Zeppelin

Learning Machine Le...

Learning Machine Learning

There are many good free online college level classes for Machine line. The course offered by Stanford Professor, Andrew Ng is the one I took from Coursera. I am continue suppliment my learning from other online classes and YouTube video and blog. This note is to share the learning experience.

Instructors:

Andrew Ng

Associate Professor, Stanford University; Chief Scientist, Baidu; Chairman and Co-founder, Coursera

Courses

Course Contents (https://www.coursera.org/learn/machine-learning)

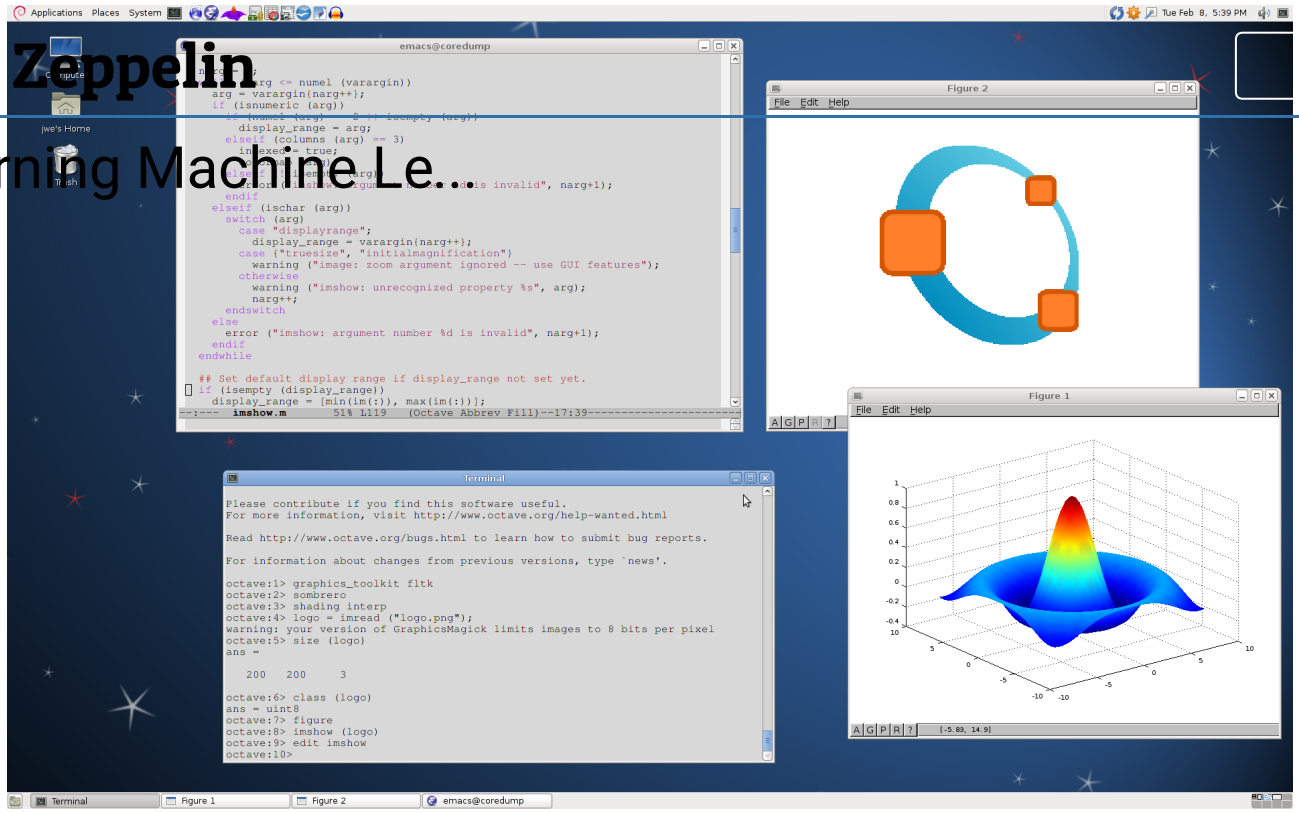
Pre-requistes

(They will be reviwed in class):

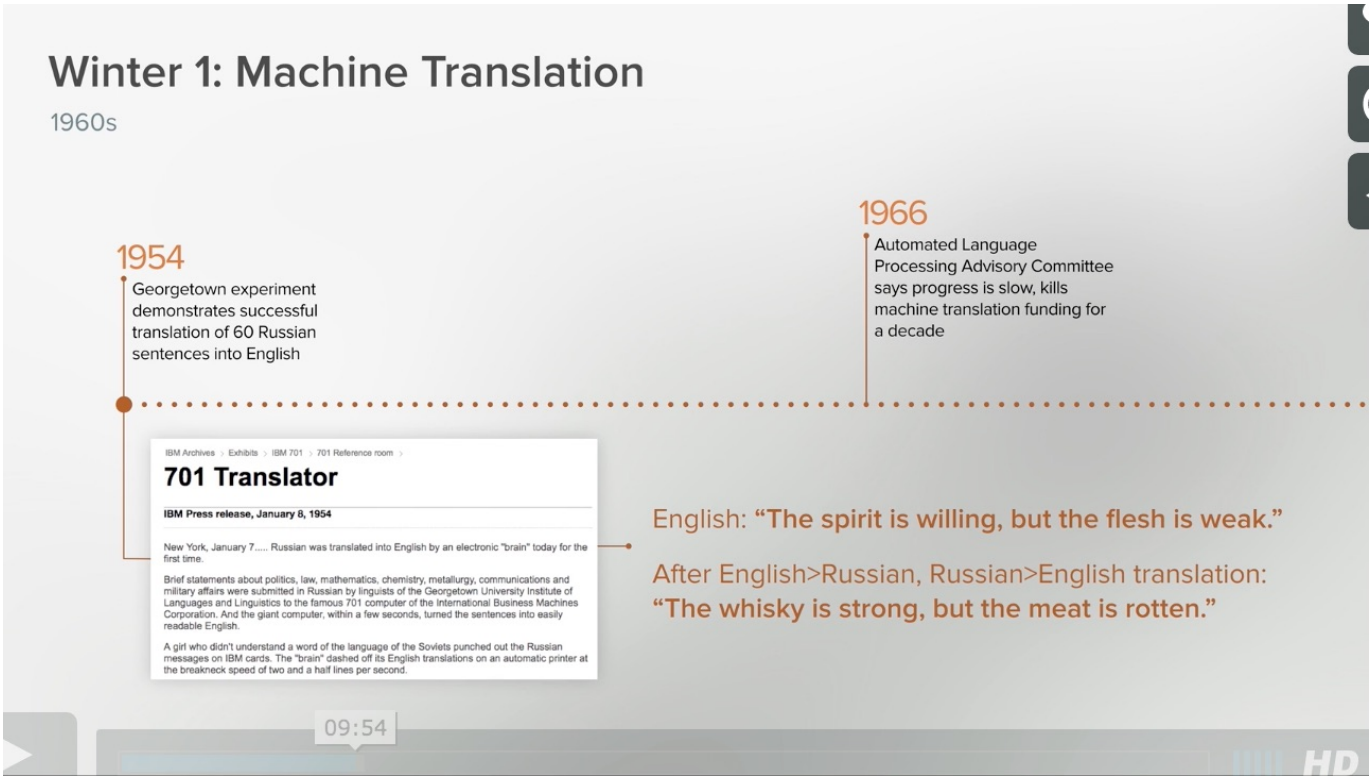
- Linear Algebra (https://www.khanacademy.org/math/linear-algebra)
- Octave (http://wiki.octave.org/Video_tutorials)

Zeppelin

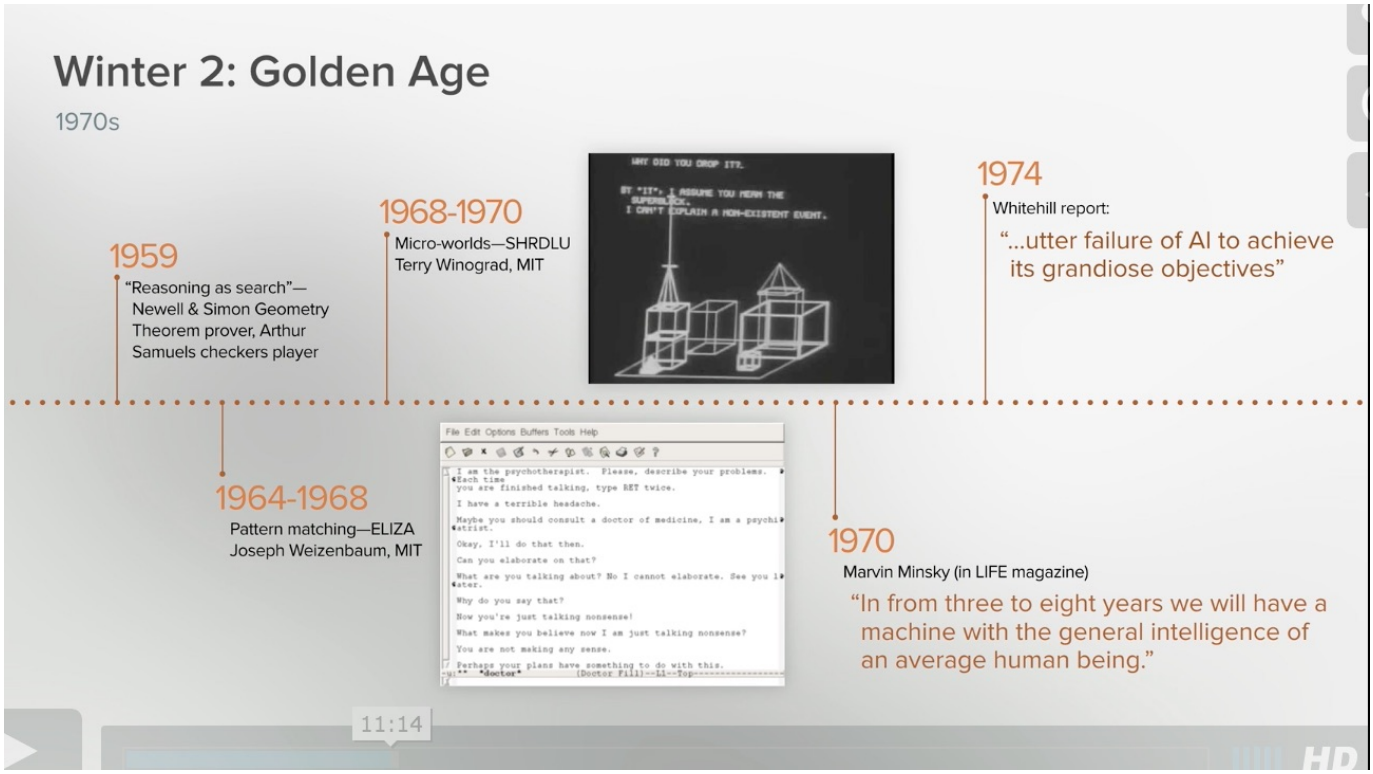
Learning Machine Le...



