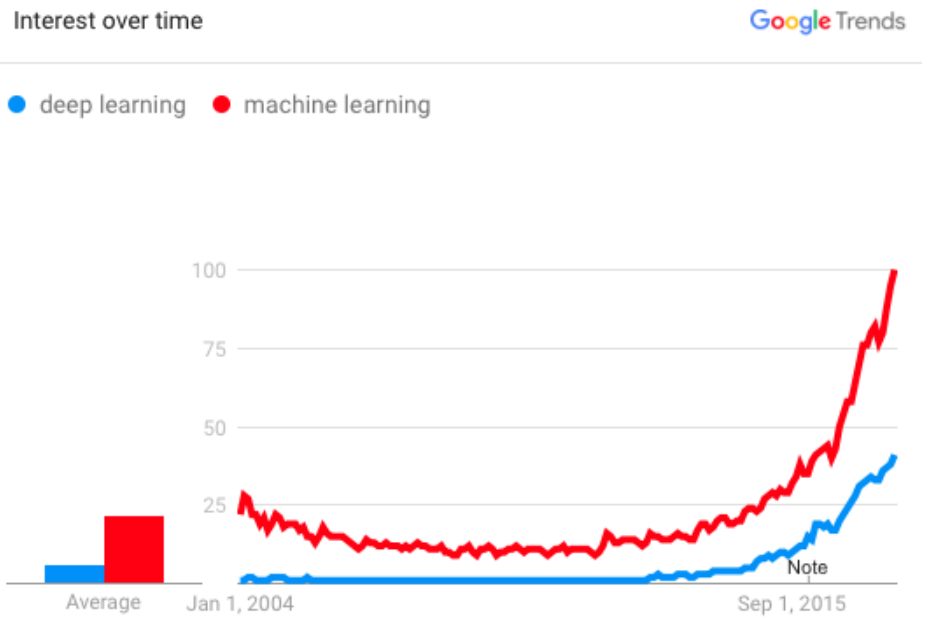
Key Thing to Understand

- Task difficulty remains unchanged
  - Learning task is NP-hard (at least)
  - Type of learning does not affect this
- ML
  - manual feature extraction/analysis
- Deep learning
  - manual architecture/hyper-parameter specification/analysis

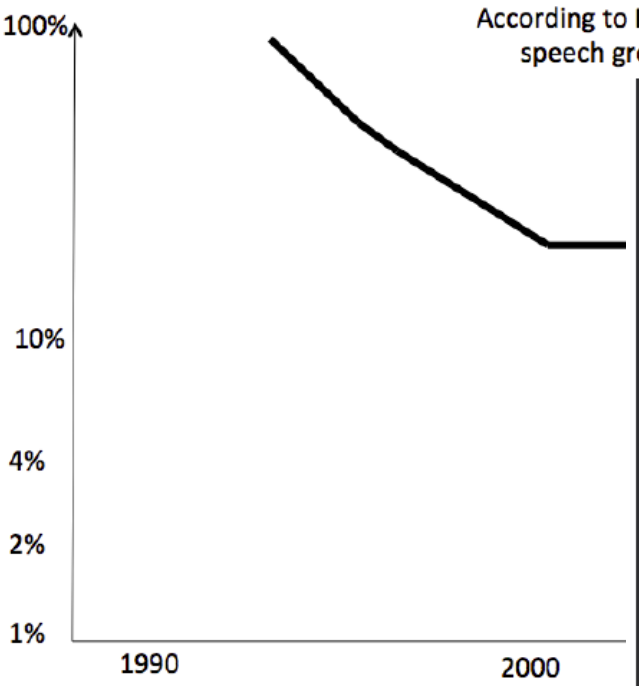
Why is DL useful?

- Manually designed features are often **over-specified**, **incomplete** and take a **long time to design** and validate
- Learned Features are **easy to adapt**, **fast** to learn
- Deep learning provides a very **flexible**, (almost?) **universal**, learnable framework for representing world, visual and linguistic information.
- Can learn both unsupervised and supervised
- Effective **end-to-end** joint system learning
- Utilize large amounts of training data

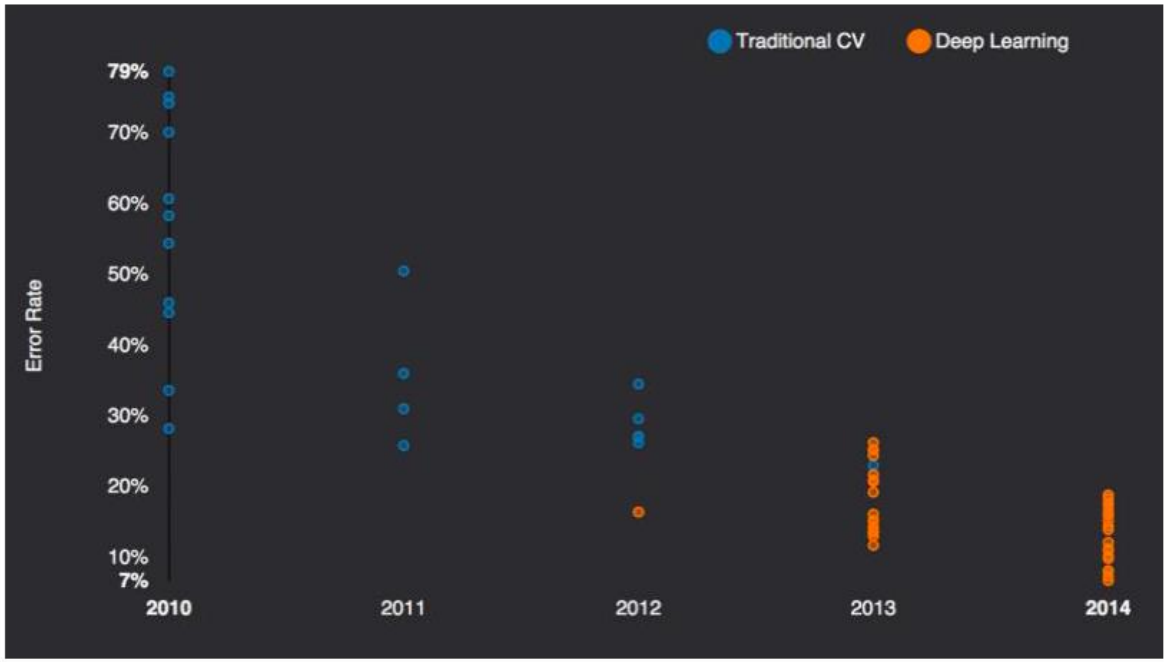
In ~2010 DL started outperforming other ML techniques
first in speech and vision, then NLP



State of the art in …



Deep Learning in Speech Recognition



ImageNet: The "computer vision World Cup"

Several big improvements in recent years in NLP

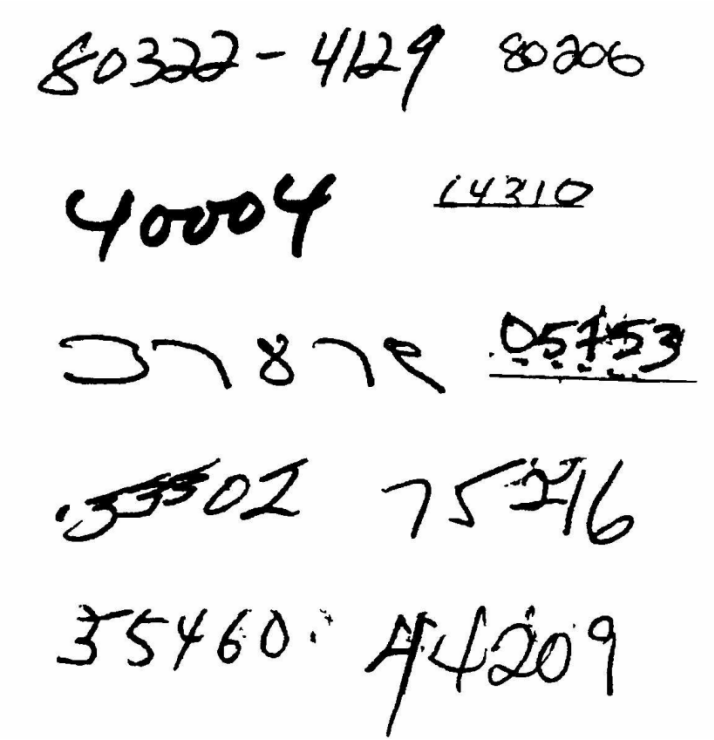
- ✓ Machine Translation
- ✓ Sentiment Analysis
- ✓ Dialogue Agents
- ✓ Question Answering
- ✓ Text Classification …

