Machine Learning and Deep Learning

Abhishek Sharma, Sujit Pal
abhisheksharma@salesforce.com

Outline

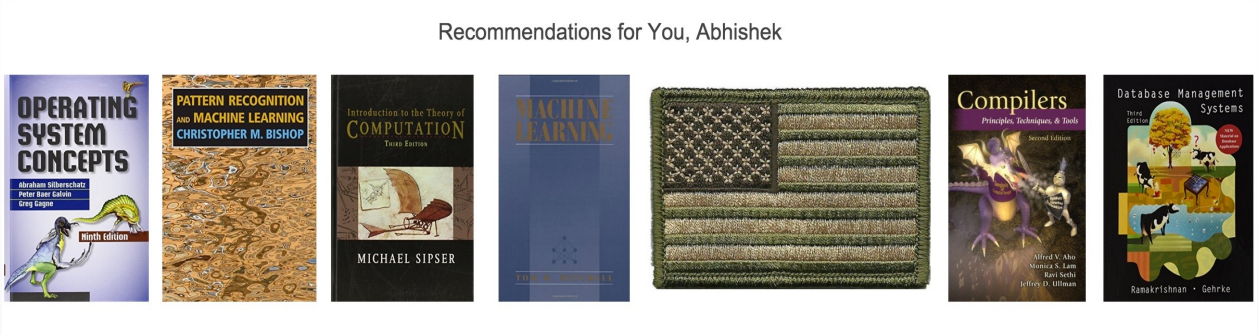
- Artificial Intelligence
- Numpy
- ML vs DL
- Machine Learning
  - Example – Character recognition
  - Another Example
  - General Flow
  - Scikit- Learn
  - Examples
  - Need for Deep Learning
- Deep Learning