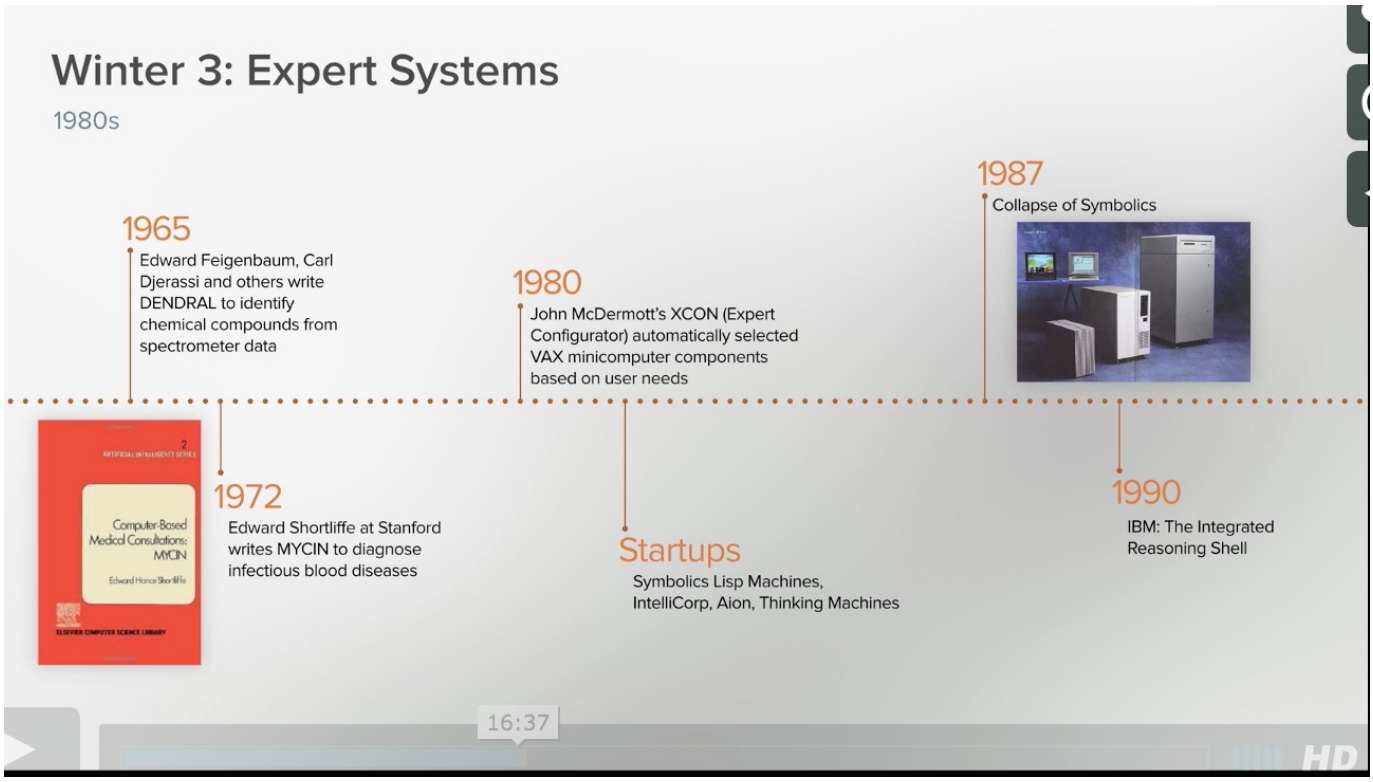
History of Machine Learning - Winter 1



History of Machine Learning - Winter 2



History of Machine Learning - Winter 3



New Period of Machine Learning

2012

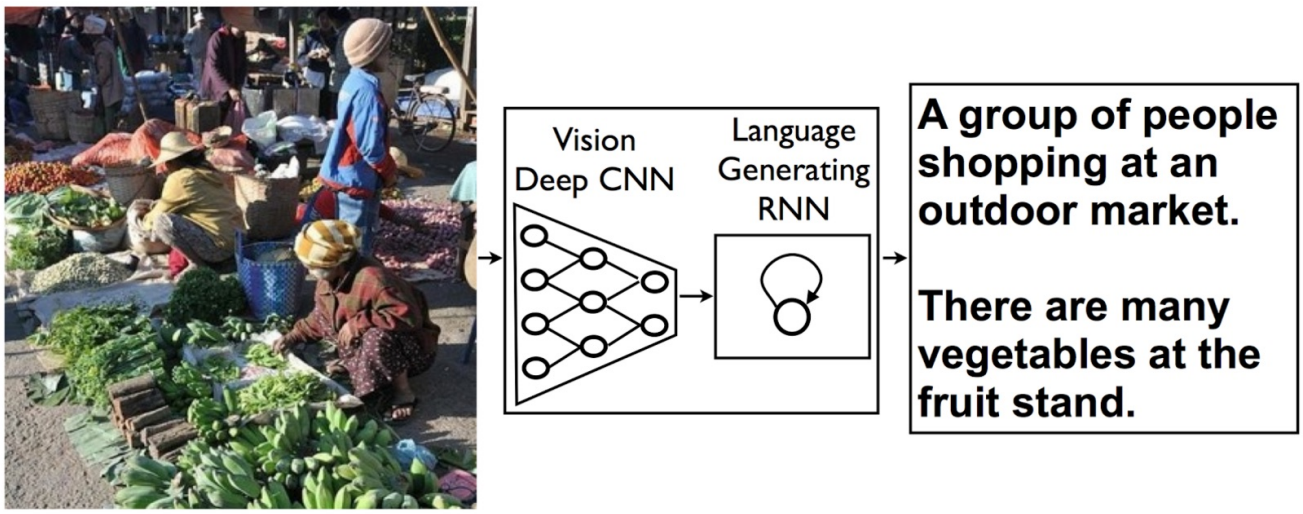
Andrew Ng and his team made major breakthough in face detector from unlabeled images: from the article Building High-level Features Using Large Scale Unsupervised Learning (http://www.robotics.stanford.edu/~ang/papers/icml12-HighLevelFeaturesUsingUnsupervisedLearning.pdf)

We train a 9-layered locally connected sparse autoencoder with pooling and local contrast normalization on a large dataset of images (the model has 1 billion connections, the dataset has 10 million 200x200 pixel images downloaded from the Internet). We train this network using model parallelism and asynchronous SGD on a cluster with 1,000 machines (16,000 cores) for three days.

Contrary to what appears to be a widely-held intuition, our experimental results reveal that it is possible to train a face detector without having to label images as containing a face or not. Control experiments show that this feature detector is robust not only to translation but also to scaling and out-of-plane rotation. We also find that the same network is sensitive to other high-level concepts such as cat faces and human bodies. Starting with these learned features, we trained our network to obtain 15.8% accuracy in recognizing 20,000 object categories from ImageNet, a leap of 70% relative improvement over the previous state-of-the-art.

2015

Stanford University worked on Neural Network which can explain unlabeled photo. For example:



Source: Google

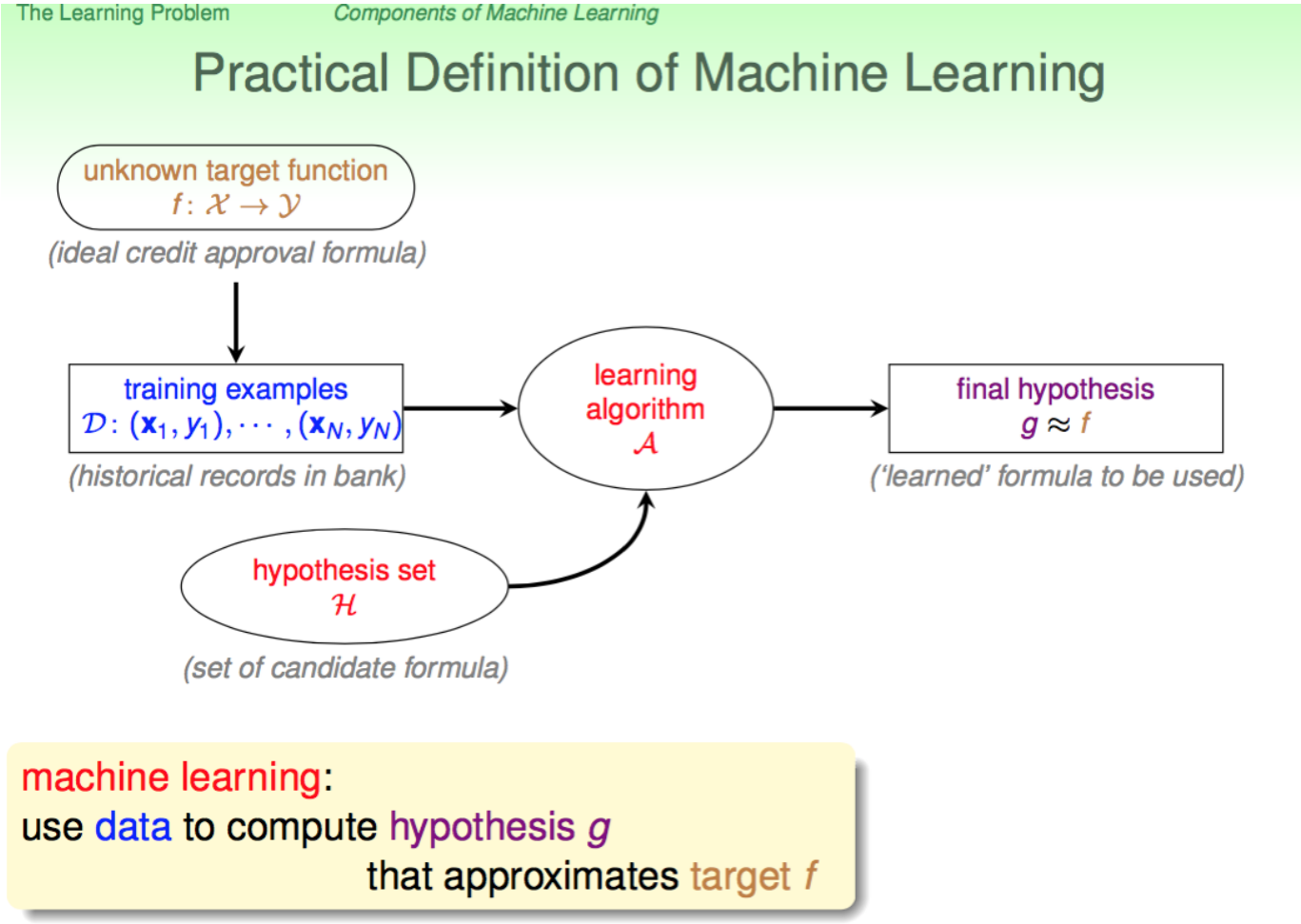
What is Machine Learning?

Machine Learning is concerned with the development, the analysis, and the application of algorithms that allow computers to learn.

Learning:

- A computer program is said to learn from experience (E) with some class of tasks (T) and a performance measure (P) if its performance at tasks in T as measured by P improves with E. (i.e. by collecting data)
- Extracting a model of a system from the sole observation (or the simulation) of this system in some situations.
    - A model = some relationships between the variables used to describe the system.
- Two main goals: make prediction and better understand the system.

Definition of Machine Learning



Components of Machine Learning

problem: unknown target function, *f*

- to find out the pattern for approving the credit card that benefit to a bank. A target function ***f***, which maps applicant ***X*** (information about different application) that leads to outcome of ***Y*** (different out comes).

training examples, *D*

- input: information of each applicant, x: age, salary, exist debts,etc
- output: out come of each applicant, y: good or bad for bank/late payment/default
- collected data, ***D***: {(x1, y1), (x2, y2), … (xn, yn)}

hypothesis set, *H*

- There is a set of h in ***H***, we like to find a specific h, good skill, hopefully have good performance. We select the best h, we call it ***g***
- function ***g***, is part of H = {hk}, that can map ***X -> Y*** with good accuracy

learning algorithm, *A*

- Use data to compute the best hypothesis, ***g***, which approximates to ***f***

target fountion, *g*

- will be used to forecast future applicants.

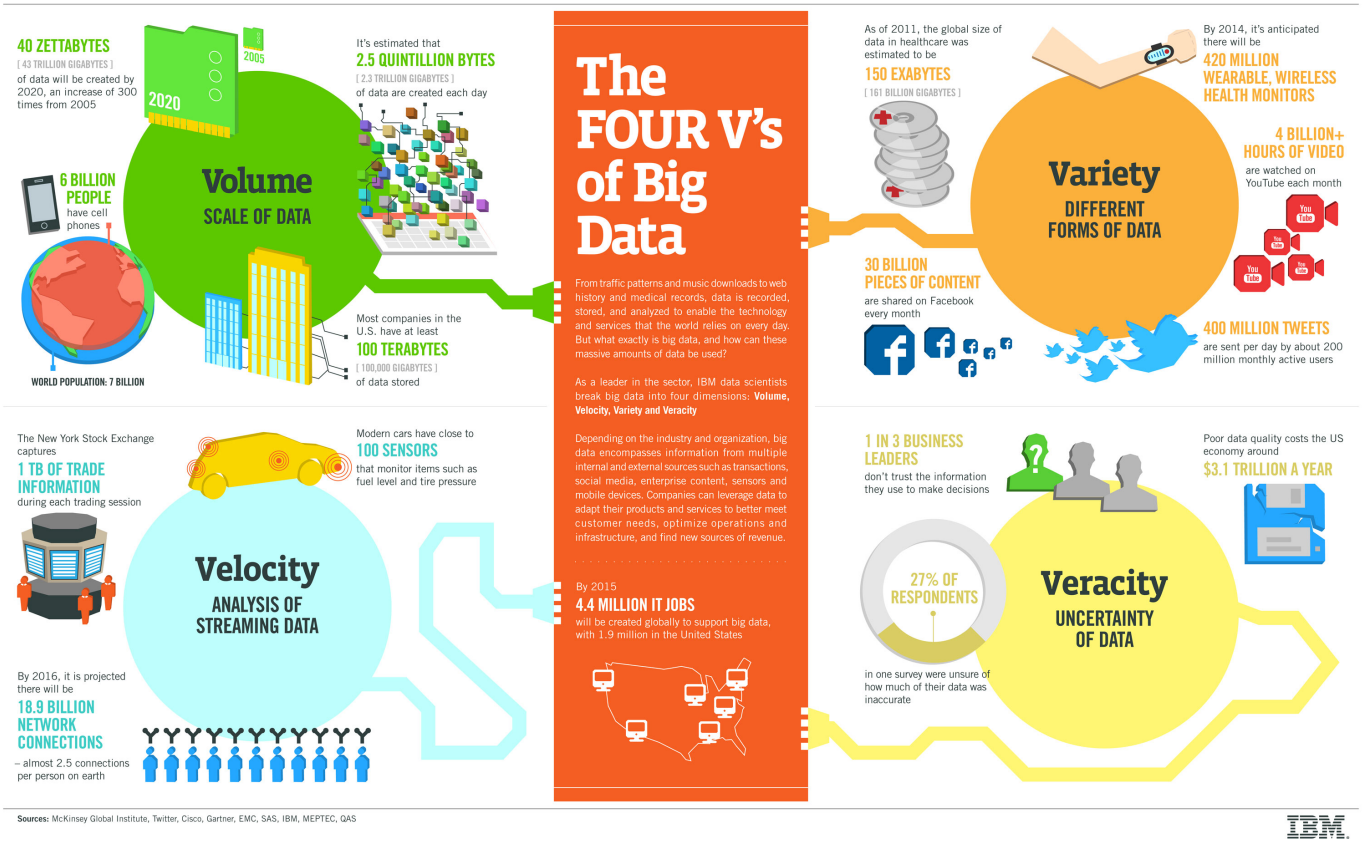
Learning Model

learning algorithm, ***A*** and hypothesis set, ***H***

Why Now?