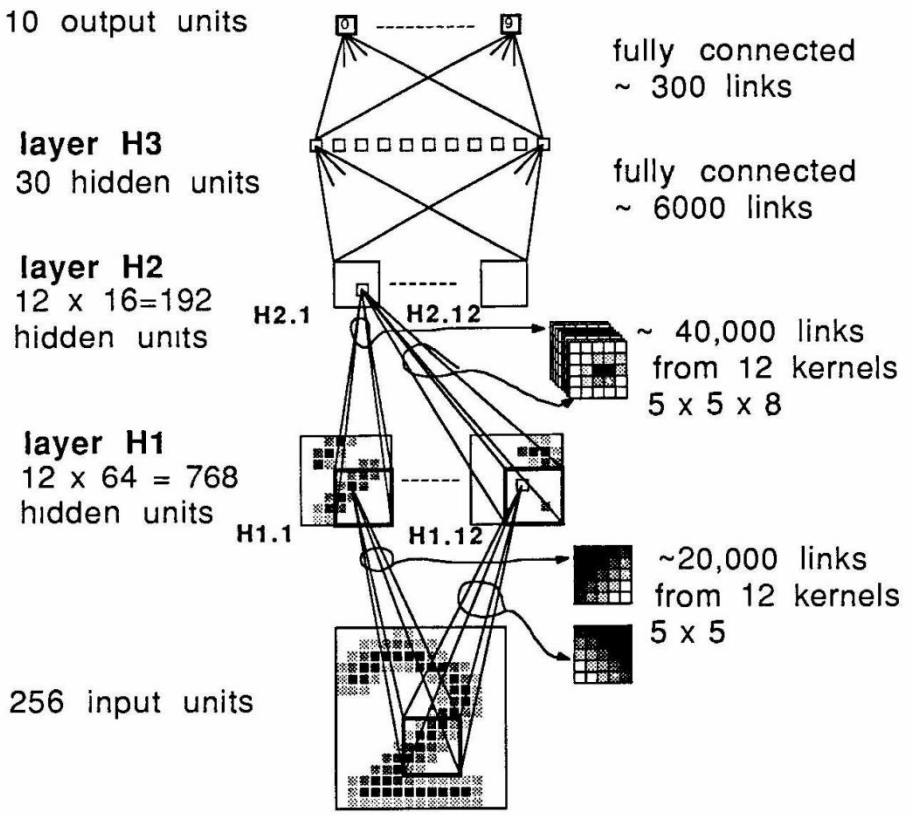
Leverage different levels of representation

- words & characters
- syntax & semantics

Milestones: Digit Recognition

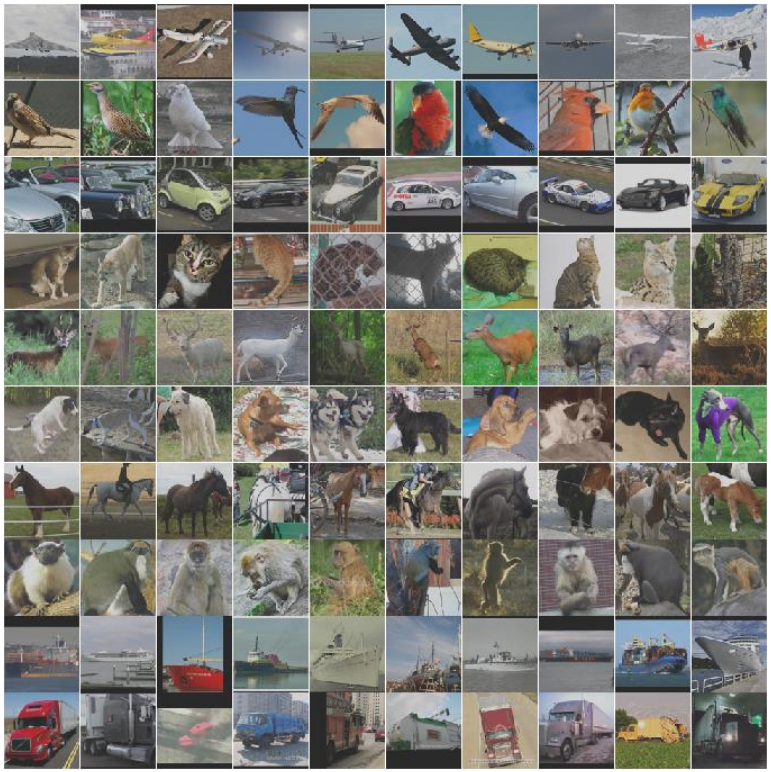
LeNet 1989: recognize zip codes, Yann Lecun, Bernhard Boser and others, ran live in US postal service

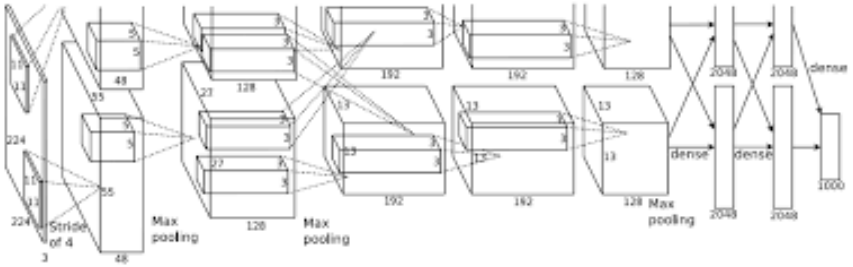


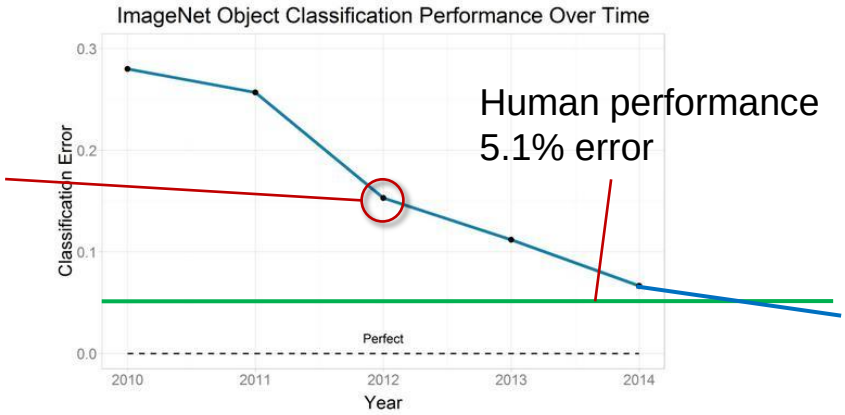


Milestones: Image Classification

Convolutional NNs: AlexNet (2012): trained on 200 GB of ImageNet Data

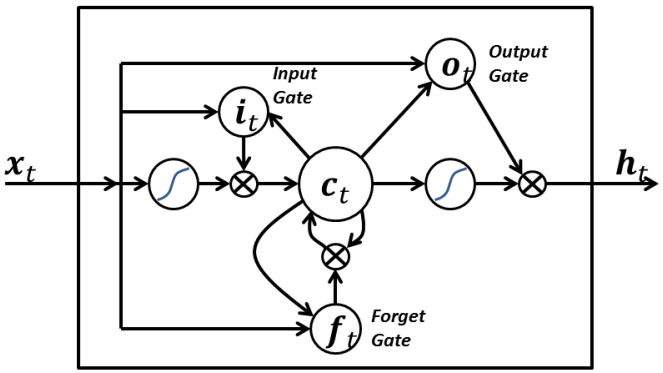


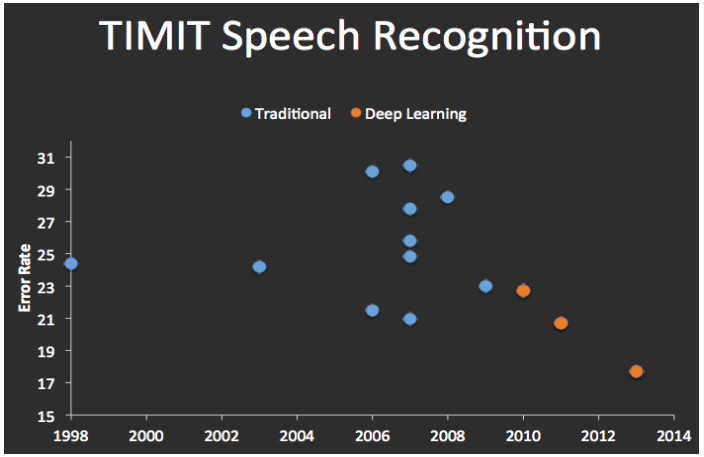


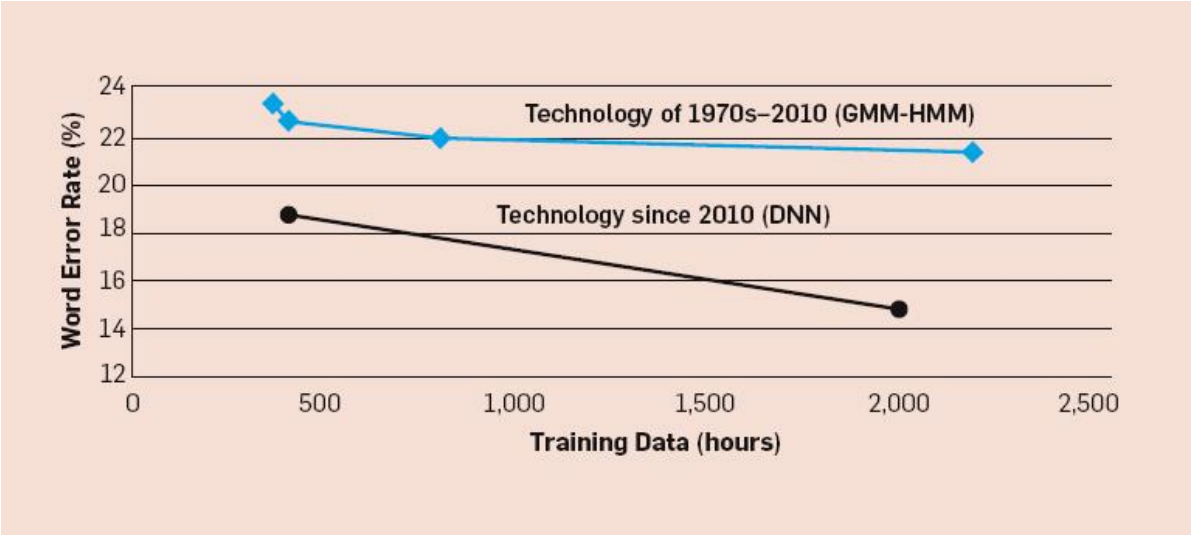


Milestones: Speech Recognition

Recurrent Nets: LSTMs (1997):