  - Intro
  - Tensorflow
- Learn more ?
- References

Artificial Intelligence

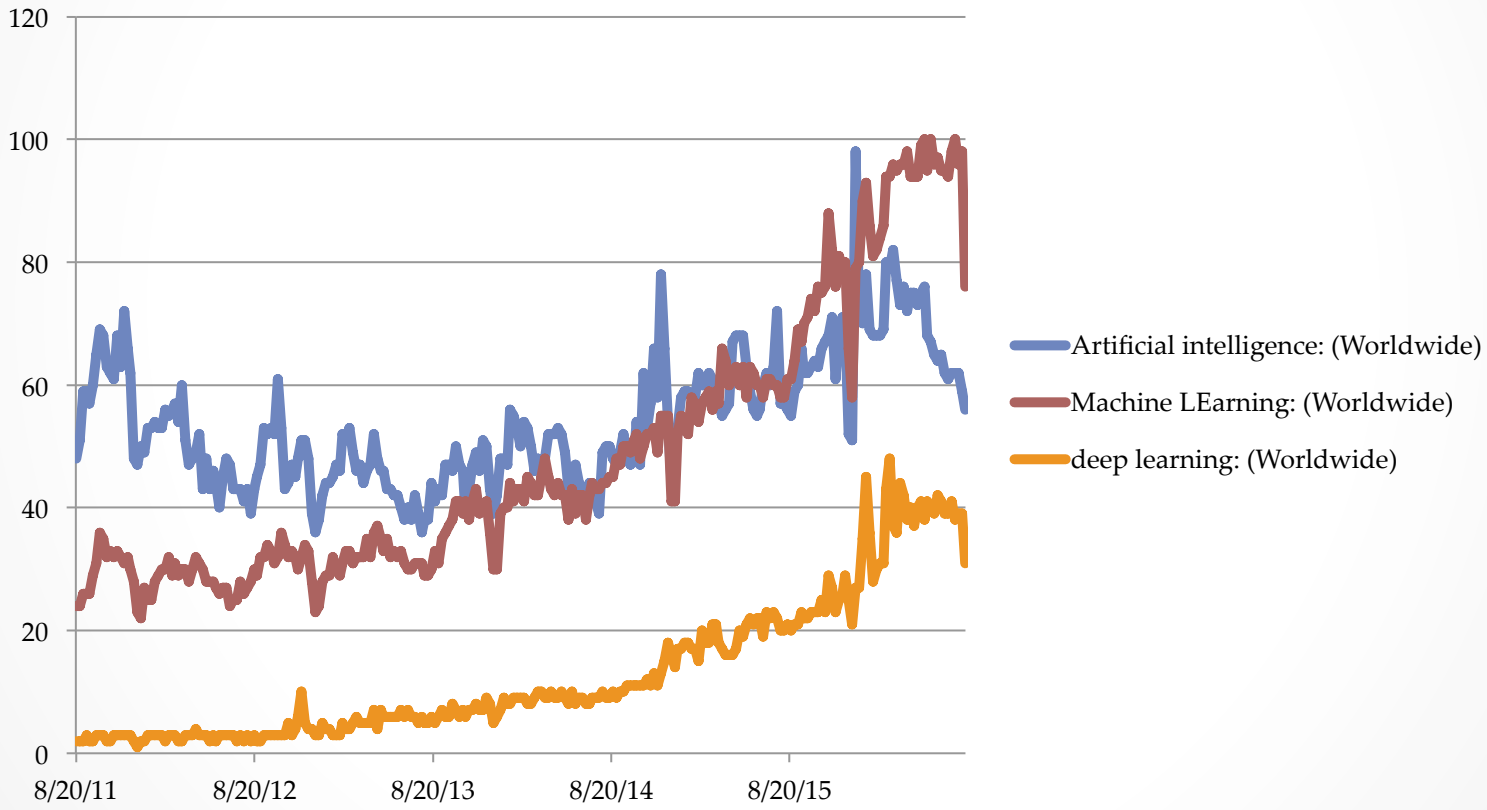
- “ [The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..."(Bellman, 1978)
- "A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1 990)
- “The capability of a machine to imitate intelligent human behavior”. (merriam-webster)