More Data

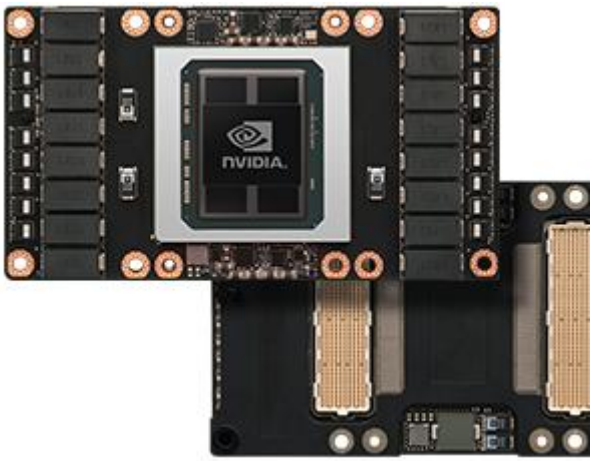
- Increase of data **Volume, Variety, Velocity, and Veracity**
- awesome-public-datasets (https://github.com/caesar0301/awesome-public-datasets)



Dedicate Machine Learning Hardware

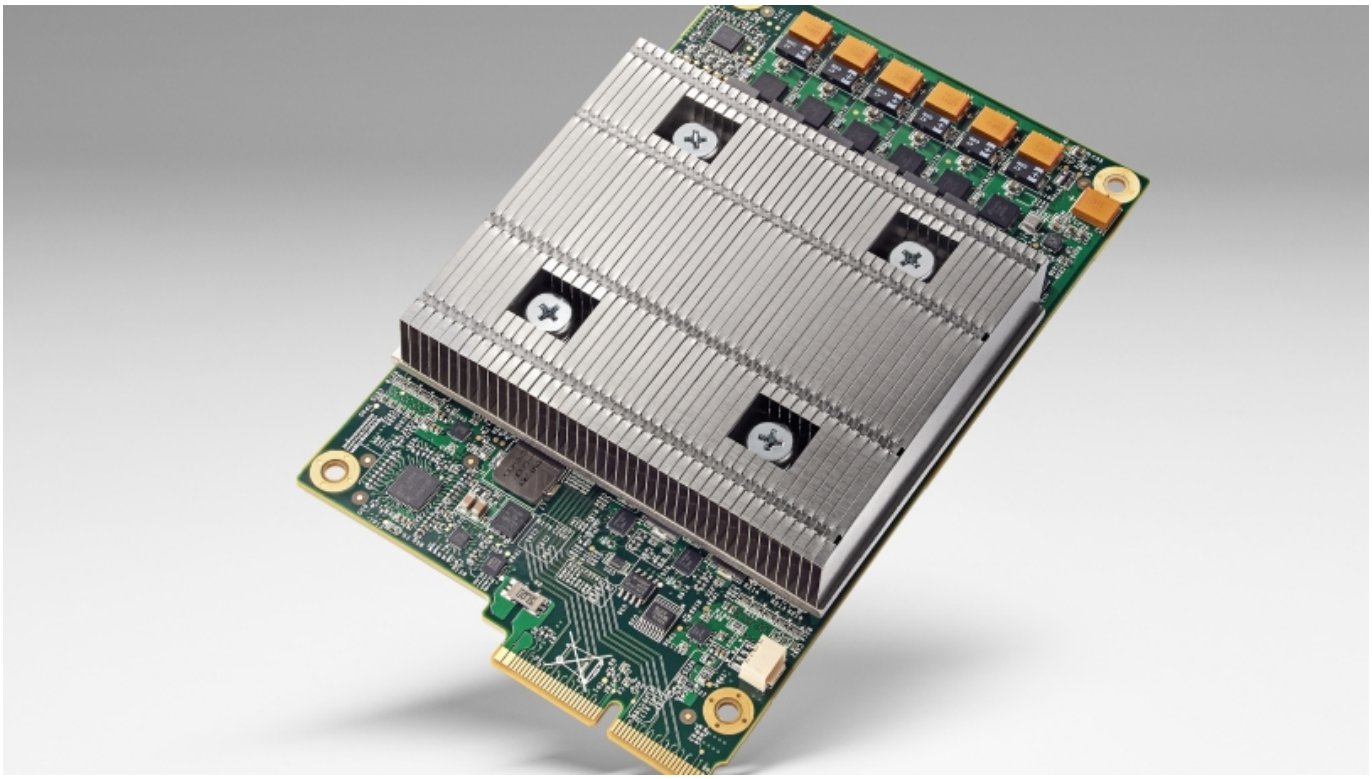
- Increase of computing power with dedicate hardware, Deep Learning Supercomputer in a box.

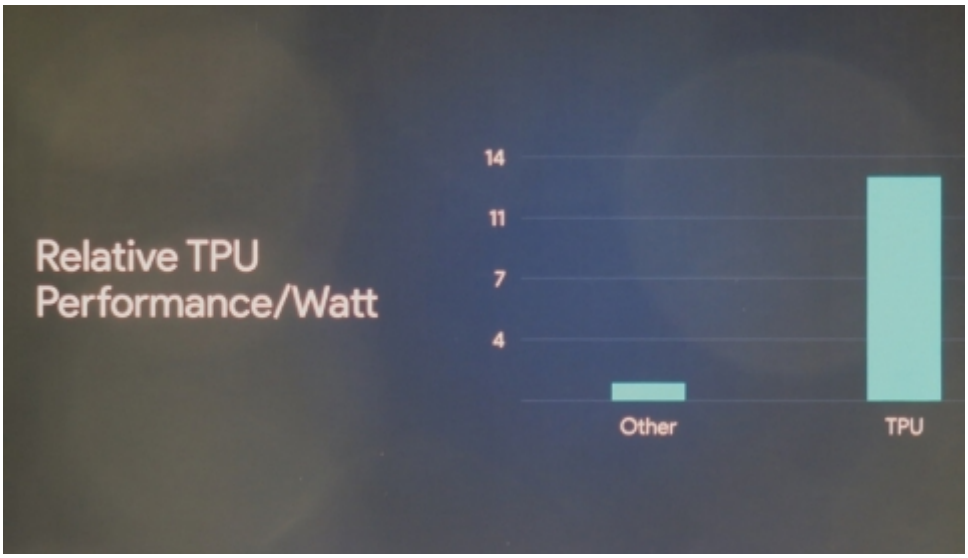
- Nvidia Tesla P100 (https://www.technologyreview.com/s/601195/a-2-billion-chip-to-accelerate-artificial-intelligence/)



Google TPU

TPU runs using 8-bit integer math, instead of the higher-precision floating-point math for which most modern CPUs and GPUs are designed. Most machine-learning algorithms can get by fine with lower resolution data, which means the chip can handle more operations in a given area and tackle more complex models efficiently.



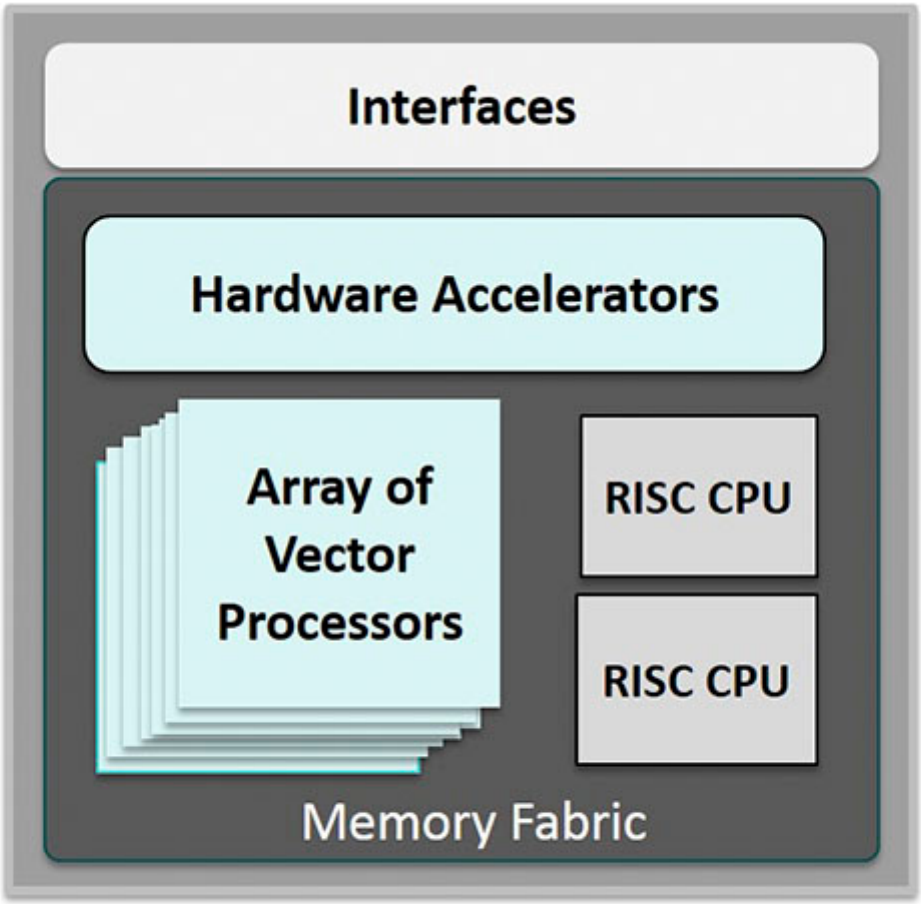


More AI Related Chips

- MIT's 168-core chip could give big brains to mobile devices and robots (http://www.pcworld.com/article/3029972/components-processors/mits-168-core-chip-could-make-mobile-devices-robots-smarter.html). MIT says its chips would require a fraction of the resources, and is 10 times more power efficient than a mobile graphics processor. It would be possible to use the chip in wearables, smartphones and battery-operated robots.

  Eyeriss will bring self-contained AI capabilities to devices with most of the processing happening locally on a device. Wi-Fi or cellular connections won't be needed to tap into cloud services or servers for image or object recognition.

- A chip startup Movidius (http://www.movidius.com/) makes low-power chips it calls vision processing units (or VPUs), which can be part of mobile device.



Myriad 2 Vision Processor Unit (VPU)

- Intel purchased Altera, a leading maker of FPGAs (field-programmable gate arrays), which are somewhere in the middle; they're not as general purpose as GPUs or as specifically designed for TensorFlow as Google's chip, but can be programmed to do a variety of tasks. Microsoft has been experimenting with Altera FPGAs for deep learning.
- IBM is developing its TrueNorth Neurosynaptic chip designed specifically for neural nets, which has recently begun to be used in a variety of applications.
- Cadence (Tensilica), Freescale, and Synopsys are pushing their DSPs (digital-signal processors) to run these models;
- Mobileye and NXP recently announced chips designed specifically for ADAS and self-driving cars;
- Nervana have announced plans for chips specifically designed for AI.

More Algorithm Development and Communities

- More machine learning algorithms and theories are developed by researchers.
- More industry support.
    - OpenAI (https://openai.com/blog/openai-technical-goals/)

When to Used?

- We cannot fully predict the problem and human expertise does not exist (navigating on Mars).
- Humans are unable to explain their expertise (speech recognition, play chess or go).

- Solution changes in time (routing on a computer network).
- Solution needs to be adapted to particular cases (user biometrics, recommendations).
- …

Computer Language for Big Data and Machine Learning

There is a quora disussion notes (https://www.quora.com/What-is-the-best-language-to-use-while-learning-machine-learning-for-the-first-time)

A performace table from Julia website (http://julialang.org/) can be used as reference.