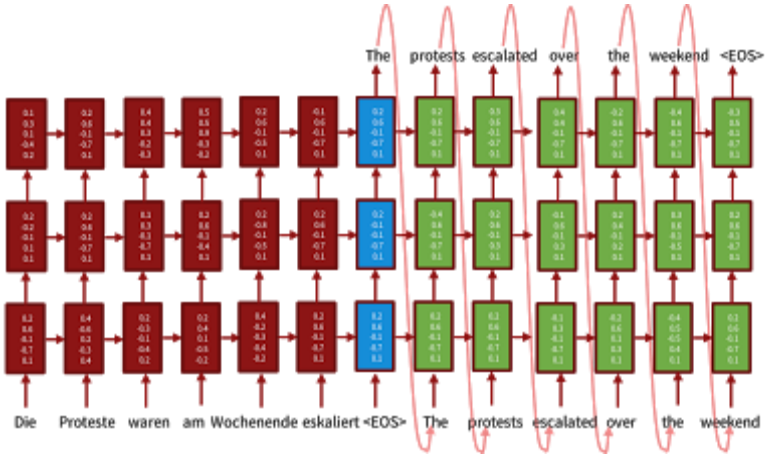


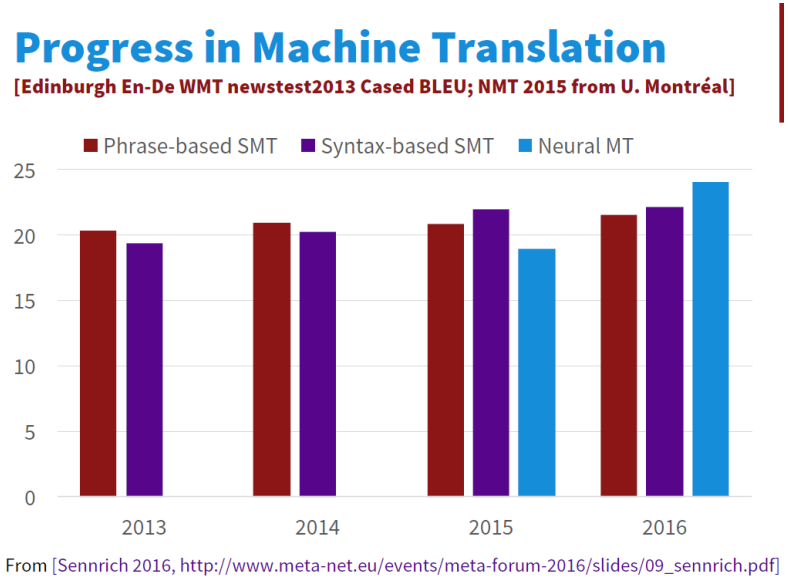




Milestones: Language Translation

Sequence-to-sequence models with LSTMs and attention:



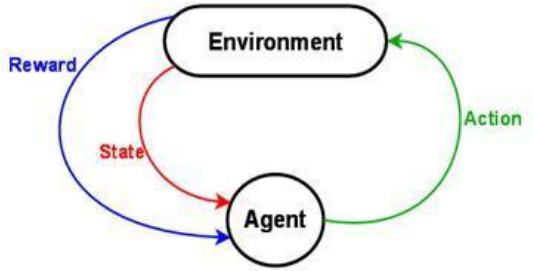


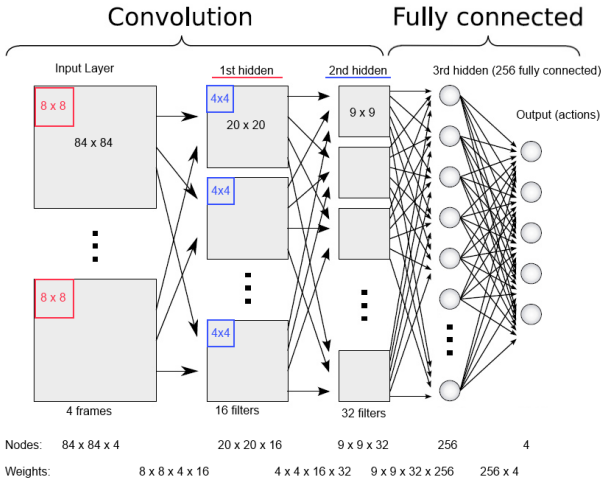
From [Sennrich 2016, http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf]

Source: Luong, Cho, Manning ACL Tutorial 2016.

Milestones: Deep Reinforcement Learning

In 2013, Deep Mind's arcade player bests human expert on six Atari Games. Acquired by Google in 2014,.





In 2016, Deep Mind's alphaGo defeats former world champion Lee Sedol

Deep Learning: Is it Hype or Hope?

Deep Learning: Is it Hype or Hope?

Yes !

Why is DL Hard?

- Creating high-performance network is a “black art”
  - Many adjustable parameters
- Lots of data needed
- Need hardware/software integration for efficiency
  - GPUs are difficult to program

ML compared to DL

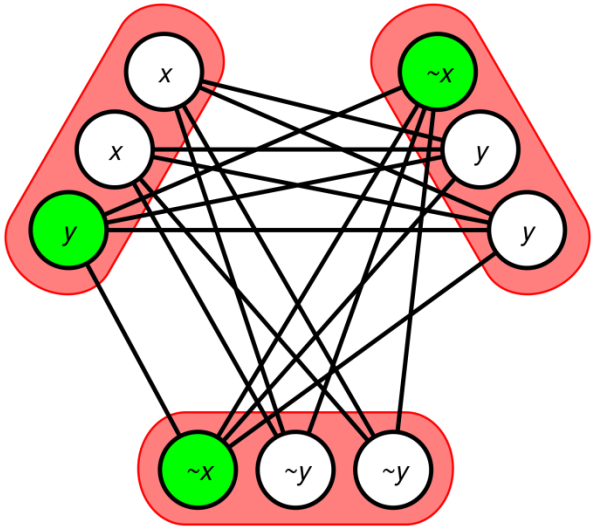
	Machine Learning	Deep Learning
Data Dependencies	Excellent performances on a small/medium dataset	Excellent performance on a big dataset
Hardware dependencies	Work on a low-end machine.	Requires powerful machine, preferably with GPU: DL performs a significant amount of matrix multiplication
Feature engineering	Need to understand the features that represent the data	No need to understand the best feature that represents the data
Execution time	From few minutes to hours	Up to weeks. Neural Network needs to compute a significant number of weights
Interpretability	Some algorithms are easy to interpret (logistic, decision tree), some are almost impossible (SVM, XGBoost)	Difficult to impossible

ML compared to DL

	Machine learning	**Deep learning**
Training dataset	Small	Large
Choose features	Yes	No
Number of algorithms	Many	Few
Training time	Short	Long

Why the success of DNNs is surprising

- From both complexity and learning theory perspectives, simple networks are very limited.
  - Can't compute parity with a small network.
  - NP-Hard to learn "simple" functions like 3SAT formulae
  - Training a DNN is NP-hard.



Why the success of DNNs is surprising

- The most successful DNN training algorithm is a version of gradient descent which will only find local optima. In other words, it's a greedy algorithm. Backprop:

$$\text{loss } L = f(g(h(y)))$$

$$dL/dy = f'(g) \times g'(h) \times h'(y)$$

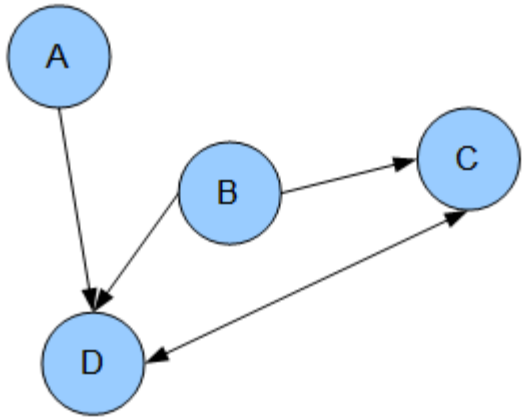
- Greedy algorithms are even more limited in what they can represent and how well they learn.

- If a problem has a greedy solution, its regarded as an "easy" problem.