Artificial Intelligence

- Examples:
  - Movie/ Songs recommendation
  - Fraud Detection – credit card purchases
  - Smart home devices
  - Facebook – tagging friends
  - Video Games
  - Virtual Personal Assistants: Siri, Cortana

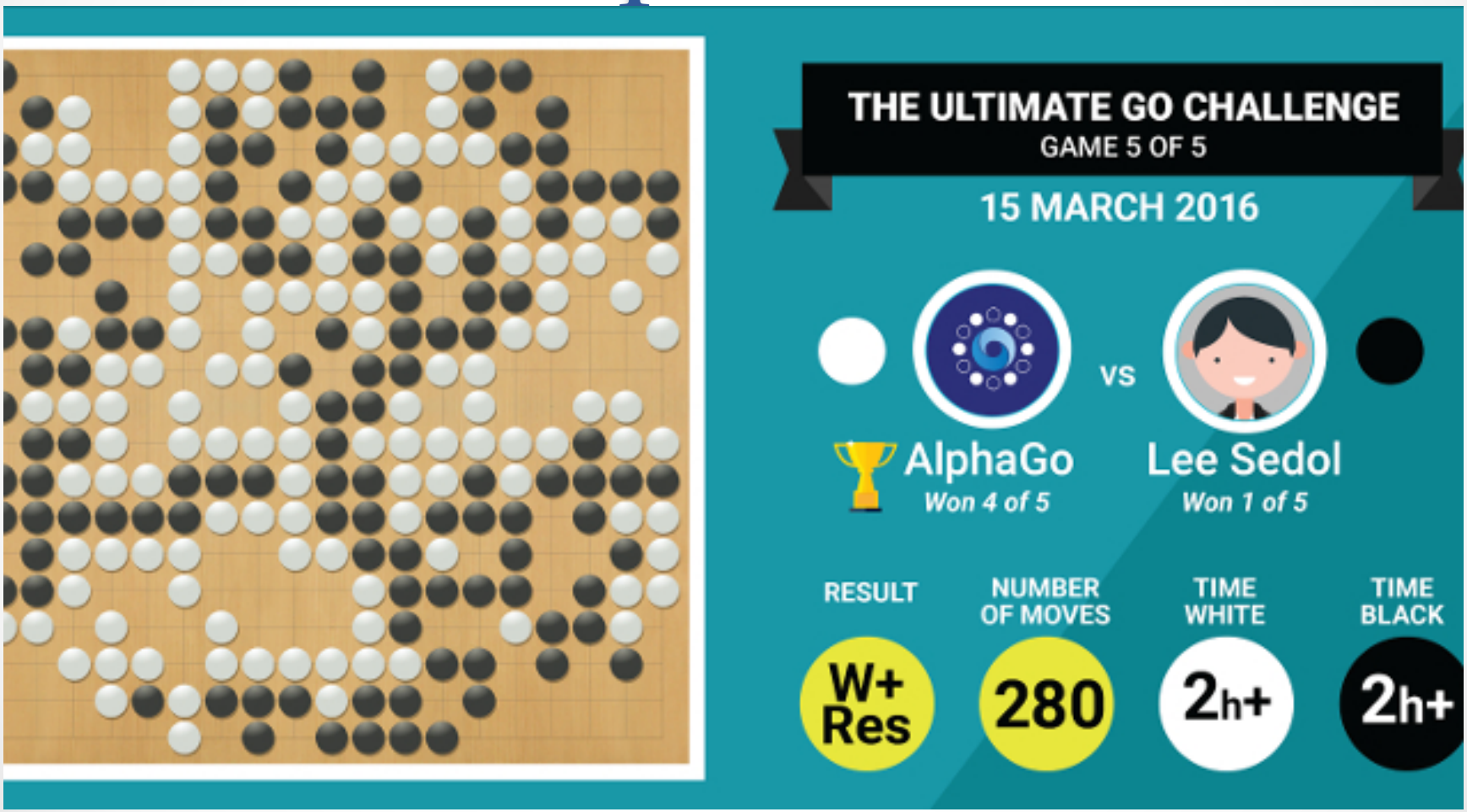


AI vs Machine Learning vs Deep Learning



Google trends

AlphaGo


THE ULTIMATE GO CHALLENGE
GAME 5 OF 5
15 MARCH 2016
VS
AlphaGo
Won 4 of 5
Lee Sedol
Won 1 of 5
RESULT
W+ Res
NUMBER OF MOVES
280
TIME WHITE
2h+
TIME BLACK
2h+

Libratus AI


BRAINS VS.
ARTIFICIAL
INTELLIGENCE
RE-MATCH!
JANUARY 11 - 30
11AM - 7PM • POKER ROOM
PARTNERS
RIVERS
Be sure to Tweet @WinBigRivers and
#SCSatCMU using #BrainsvsAI
PROBLEM? CALL 1-800-GAMBLER.
IT'S BACK!
GAMBLING

Machine Learning

- Algorithms that do the learning without human intervention.
- Learning is done based on examples (aka dataset).
- Goal:
  - learning function *f: x* ➔ *y* to make correct prediction for new input data
    - Choose function family (logistic regression, support vector machines)
    - Optimize parameters on training data: Minimize Loss $\sum(f(x) - y)^2$

Supervised Learning

- Supervised Learning:
  - Data: (x,y)
    - X is data, y is label
  - Goal: learn a function to map x => y
  - Example: classification, regression, image classification

Unsupervised Learning

- Unsupervised Learning:
  - Data: x
    - Just data
  - Goal:
    - Learn some structure of the data
  - Example:
    - Clustering
    - Feature learning