High-Performance JIT Compiler

Julia's LLVM-based just-in-time (JIT) compiler combined with the language's design allow it to approach and often match the performance of C. To get a sense of relative performance of Julia compared to other languages that can or could be used for numerical and scientific computing, we've written a small set of micro-benchmarks in a variety of languages: C, Fortran, Julia, Python, Matlab/Octave, R, JavaScript, Java, Lua, Go, and Mathematica. We encourage you to skim the code to get a sense for how easy or difficult numerical programming in each language is. The following micro-benchmark results were obtained on a single core (serial execution) on an Intel(R) Xeon(R) CPU E7-8850 2.00GHz CPU with 1TB of 1067MHz DDR3 RAM, running Linux:

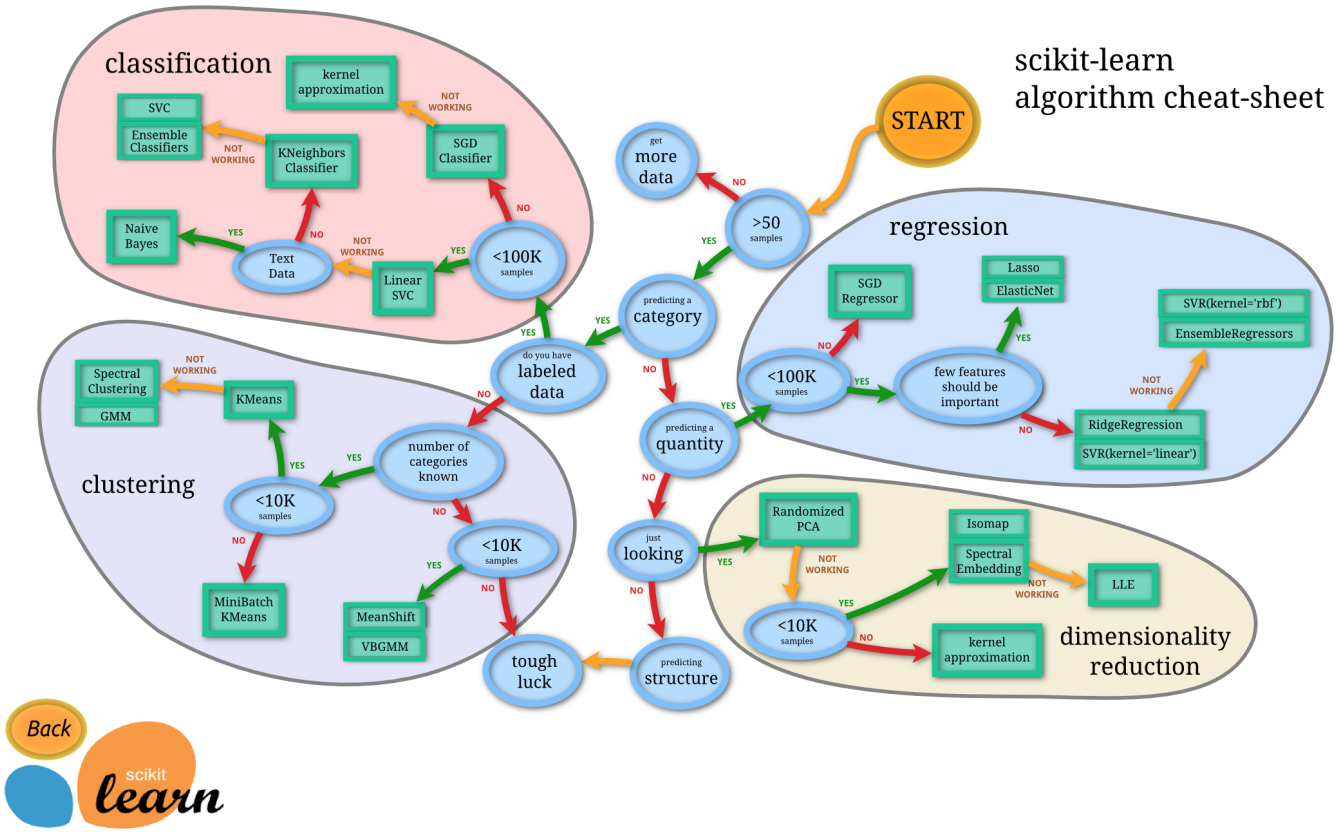
	Fortran gcc 5.1.1	Julia 0.4.0	Python 3.4.3	R 3.2.2	Matlab R2015b	Octave 4.0.0	Mathe-matica 10.2.0	JavaScript V8 3.28.71.19	Go go1.5	LuaJIT gsl-shell 2.3.1	Java 1.8.0_45
fib	0.70	2.11	77.76	533.52	26.89	9324.35	118.53	3.36	1.86	1.71	1.21
parse_int	5.05	1.45	17.02	45.73	802.52	9581.44	15.02	6.06	1.20	5.77	3.35
quicksort	1.31	1.15	32.89	264.54	4.92	1866.01	43.23	2.70	1.29	2.03	2.60
mandel	0.81	0.79	15.32	53.16	7.58	451.81	5.13	0.66	1.11	0.67	1.35
pi_sum	1.00	1.00	21.99	9.56	1.00	299.31	1.69	1.01	1.00	1.00	1.00
rand_mat_stat	1.45	1.66	17.93	14.56	14.52	30.93	5.95	2.30	2.96	3.27	3.92
rand_mat_mul	3.48	1.02	1.14	1.57	1.12	1.12	1.30	15.07	1.42	1.16	2.36

Figure: benchmark times relative to C (smaller is better, C performance = 1.0).

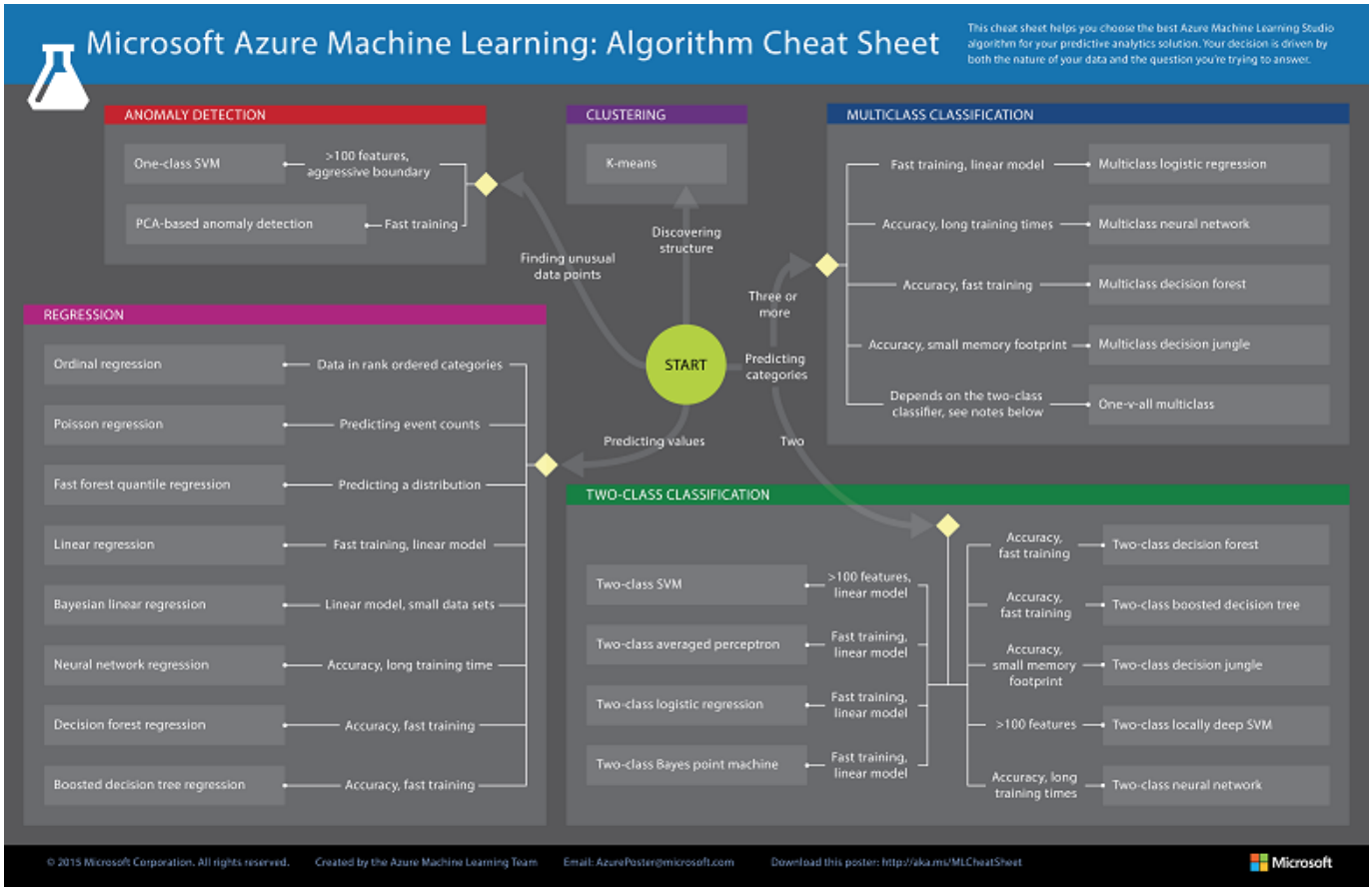
C and Fortran compiled by gcc 5.1.1, taking best timing from all optimization levels (-O0 through -O3). C, Fortran, Go, and Julia use OpenBLAS v0.2.14. Python 3 was installed from the Anaconda distribution. The Python implementations of `rand_mat_stat` and `rand_mat_mul` use NumPy (v1.9.2) functions; the rest are pure Python implementations.

Benchmarks can also be seen here as a plot created with Gadfly.

Algorithm - scikit-learn

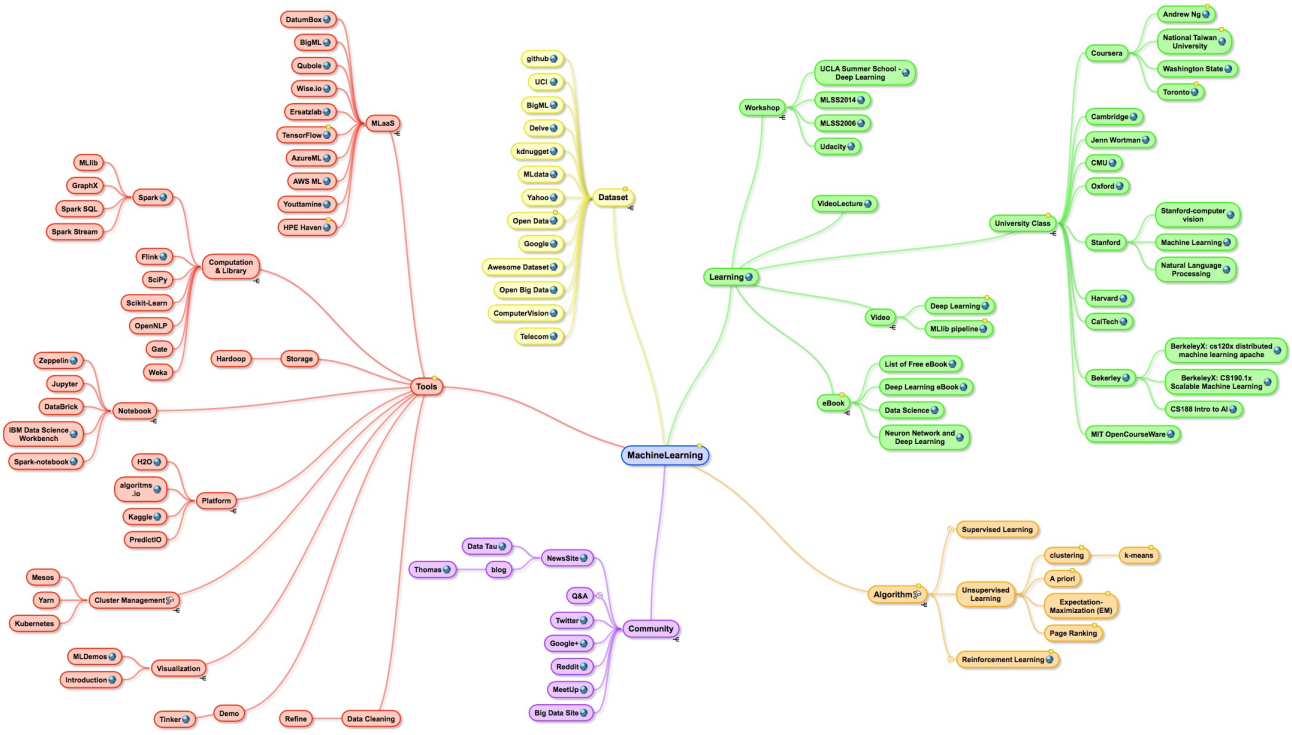


Algorithm - Microsoft Azure Machine Learning



Learning Machine Learning

An ecosystem for learning machine learning.