Why the success of DNNs is surprising

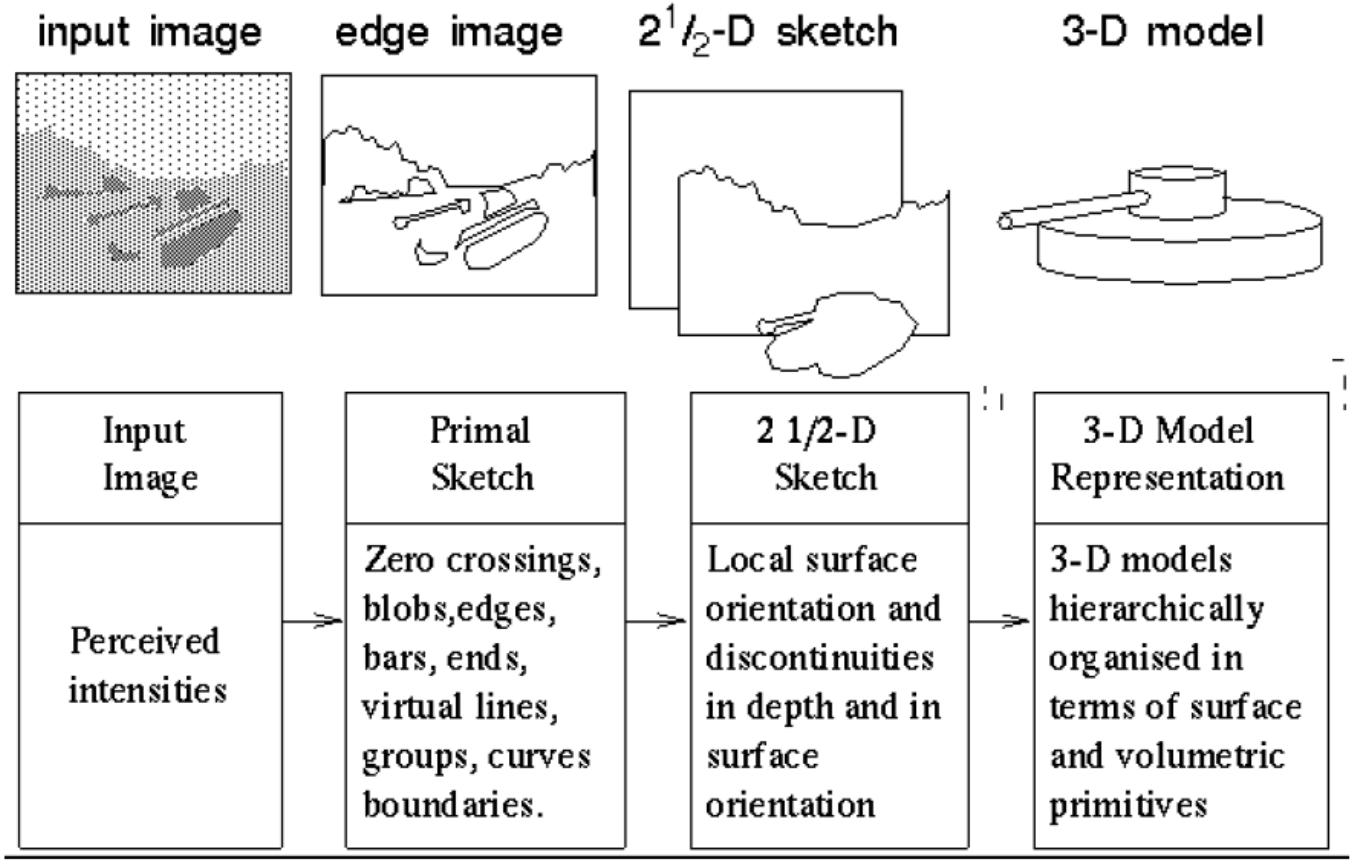
- In graphical models, values in a network represent random variables, and have a clear meaning. The network structure encodes dependency information, i.e. you can represent rich models.

- In a DNN, node activations encode nothing in particular, and the network structure only encodes (trivially) how they derive from each other.



Why the success of DNNs is ~~surprising~~ obvious

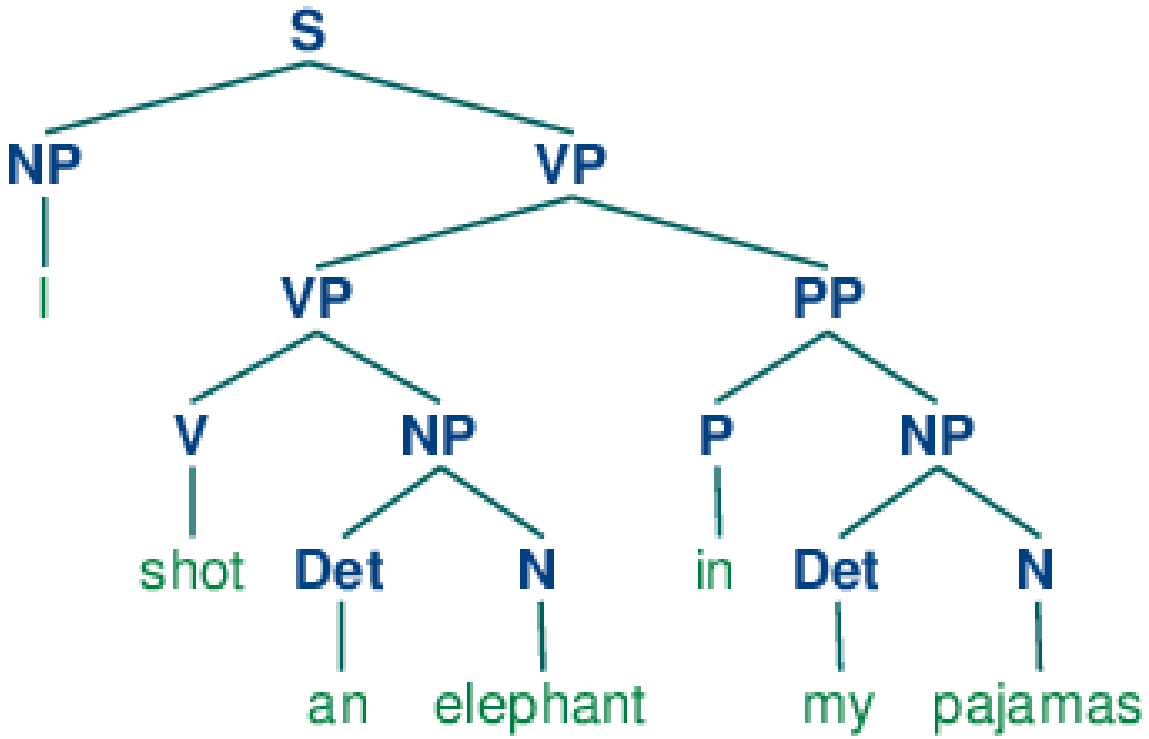
- Hierarchical representations are ubiquitous in AI. Computer vision:



Stages of Visual Representation, David Marr, 1970s

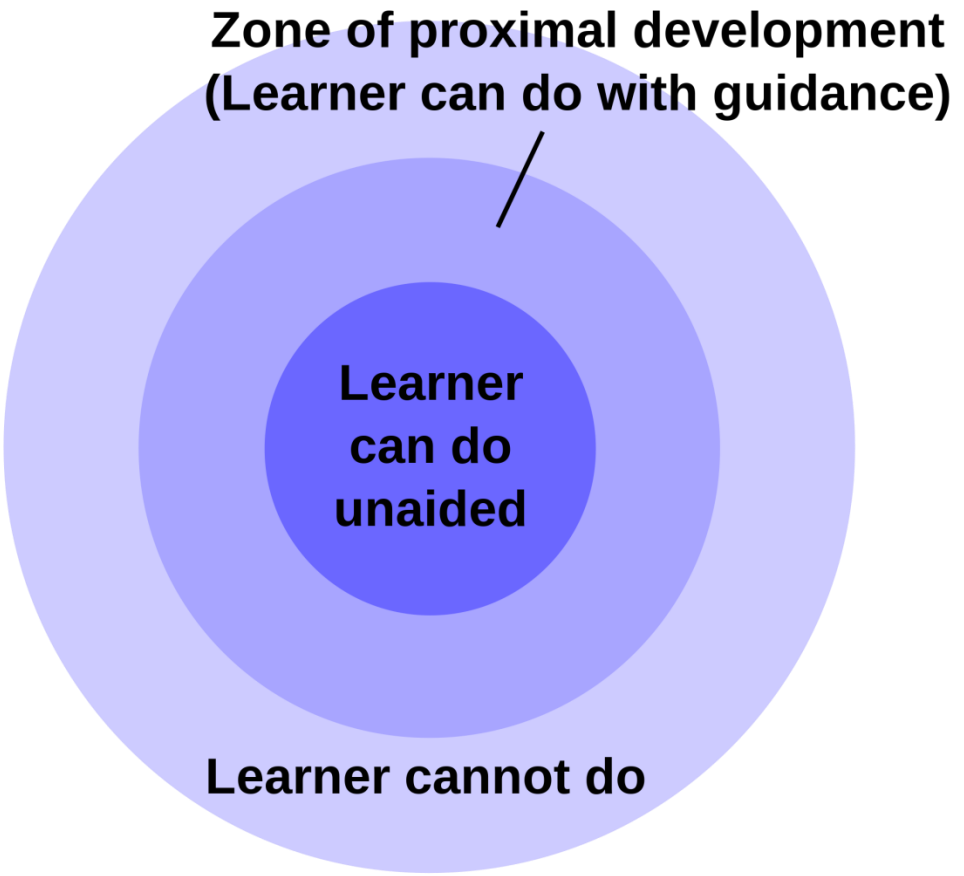
Why the success of DNNs is ~~surprising~~ obvious