Classification

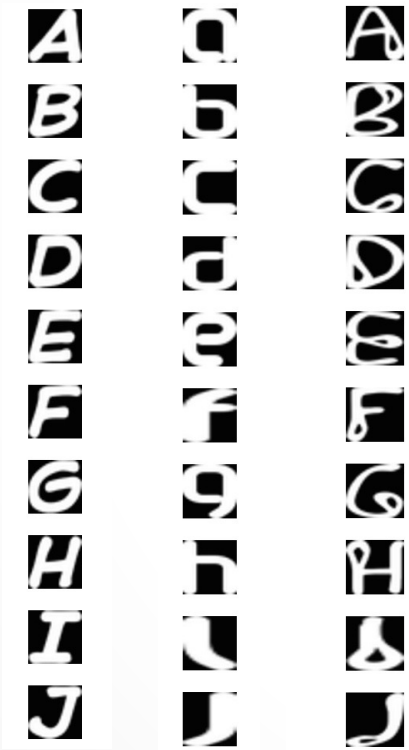
- Predict if you are present in a picture (tagging people)
- Predict model of the car
- Which category does an article belongs to: sports, politics, health etc

Numpy

- https://github.com/abhi21/intro_to_ml_dl

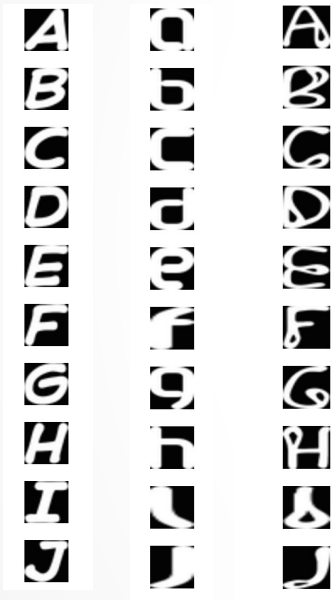
Character Recognition

- Dataset: notMNIST
  - collection of extracted glyphs from publicly available fonts



notMNIST dataset: http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html

Character Recognition



prepare model; learn weights

```
function = LogisticRegression()
features_testset =
test_dataset.reshape(test_dataset.shape[0], 28 * 28)
labels_testset = test_labels

feature_trainingset =
train_dataset[:sample_size].reshape(sample_size, 28*28)
labels_traininigset = train_labels[:sample_size]
function.fit(feature_trainingset, labels_traininigset)
```

prediction/ inference

```
function.score(features_testset, labels_testset)
```

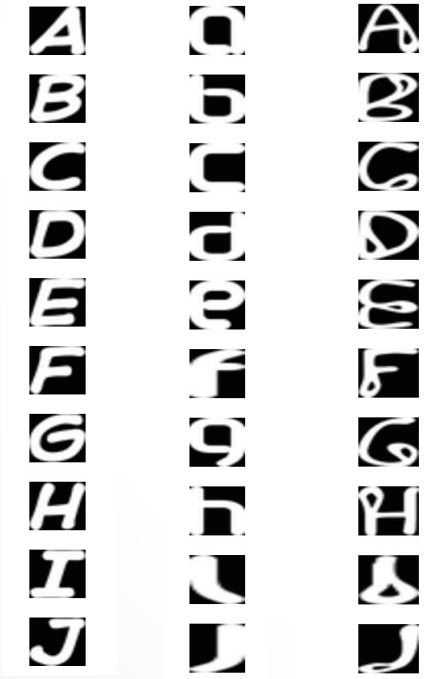
evaluation

Accuracy: 0.856199

*Library: Scikit-learn

Behind the scenes:

- Training phase:
  - Feature Extraction
  - Data Normalization



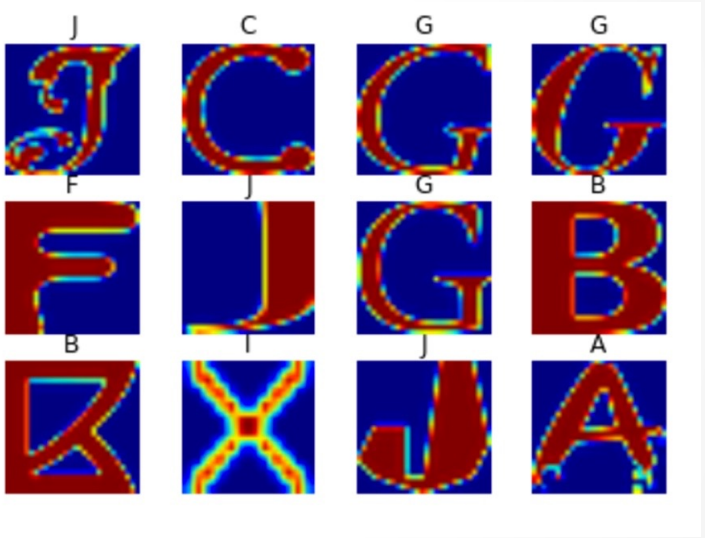
Extract features; normalize data

0	0	0	0	1	1	1	1
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0
1	0	0	0	1	0	0	1
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0
1	0	0	0	1	0	0	1
1	0	0	0	1	0	0	1
1	1	1	1	0	1	0	0
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0
1	0	0	0	1	0	0	1
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0

Behind the scenes:

0	0	0	0	1	1	1	1
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0
1	0	0	0	1	0	0	1
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0
1	0	0	0	1	0	0	1
1	0	0	0	1	0	0	1
1	1	1	1	0	1	0	0
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0
1	0	0	0	1	0	0	1
1	1	1	1	0	1	0	0
1	0	0	1	0	1	0	0

is actually

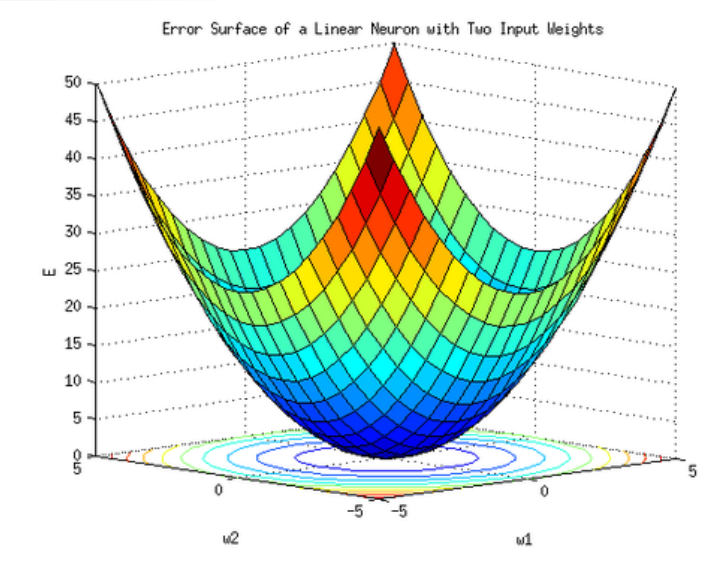


Saved as pickle (python; serializes objects);
plotted used pyplot

Behind the scenes:

- Training phase:
  - Model
    - Learn weights/ parameters based on training data
  - Goal: Minimize loss
    - Function: Logistic Regression (aka logit)
      - $y = w^T x + b$
        - w: weights ; x: features ; b: bias terms used for regularization
        - y: predicted value
    - Minimize: gradient descent approach/ stochastic gradient descent
      - Iterative process

Gradient Descent



Picture: en.wikipedia.org/Creative Commons

$$L(w) = \sum_{i=1}^{n} L(w_i)$$

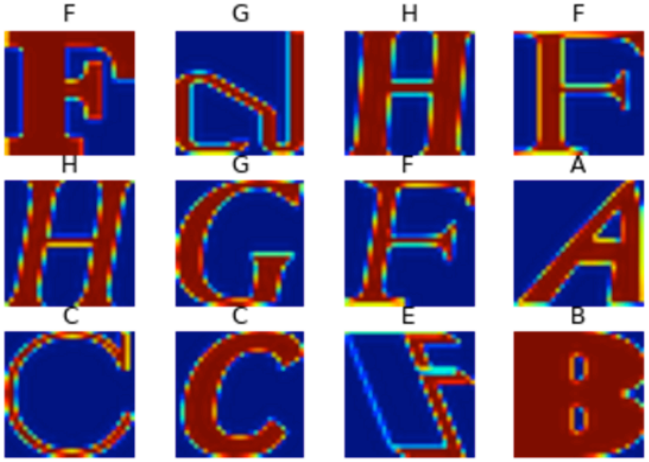
$$w := w - \eta \frac{\partial L(w)}{\partial w}$$