Algorithm

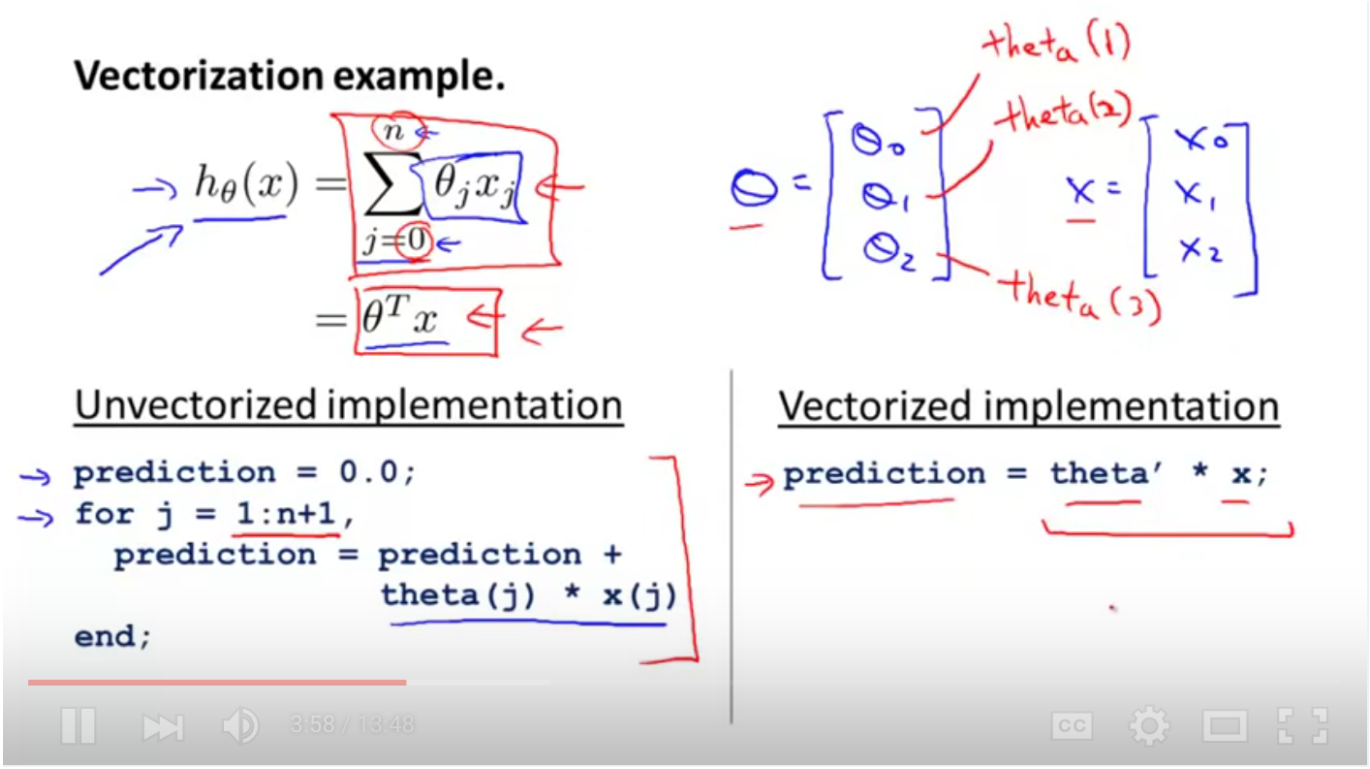
A subset of machine learing algorithm.

eBook

machine learning ebooks (https://github.com/rasbt/pattern_classification/blob/master/resources/machine_learning_ebooks.md),
deep learning (http://www.deeplearningbook.org/)
data science books (http://www.wzchen.com/data-science-books/)
neural network and deep learning (http://neuralnetworksanddeeplearning.com/chap1.html)
hacker lists - free machine learning books (https://hackerlists.com/free-machine-learning-books/)

Vectorization

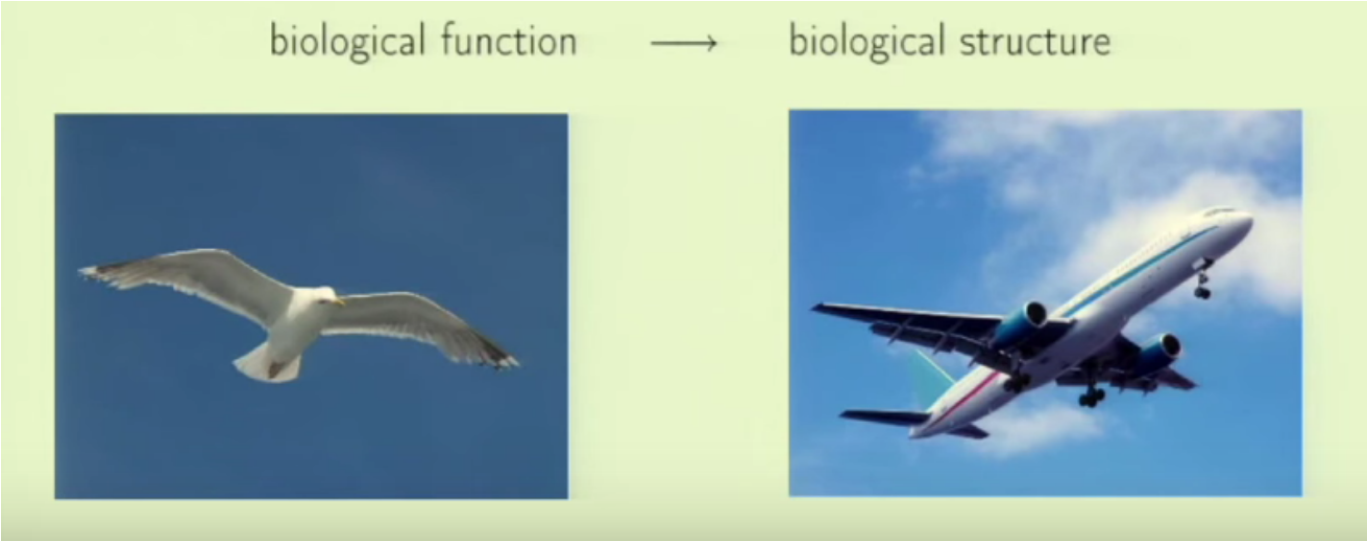
Vectorization (https://en.wikipedia.org/wiki/Vectorization_(mathematics)) makes coding easier and more readable. Many languages (https://en.wikipedia.org/wiki/List_of_programming_languages_by_type#Array_language) support Array Programming (https://en.wikipedia.org/wiki?curid=890887).

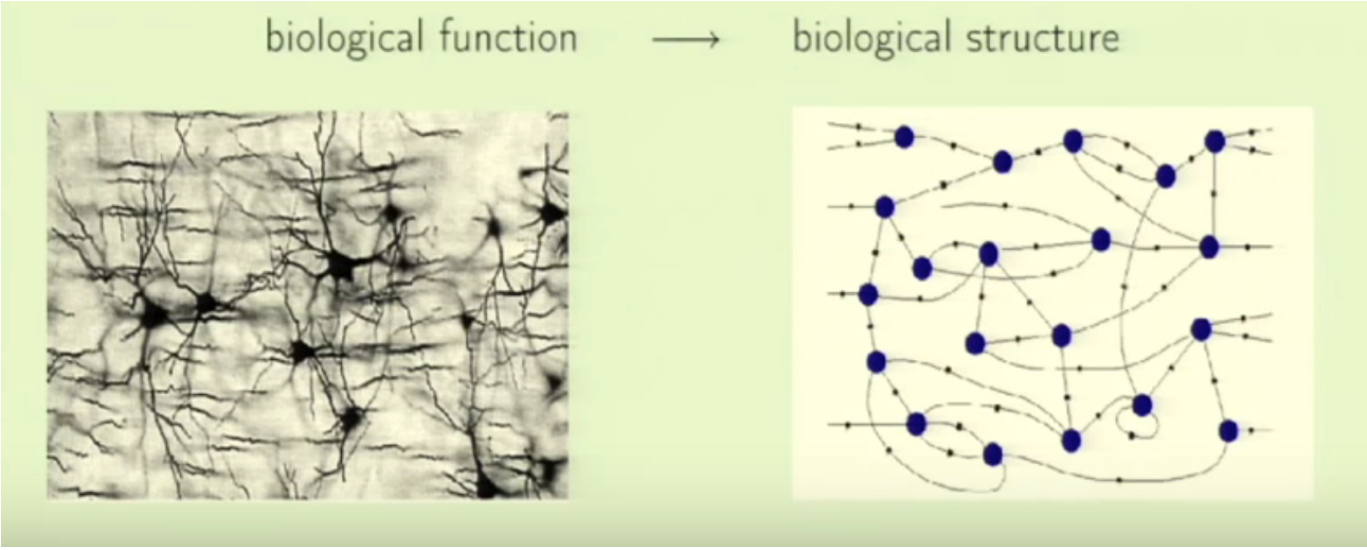


Functional Programming and Computing Nodes

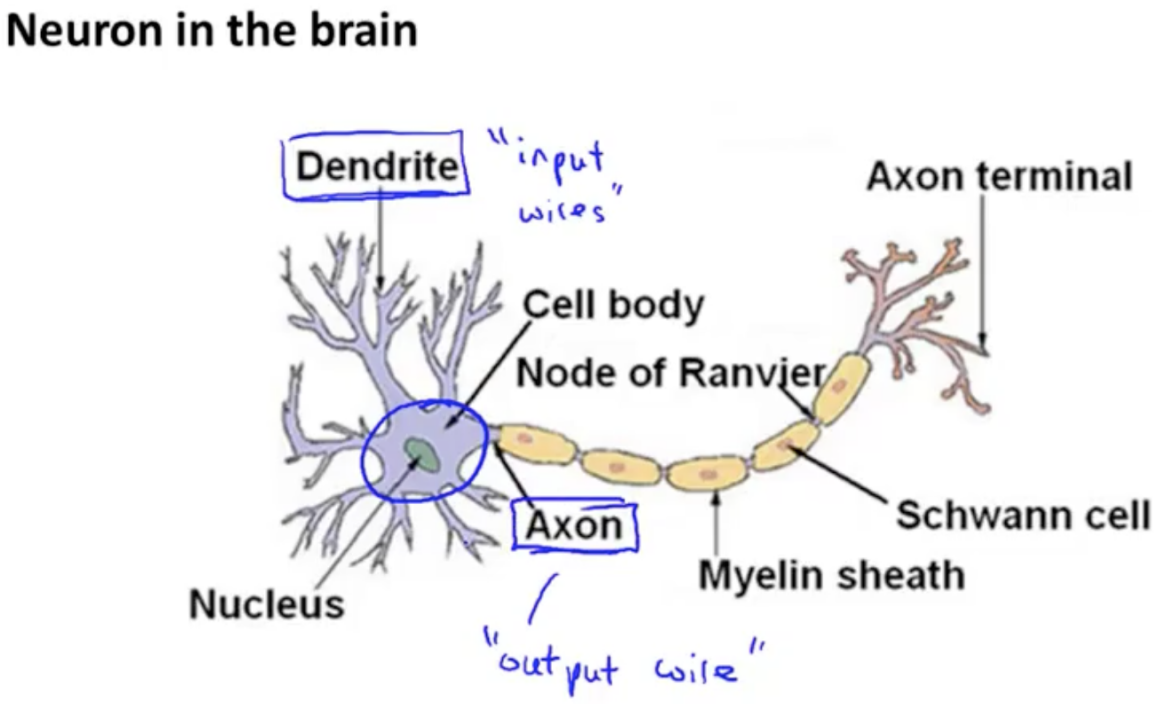
We can treat algorithm like a graph, with each node is computation and each link is data flow. This concept is described in Directed Acyclic Graph ([https://en.wikipedia.org/wiki/Directed_acyclic_graph]) (DAG)