- Natural language:



Why the success of DNNs is ~~surprising~~ obvious

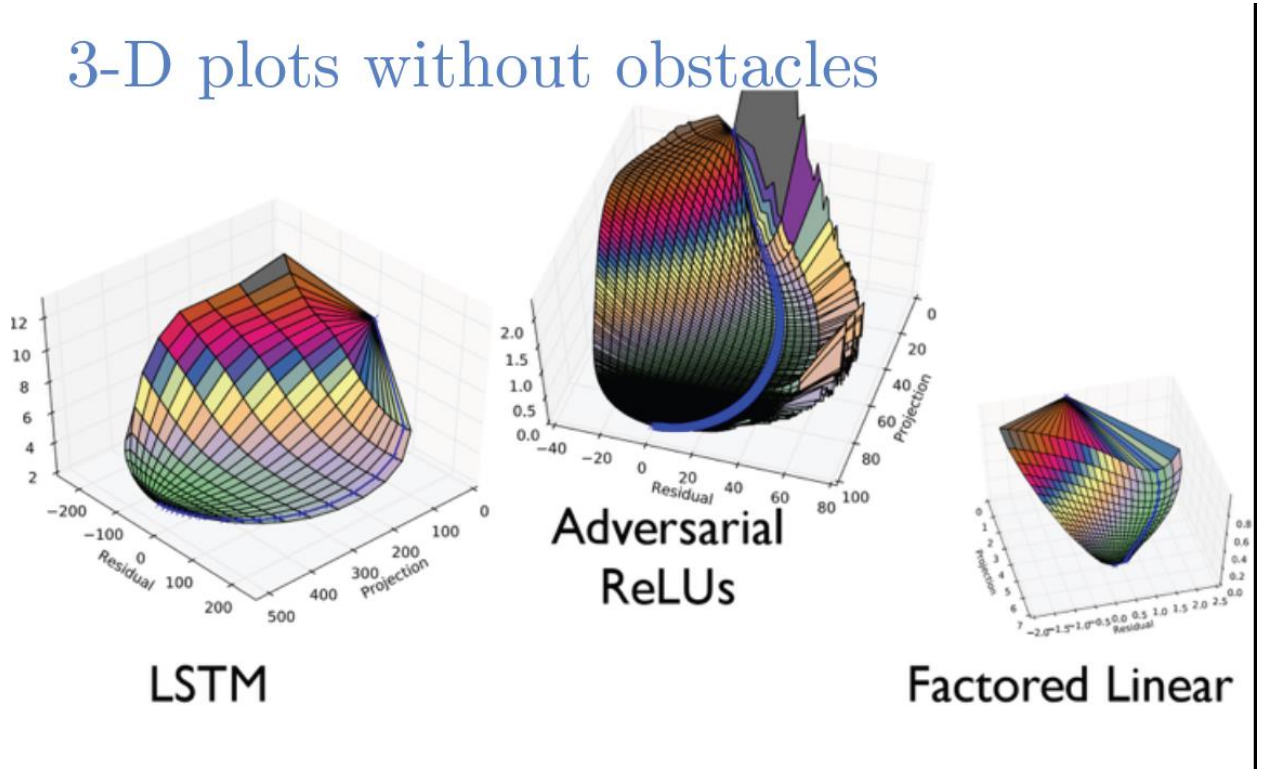
- Human Learning: is deeply layered.



Deep expertise

Why the success of DNNs is ~~surprising~~ obvious

- What about greedy optimization?
- Less obvious, but it looks like many learning problems (e.g. image classification) are actually “easy” i.e. have reliable steepest descent paths to a good model.



Neural Network: Basis of Deep Learning

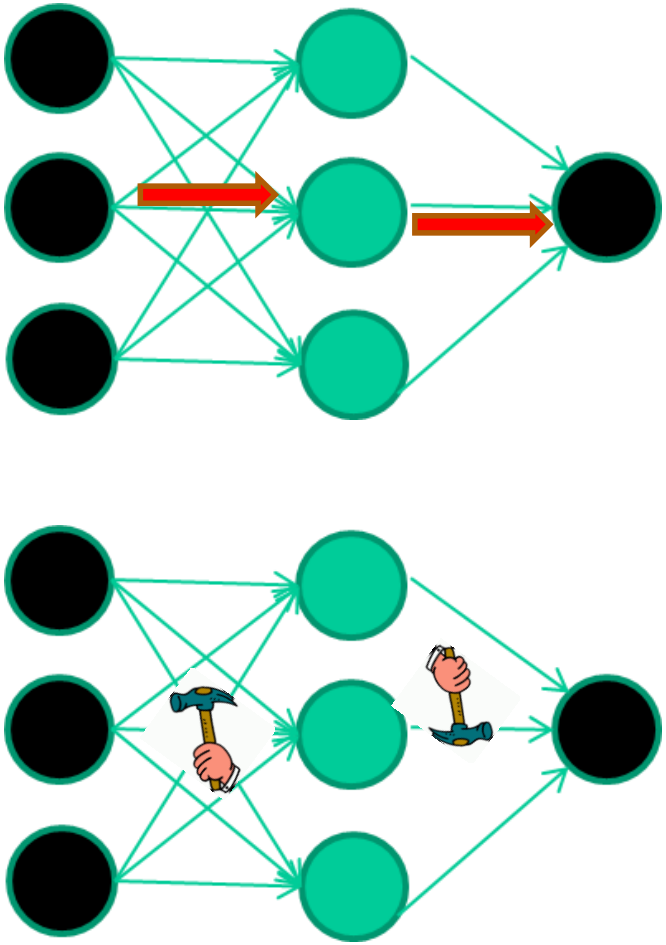
- Overview of neural network
- Functional approximation
  - NN is a universal function approximator
    - Can approximate an arbitrary non-linear function

Artificial Neural Network (NN)

- NN: mathematical model designed to solve engineering problems
  - Group of highly connected artificial neurons to realize compositions of non-linear functions
- Tasks
  - Classification
  - Discrimination
  - Estimation
- 2 types of networks
  - Feed forward Neural Networks
  - Recurrent Neural Networks

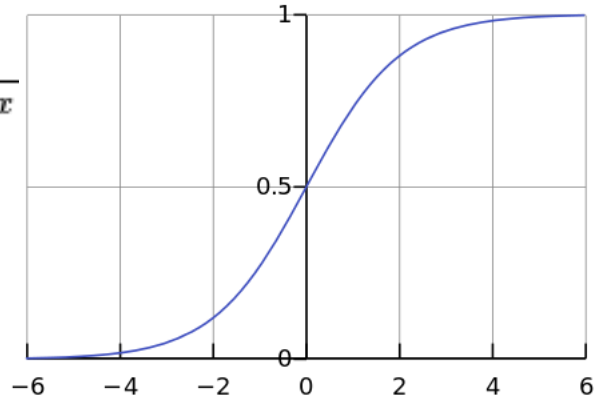
Neural Network Basics

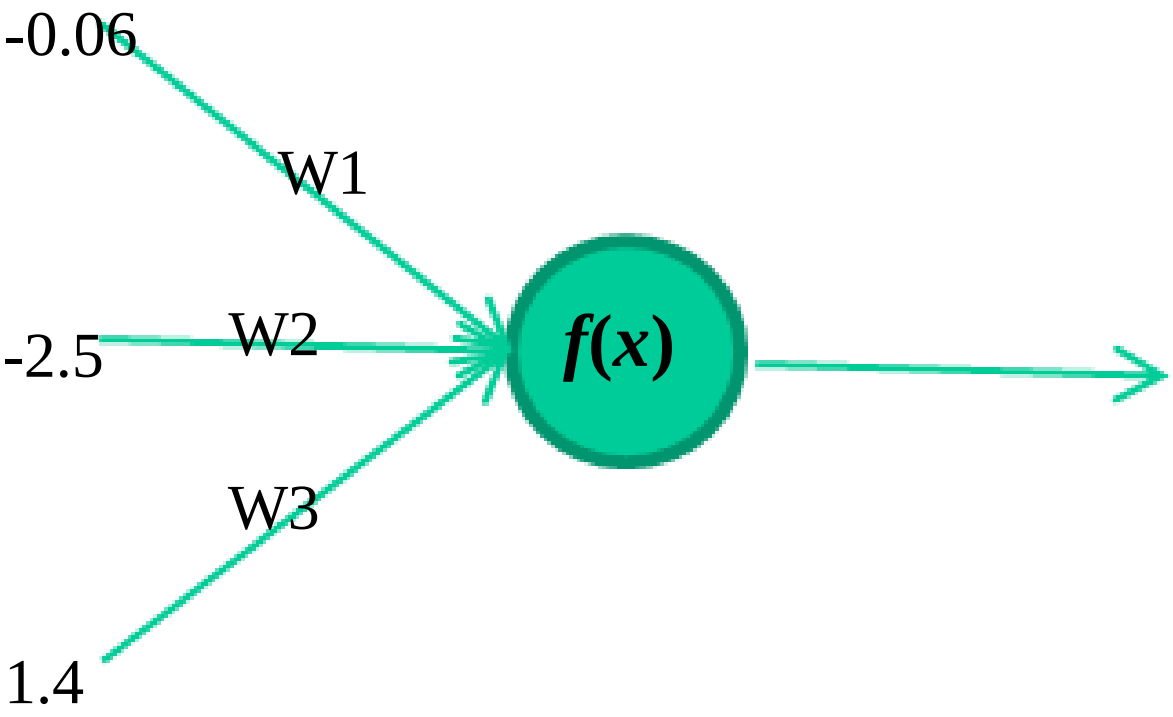
- Two main operations
  - Forward propagation
  - Backward propagation (weight adjustment)



Artificial neuron

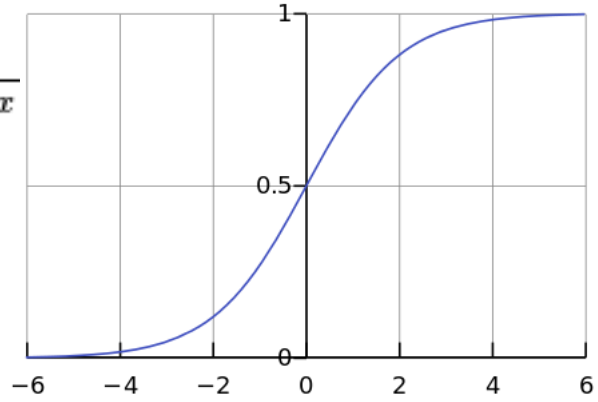
$$f(x) = \frac{1}{1 + e^{-x}}$$

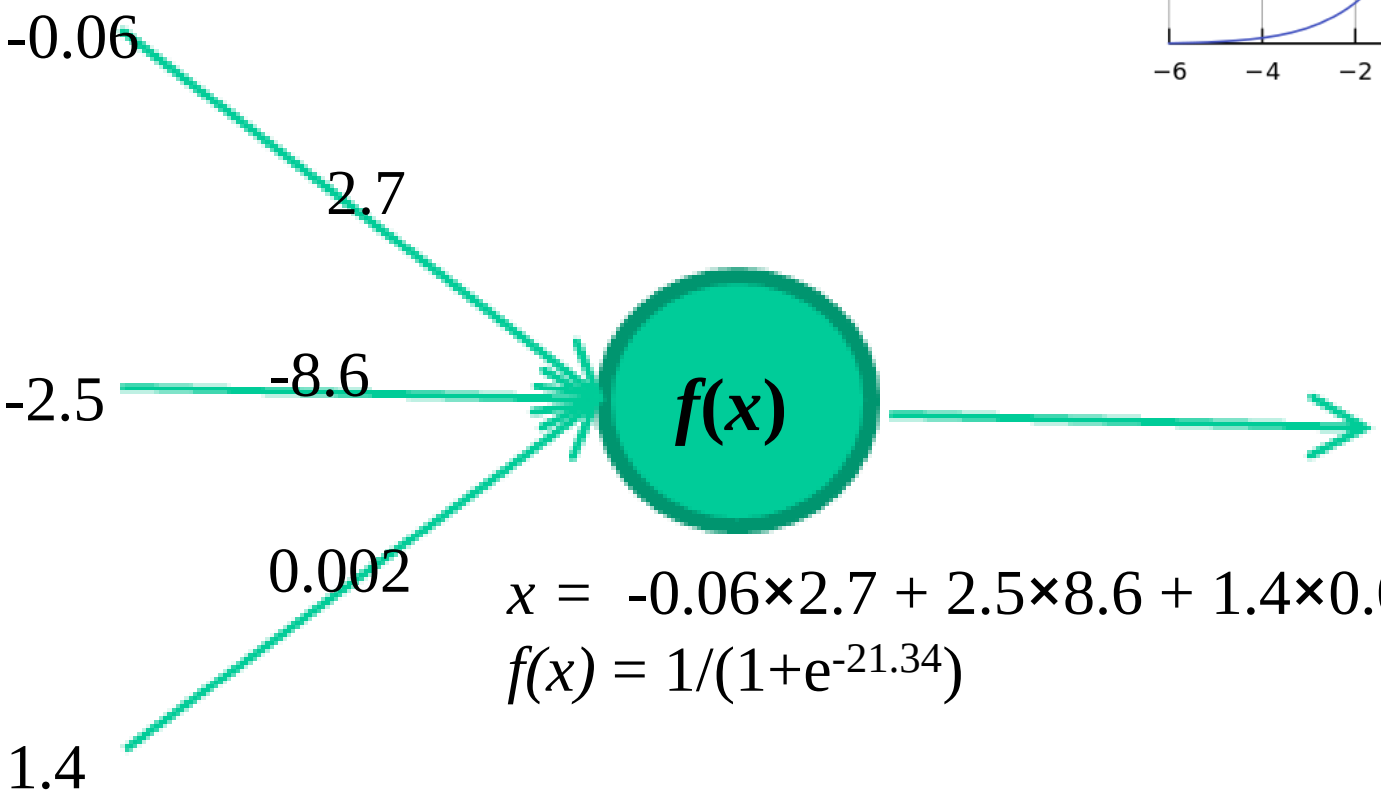




Artificial neuron

$$f(x) = \frac{1}{1 + e^{-x}}$$



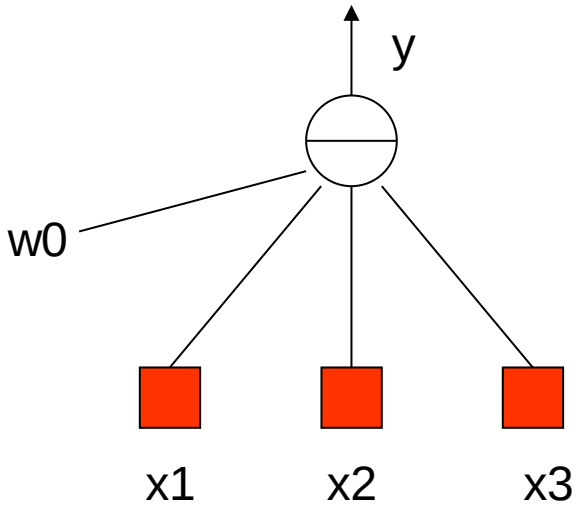


$x = -0.06 \times 2.7 + 2.5 \times 8.6 + 1.4 \times 0.002 = 21.34$

$f(x) = 1/(1+e^{-21.34})$

General View: Artificial neuron

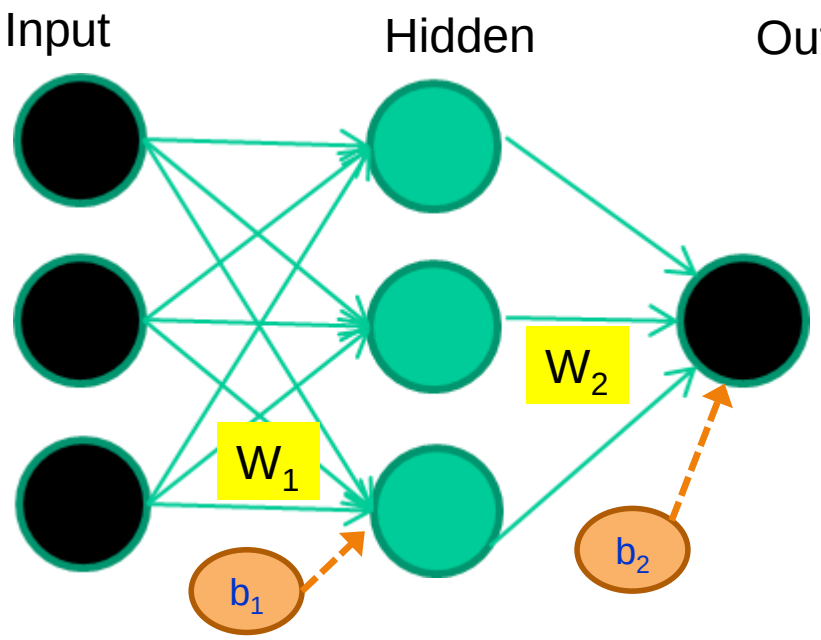
- An *Artificial Neuron* is a non-linear parameterized function with restricted output range