Learning from Nature

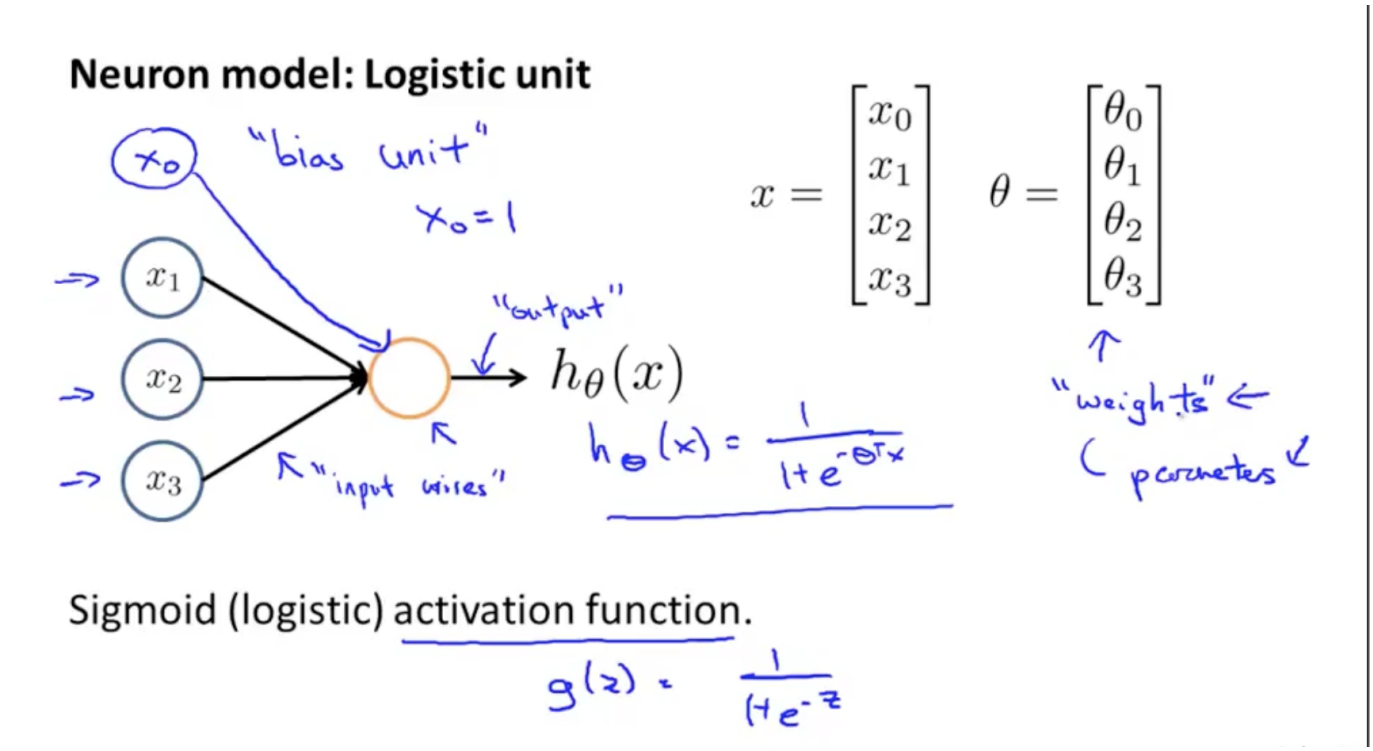




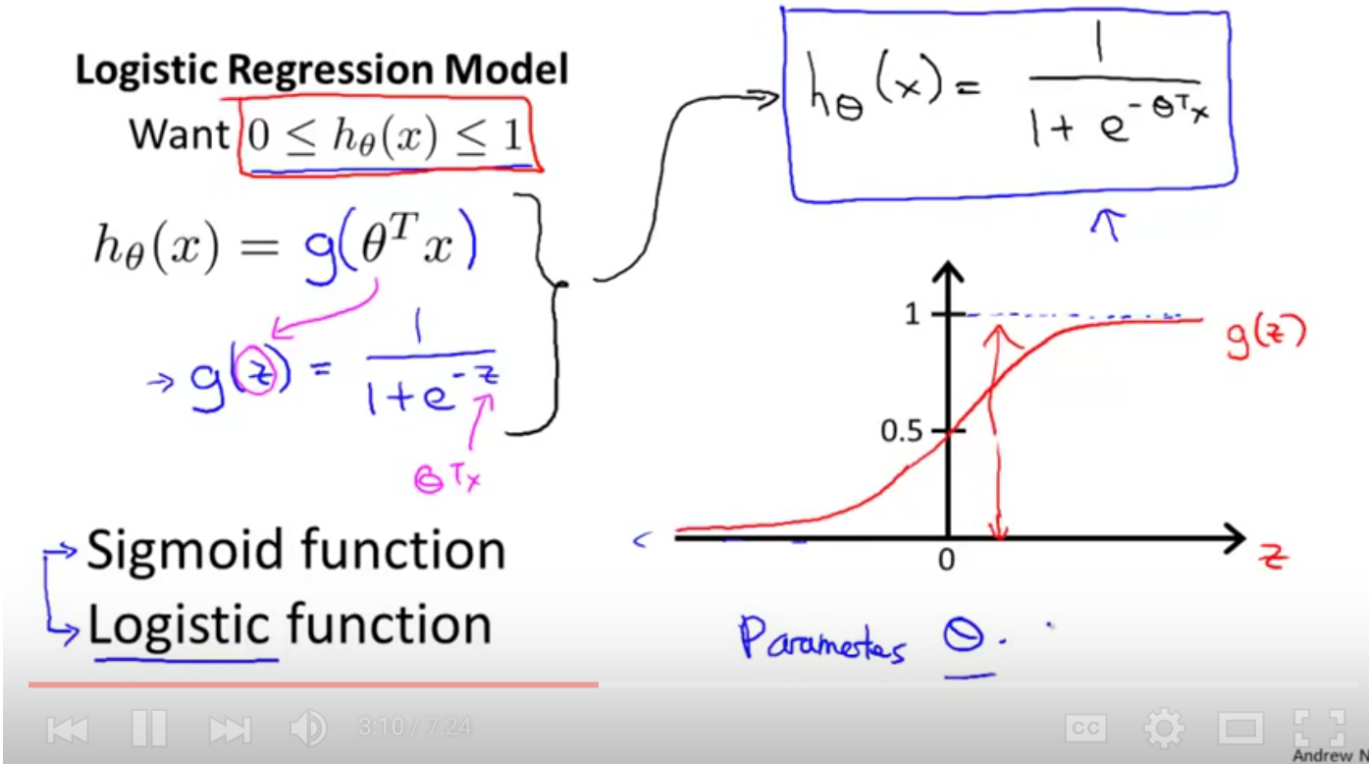
Bio Neural Network



Artificial Neuron Wiki (https://en.wikipedia.org/wiki/Artificial_neuron#Comparison_to_biological_neurons)

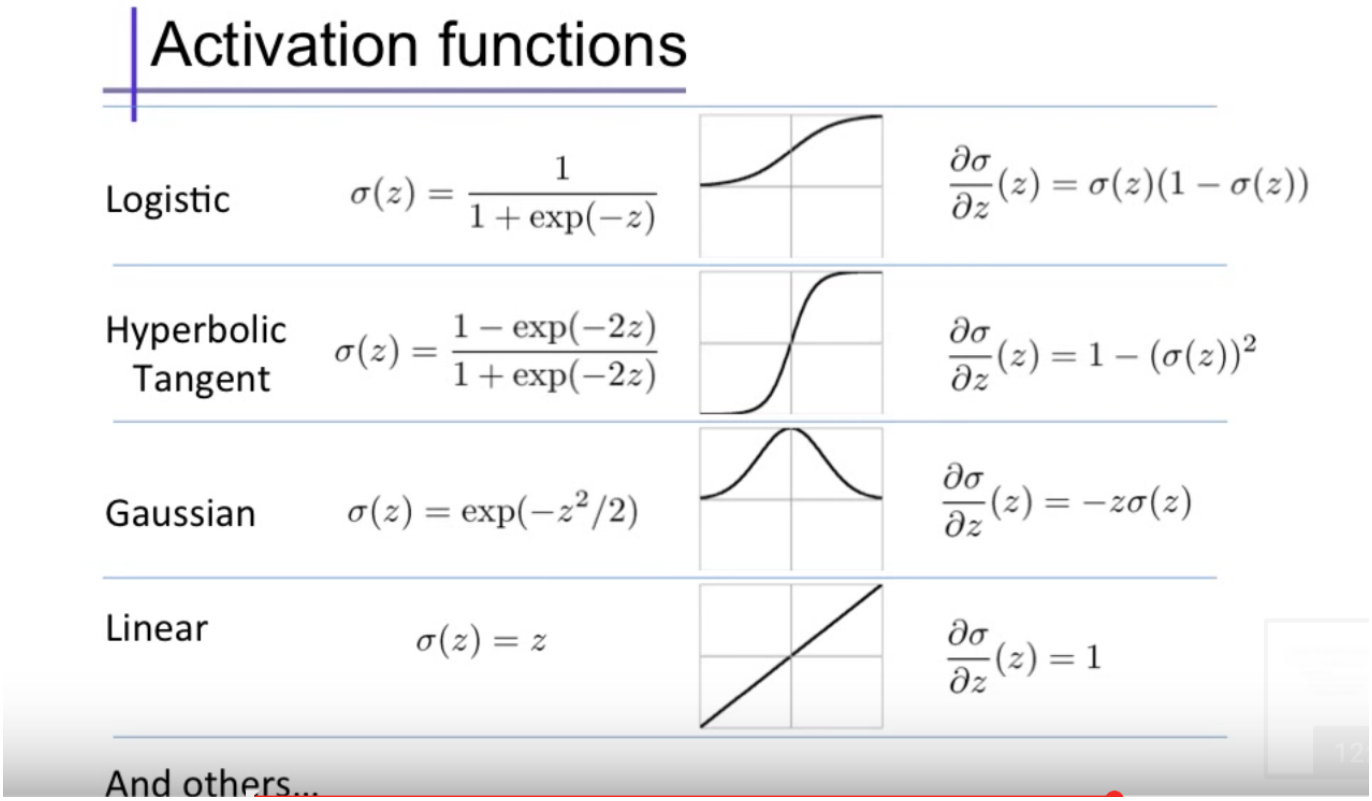


The first layer can be considered as a (trained) feature layer, then second layer is feature-feature layer,…



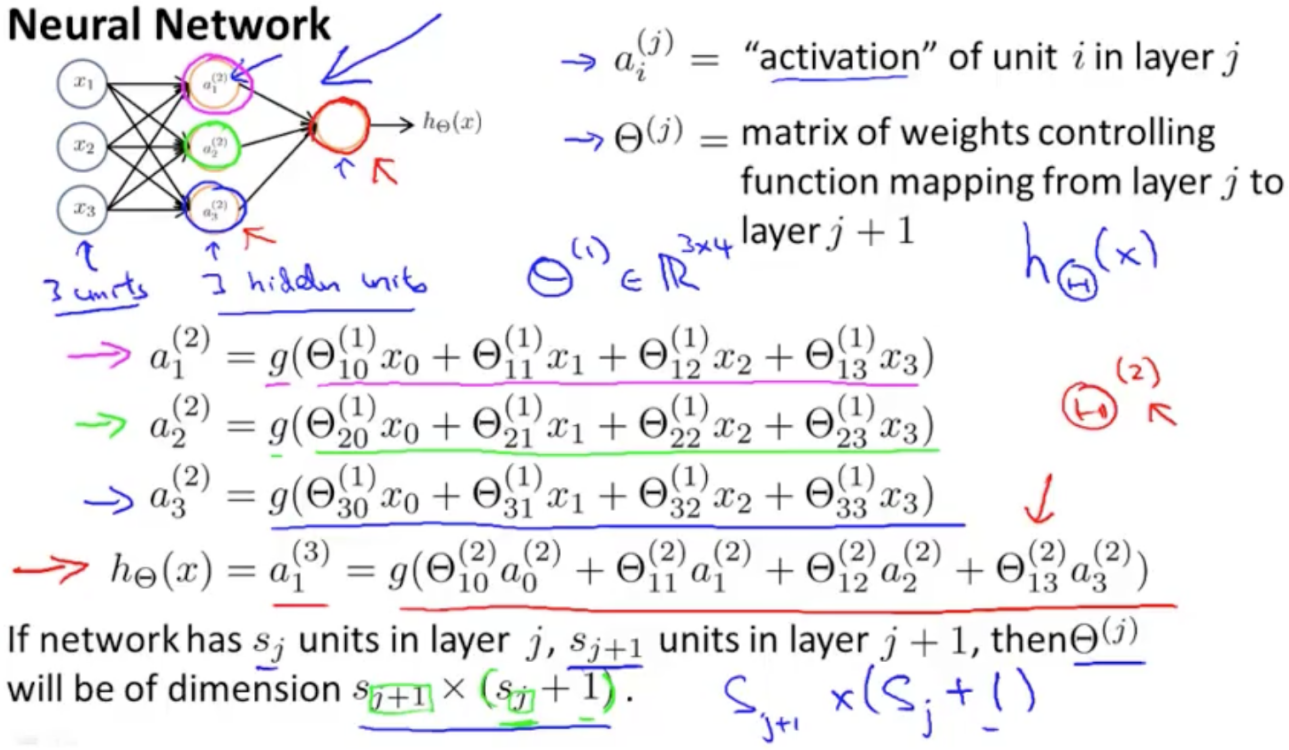
Activation Function

There are different types of activation functions.



.

Most of those functions have “mathematics convience”, which simplifies the downstream equation calculation.