$$y = f\left(w_0 + \sum_{i=1}^{n-1} w_i x_i\right)$$

w_0 also called *a bias term (b_i)*

Multiple Layers: Artificial neuron

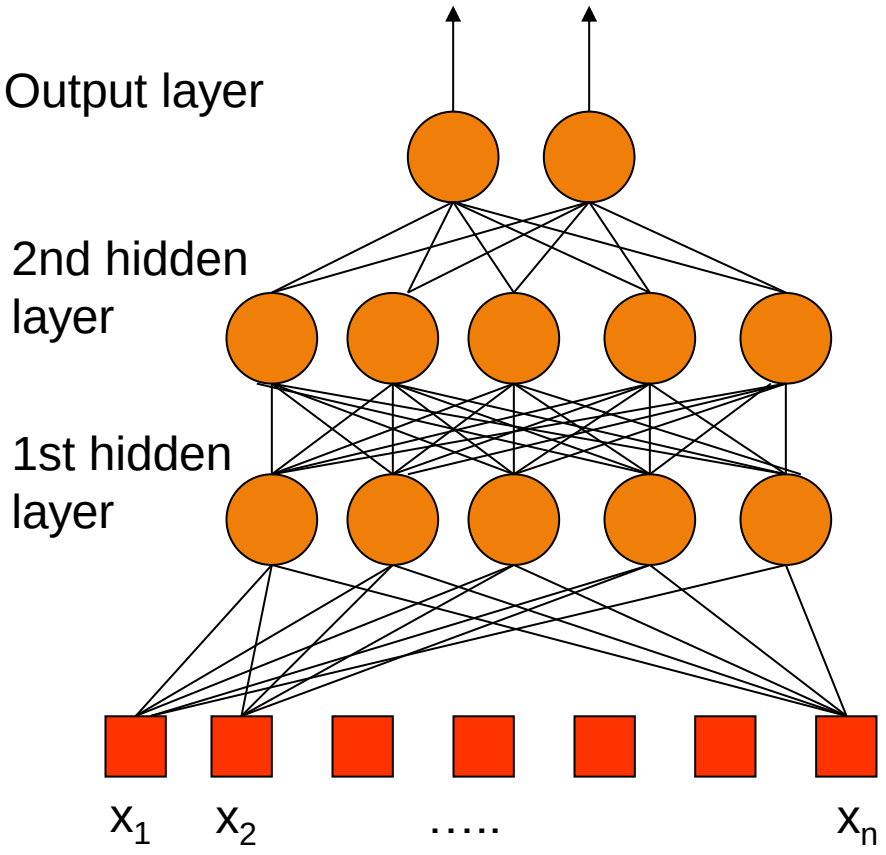


x $\quad$ **y_1** $\quad$ **y_2** $\quad$ VECTORS

$$\mathbf{y}_1 = f_1(W_1\,\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{y}_2 = f_2(W_2\,\mathbf{y}_1 + \mathbf{b}_2)$$

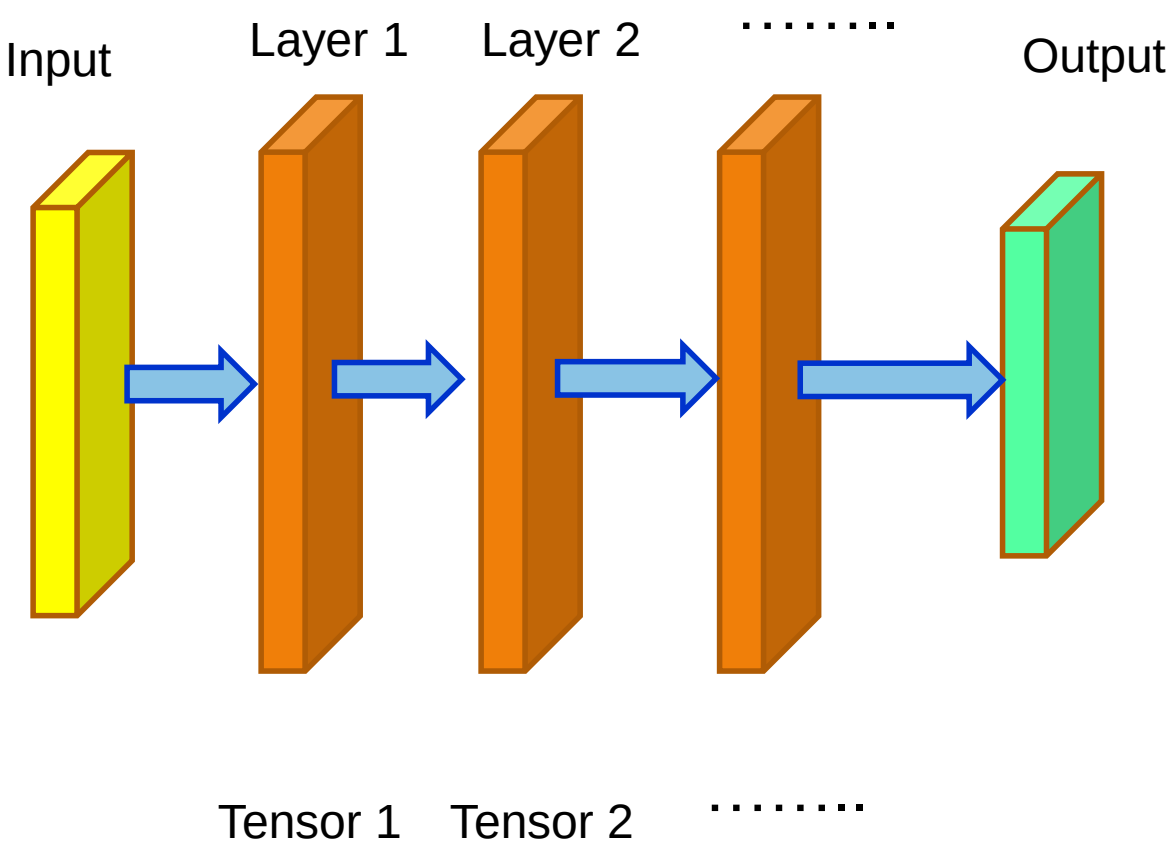
Feed Forward Neural Networks



- The information is propagated from the inputs to the outputs
  - Directed Acyclic Graph (DAG)
- Computes one or more non-linear functions
  - Computation is carried out by composition of some number of algebraic functions implemented by the connections, weights and biases of the hidden and output layers
- Hidden layers compute intermediate representations
  - Dimension reduction
- Time has no role -- no cycles between outputs and inputs

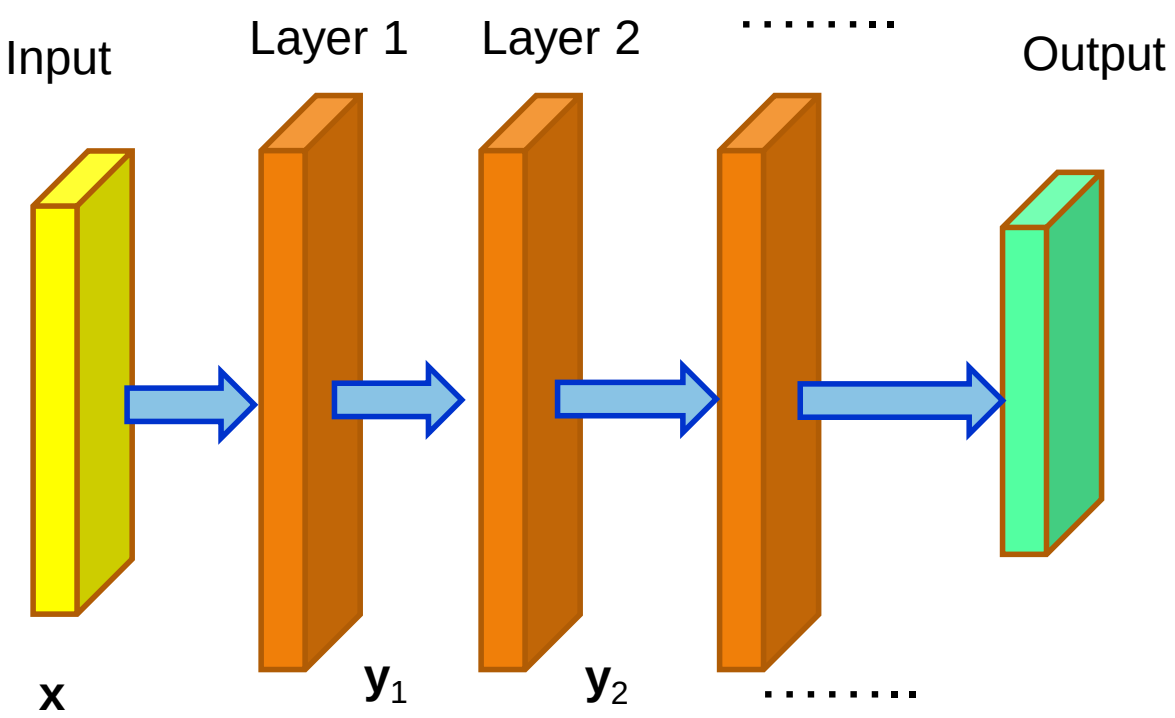
We say that the input data, or features, are *n* dimensional

General Case: Deep Network



tensor: an algebraic object that describes a mapping from one set of algebraic objects to another

General Case: Deep Network



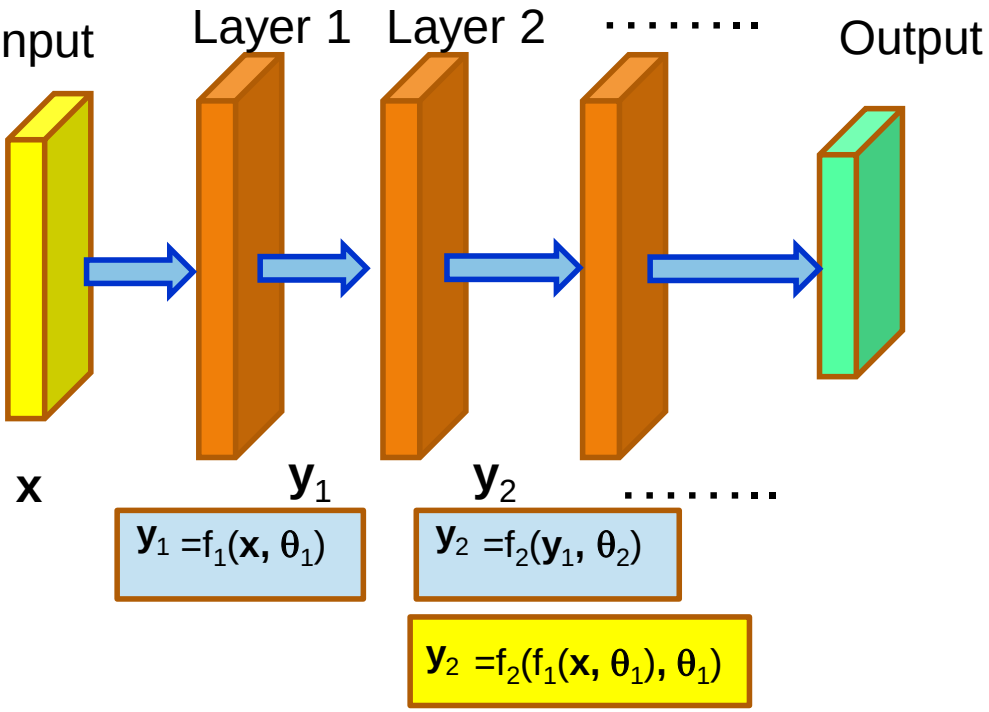
$$y_1 = f_1(x, \theta_1)$$

$$y_2 = f_2(y_1, \theta_2)$$

$$y_2 = f_2(f_1(x, \theta_1), \theta_1)$$

$$y_N(x, \theta_1, \theta_2, \ldots \theta_N) = f_N(f_{N-1}(\ldots(f_1(x, \theta_1), \theta_{N-1}), \theta_N)$$