Artificial Neural Network (https://en.wikipedia.org/wiki/Artificial_neural_network)
Deep Learning (https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks)

Cost function

i - index for training data m
sl - number of units in layer l
L - total number of layers
l - index for layer L
n - number of features
m - number of test data
K - number of output/class

for each test data
find cost for each class
+
for each layer
calculate THETA of each node

Logistic regression:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Neural network:

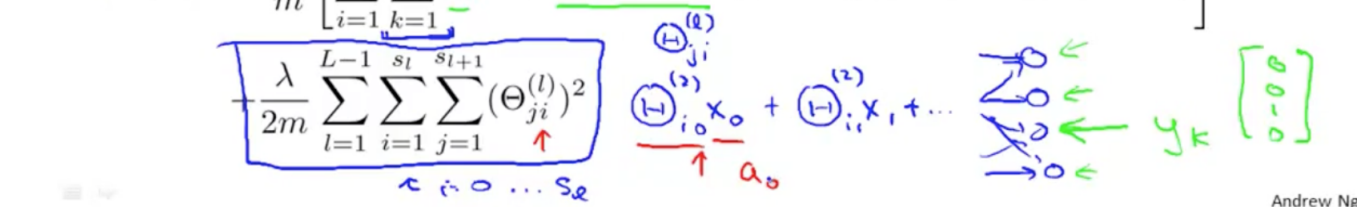
THETA is next layer sl x previous layer sl + 1 (bias)

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

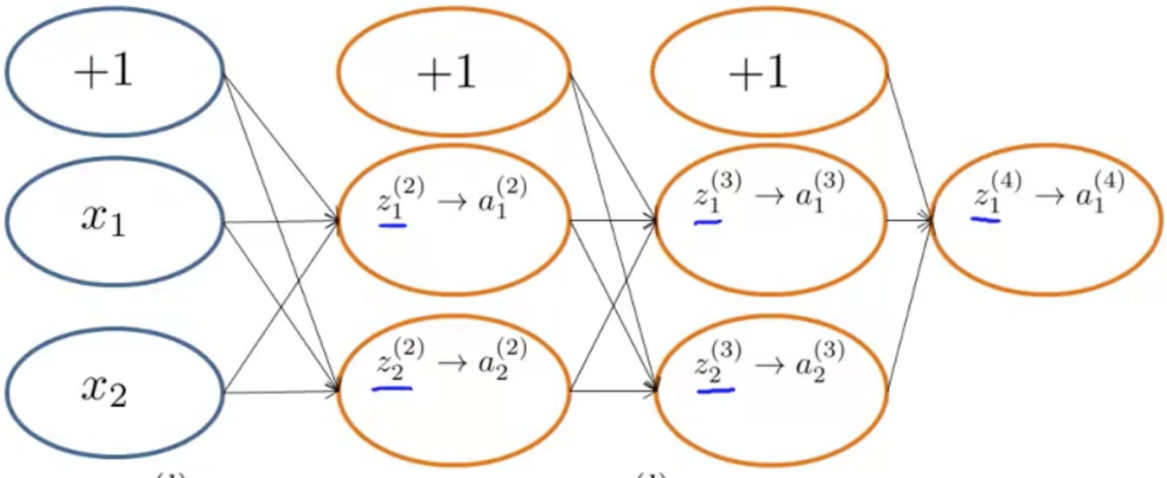
$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log((h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

$i = 0 \ldots s_l$



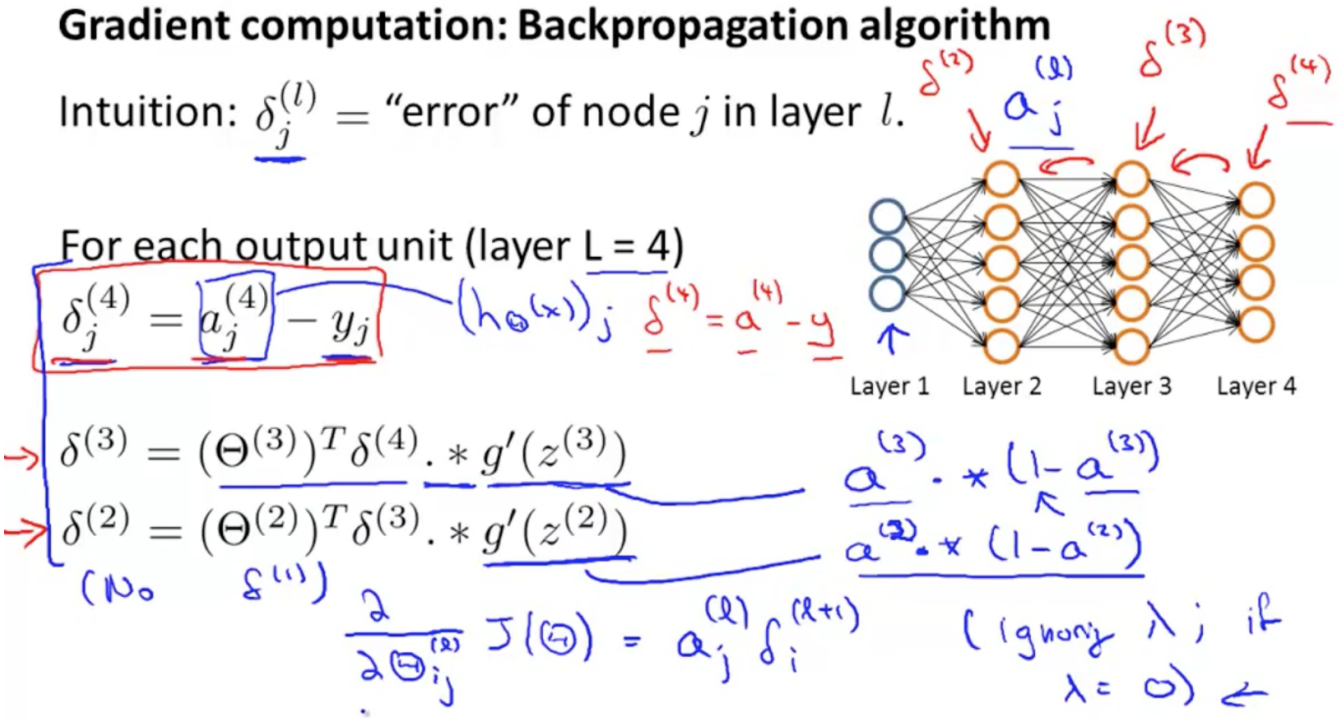
Forward Propagation



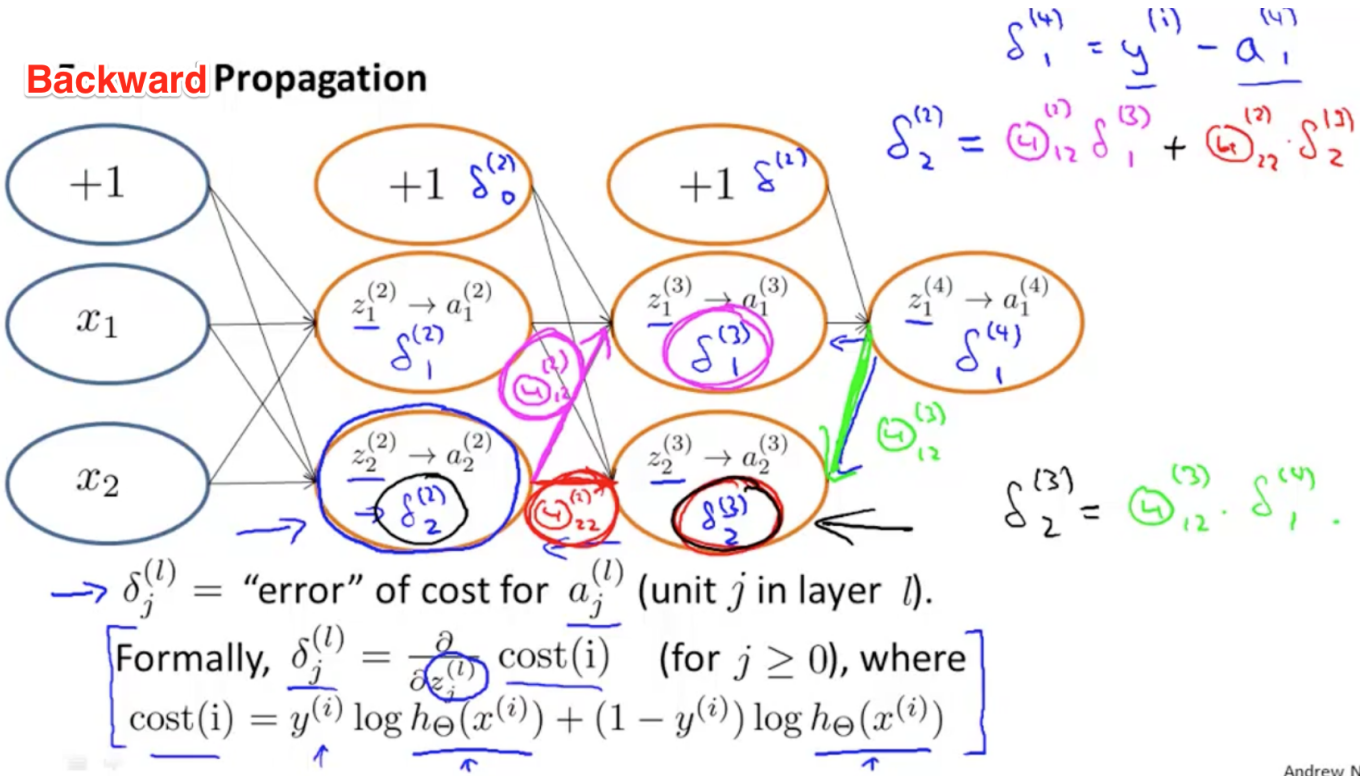
$\delta_j^{(l)}$ = "error" of cost for $a_j^{(l)}$ (unit j in layer l).

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost(i)}$ (for $j \geq 0$), where

$$\text{cost(i)} = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})$$

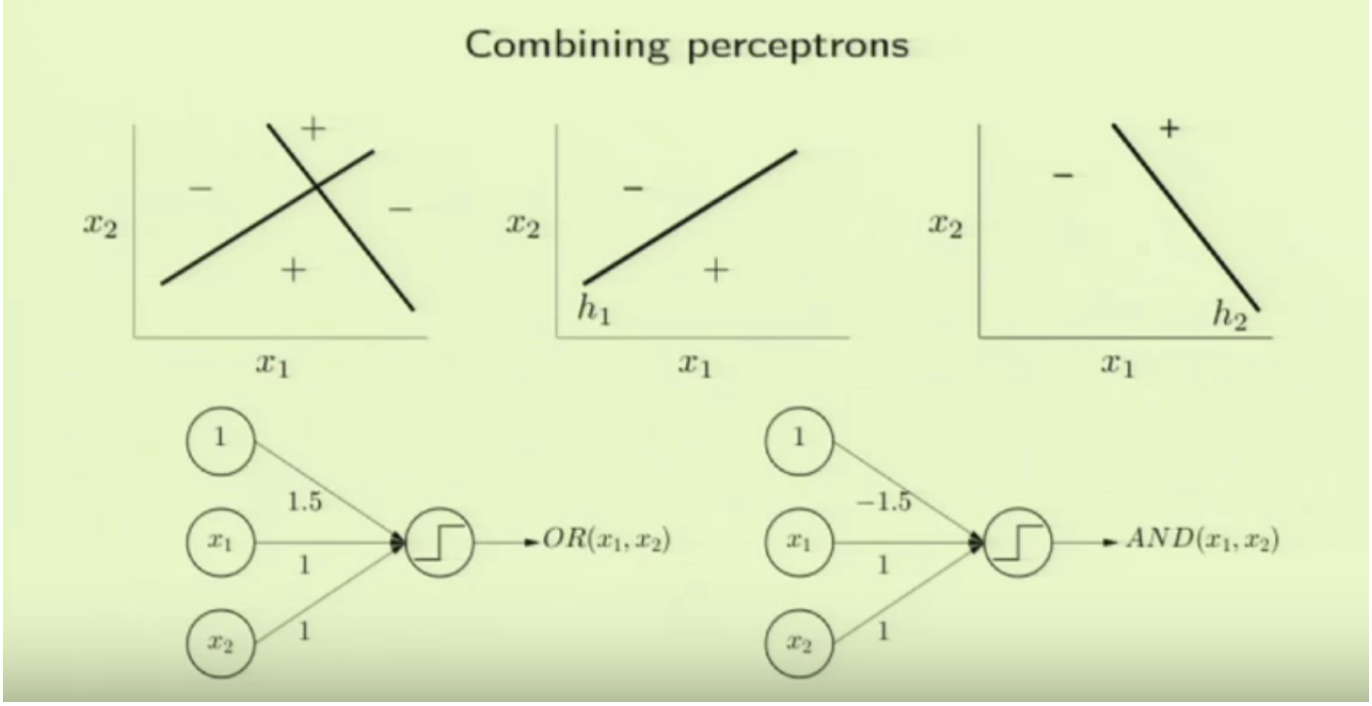


See MIT 6.034 lecture-12 for derivation of gradient descent formula; a3 .* (1 - a3)

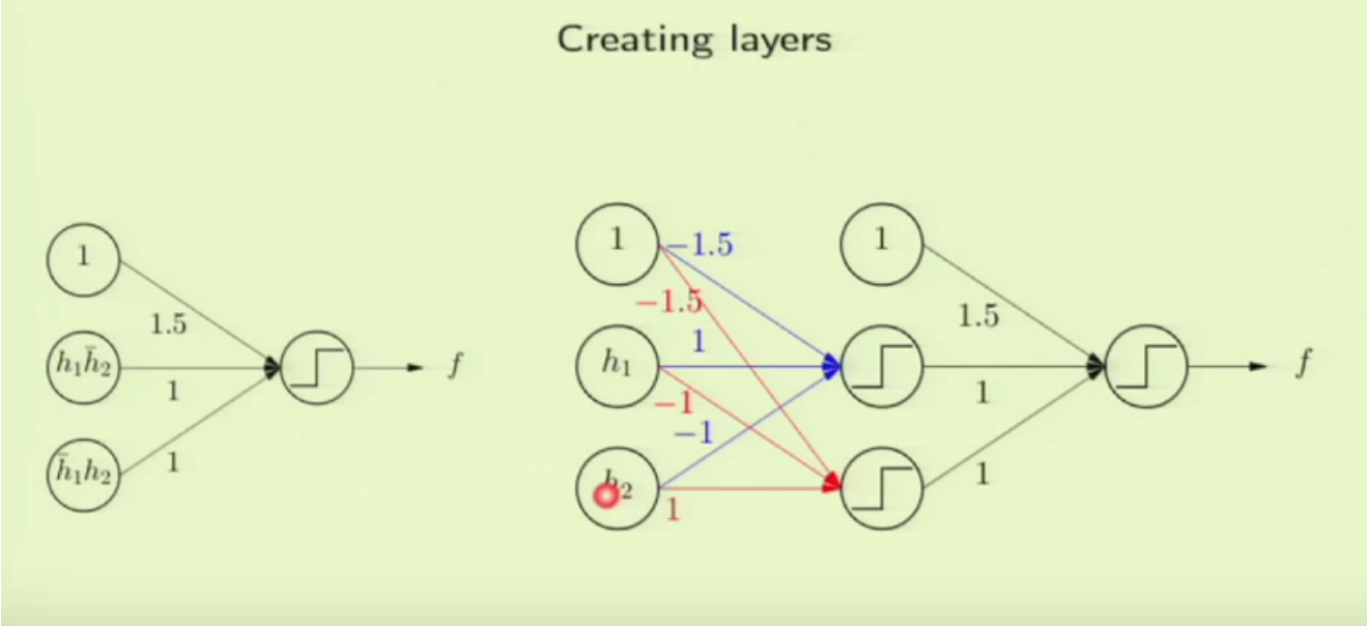


More detailed computation steps.