Training: Deep Network



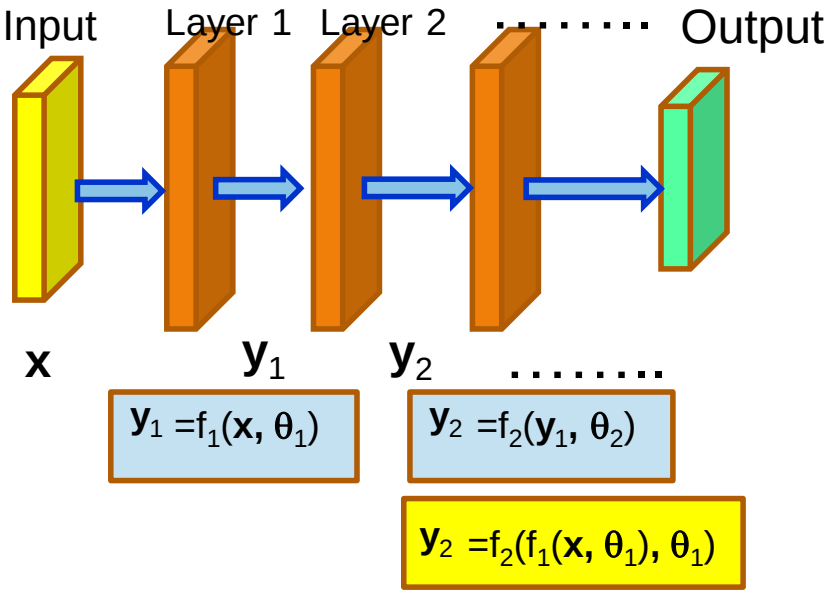
J: objective function (learning)

$$y_N(x, \theta_1, \theta_2, \ldots \theta_N) = f_N(f_{N-1}(\ldots(f_1(x, \theta_1), \theta_{N-1}), \theta_N)$$

Loss (cost) function

$$\theta^* = \arg\min_\theta \sum_{(x,y)\in(X,Y)} J[y, fL(x, \theta_{1, \ldots L})]$$

Summary: Deep Network



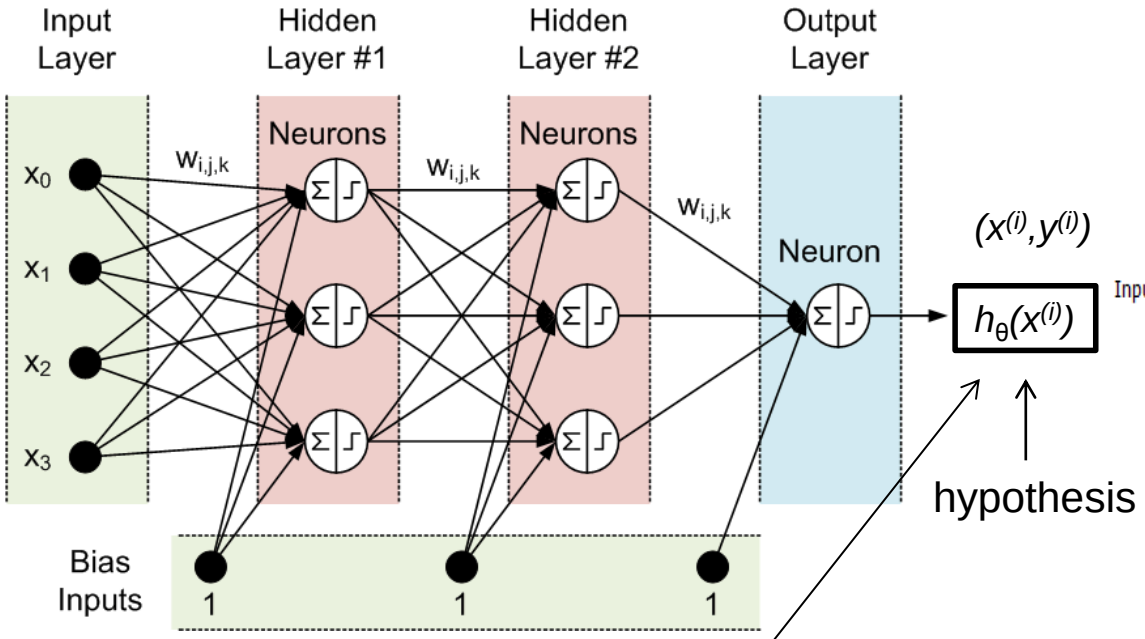
Deep Network: family of parametric, non-linear, hierarchical representations

$$y_N(x, \theta_1, \theta_2, \ldots \theta_N) = f_N(f_{N-1}(\ldots(f_1(x, \theta_1), \theta_{N-1}), \theta_N)$$

Training: optimize network parameters to minimise loss over training set

$$\theta^* = \arg\min_\theta \sum_{(x,y)\in(X,Y)} J[y, fL(x, \theta_{1, \ldots L})]$$

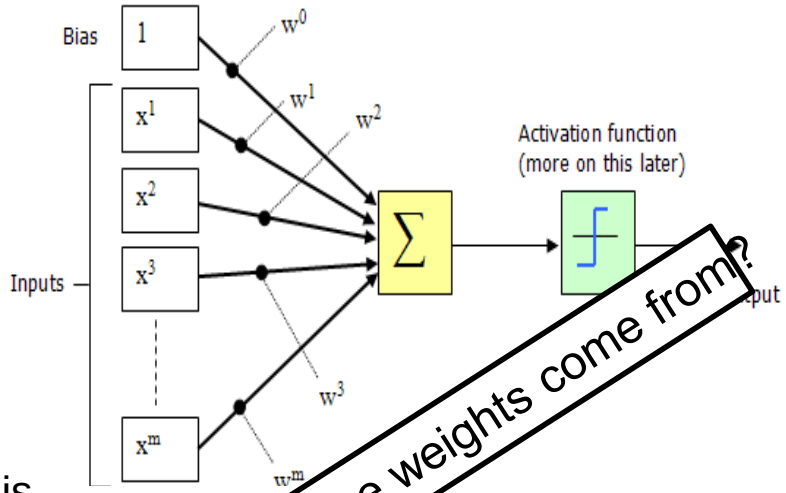
Deep Feed Forward Neural Nets (in 1 Slide (☺))



$(x^{(i)},y^{(i)})$

$h_\theta(x^{(i)})$

hypothesis

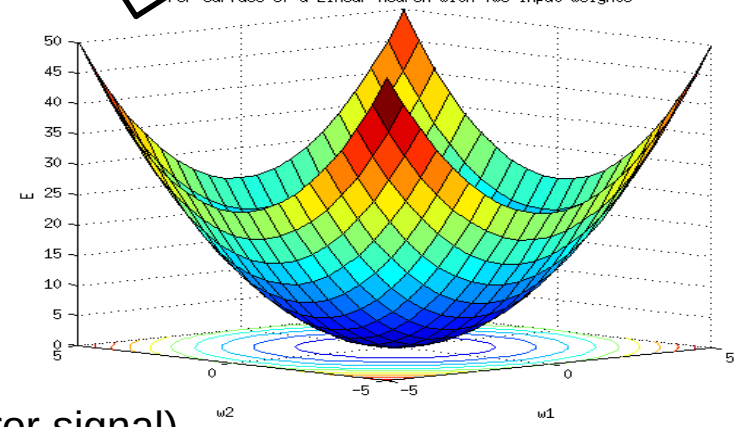


Where do the weights come from?

Forward Propagation

$$J(\theta_0,\theta_1,\ldots,\theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

g?



Learning is the adjusting of the weights $w_{i,j}$ such that the cost function $J(\boldsymbol{\theta})$ is minimized (a form of Hebbian learning).

Simple learning procedure: *Back Propagation* (of the error signal)

Overview of Topics

- Representations for Deep Learning
- Underlying Mathematics
- Inference
  - TensorFlow
  - BackPropagation
- Network Classes
  - CNN
  - Temporal Deep Networks (RNN, LSTM)
  - Deep Reinforcement Learning
- Practical Aspects