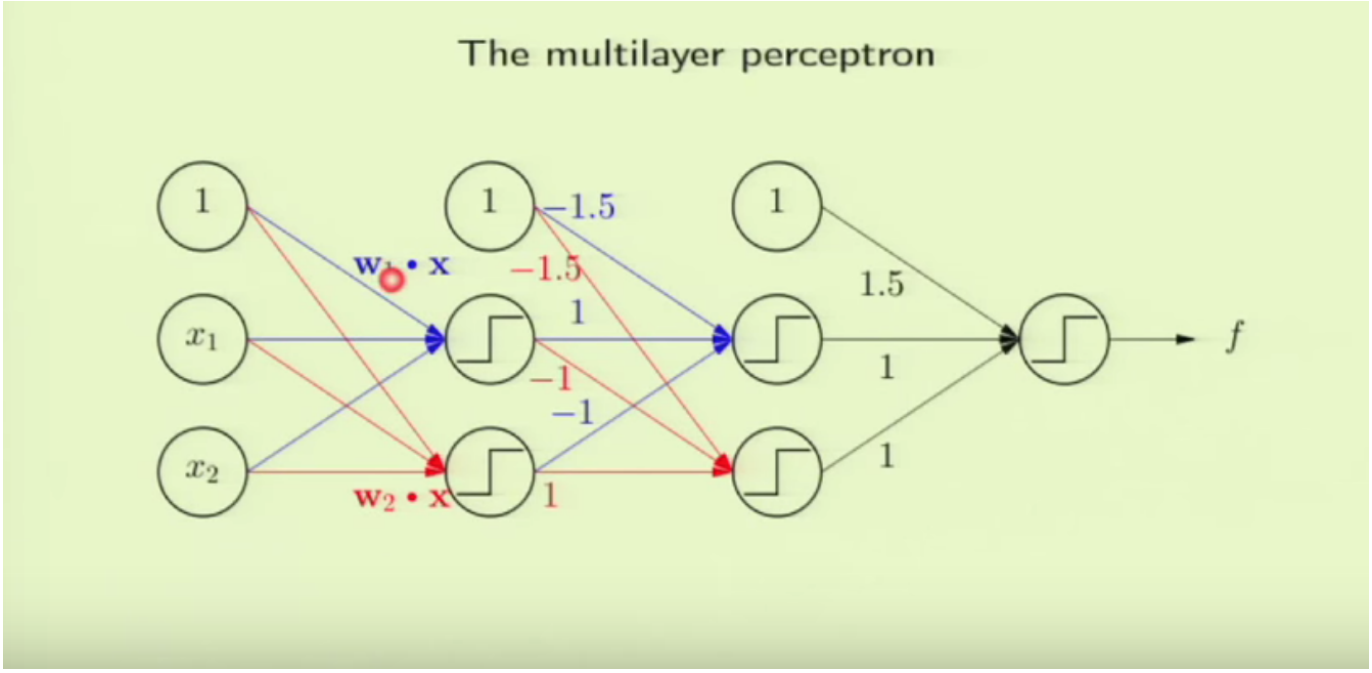
Caltech CS 156 Machine Learning - Learning from Data lecture-10 (http://work.caltech.edu/telecourse.html)



One simple logistic regression can not separate the testing data. There are many different activation functions can be used here, tanh(s) is here. See selection approach (http://stats.stackexchange.com/questions/101560/tanh-activation-function-vs-sigmoid-activation-function)

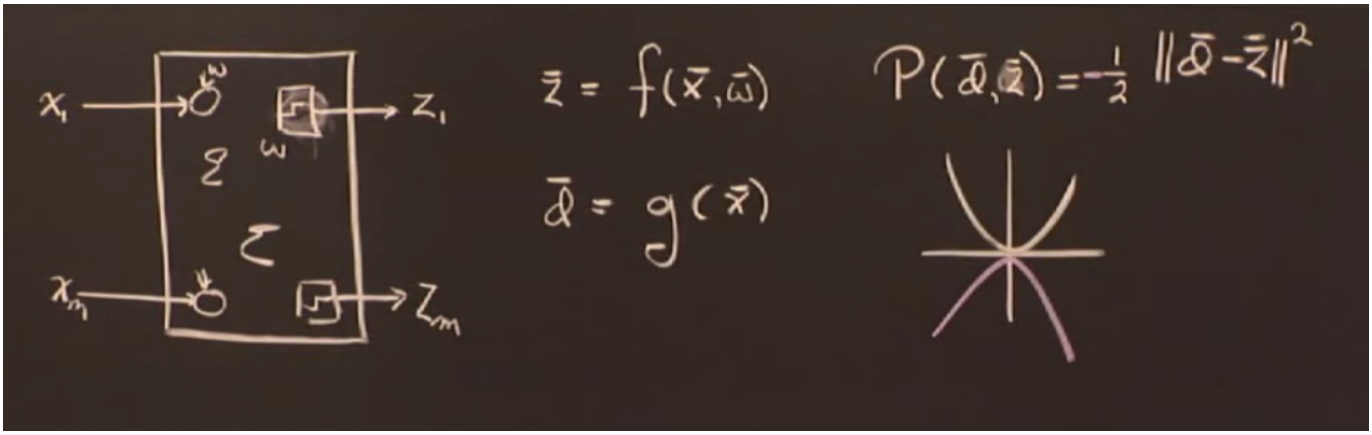


We have to use two separate nodes to cover the problem space.

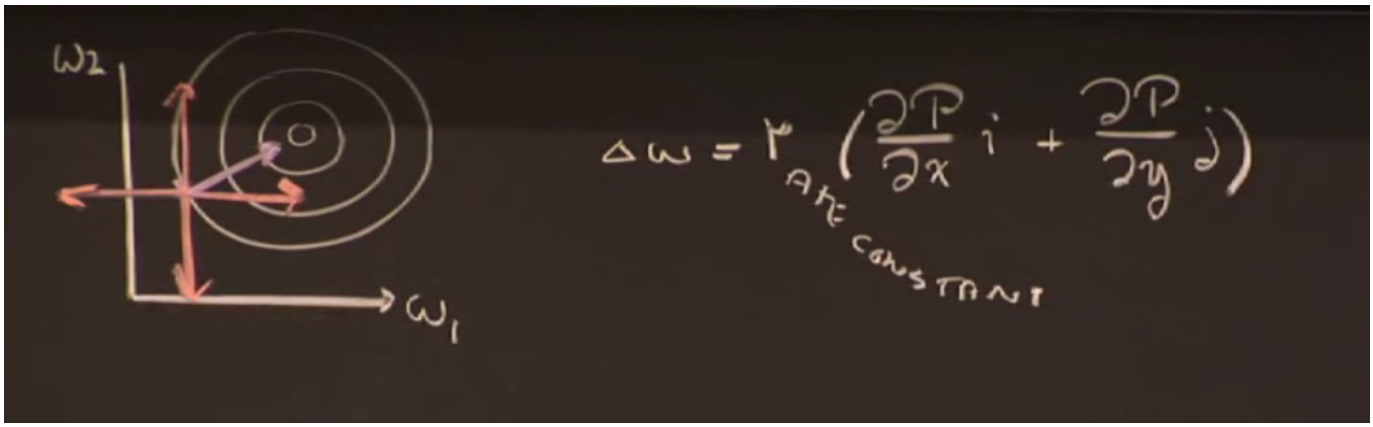


This is two features (n=2), two hiden layers (L=2), one classification (K=1)

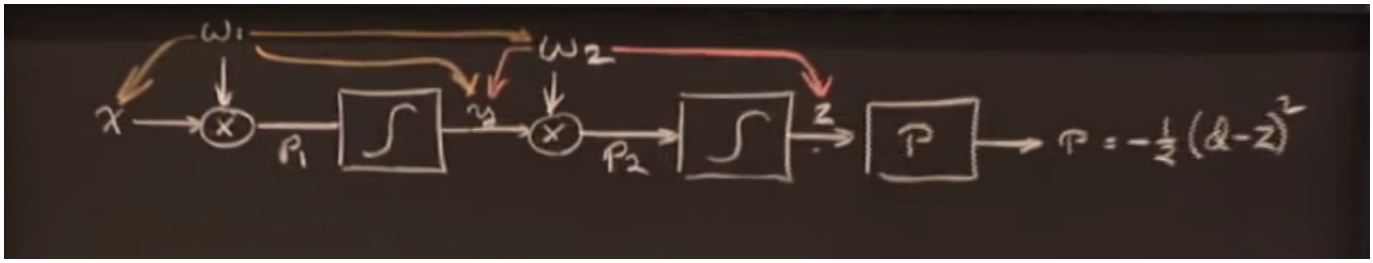
MIT Course Number 6.034 lecture-12 (https://www.youtube.com/watch?v=q0pm3BrIUFo)



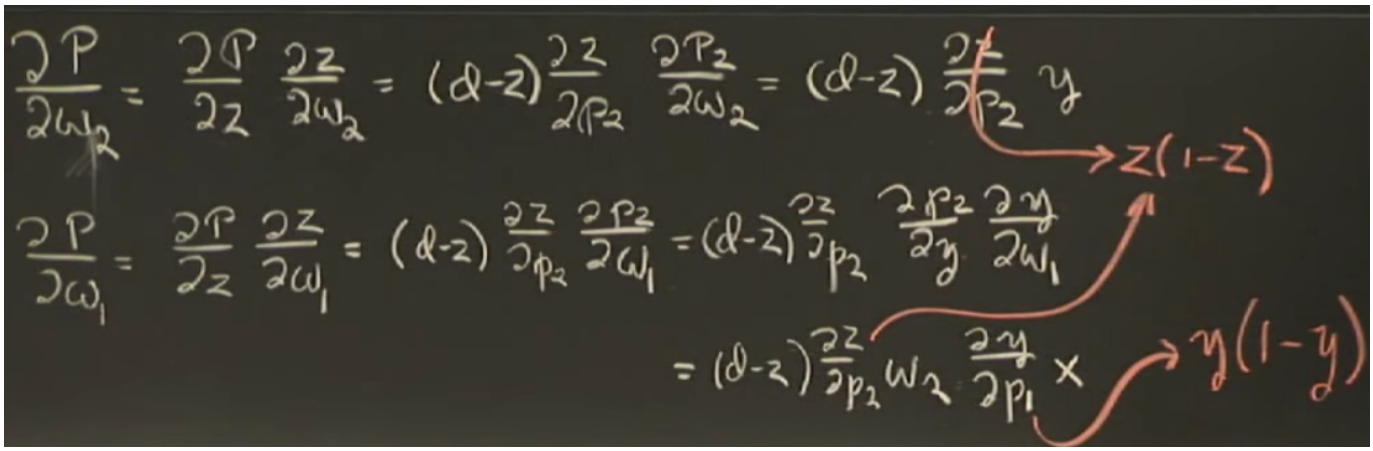
x1,…xn is input,
z1,…zn is outpu, which equalvent to y1, …yn in prof. Ng's lecture, and (x1, z1) is a pair.
d is y hat, it is the value calculated based on hypothesis.
P is performance, is a cost function.
y is a2 in prof. Ng's notes.



There is a typo in picture, x and y should be w1 and w2.



w1 is input layer, consider x is a single variable or a vector.
w2 is hiden layer,
z is output layer,
p is error, cost function.
This model is set for proving backpropagation.



This exercise proves the performance improvement is local dependency, e.g. for $$$\partial(p/w2)$$$ is dependent on (d-z), y, and $$$\partial(z/p2)$$$.

$$\partial(z/p2) = z*(1-z)$$

Use Cases

- equipment failure prediction

- facial recognition
- speech recognition
- text classification
- self-driving car
- smart home
- surveillance and security
- medical image and diagnostic
- spam discovery and filtering
- predictive maintenance
- …

Versions

A study note about Learning Machine Learning,

- v.0.0.1, April-10 2016, initial version, Richard Kuo, at La Boulanger, Mountain View, CA
- v.0.0.2, June-20 2016, add history and update mindmap, Richard Kuo, at La Boulanger, Mountain View, CA