- Assume locally convex Loss function.
- Initialize weight vector W with random values.
- Compute L(w)
- Randomly change one (or some, or all values of W)
- Compute new L(w) and difference.
- Update W with the difference multiplied by learning rate η .
- Repeat until W converges or $\nabla L(w) \to 0$

Behind the scenes:

- Inference phase:
  - Run the model against test data
  - predicting how the model performs on new input data (data which it did not encounter during training phase)

  - Accuracy: 0.856199



Another example

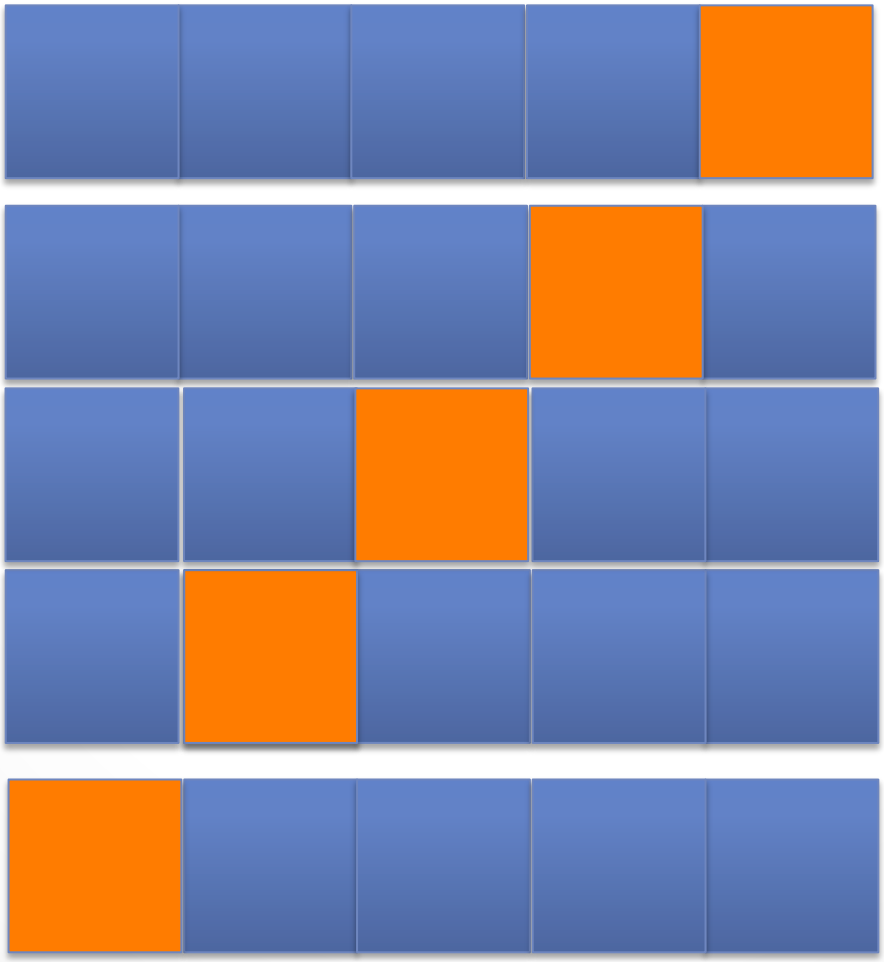
- Classify email as spam or not
  - https://github.com/abhi21/Spam-Classifier-NB

Evaluating performance

- Cross Validation
  - *Partition training data into k folds.*
    - *Use one of the fold for testing and the remaining folds for training*
    - *Repeat for all folds*
    - *Evaluate performance by testing against the Validation set*

Evaluating performance

- Cross Validation



Evaluating performance

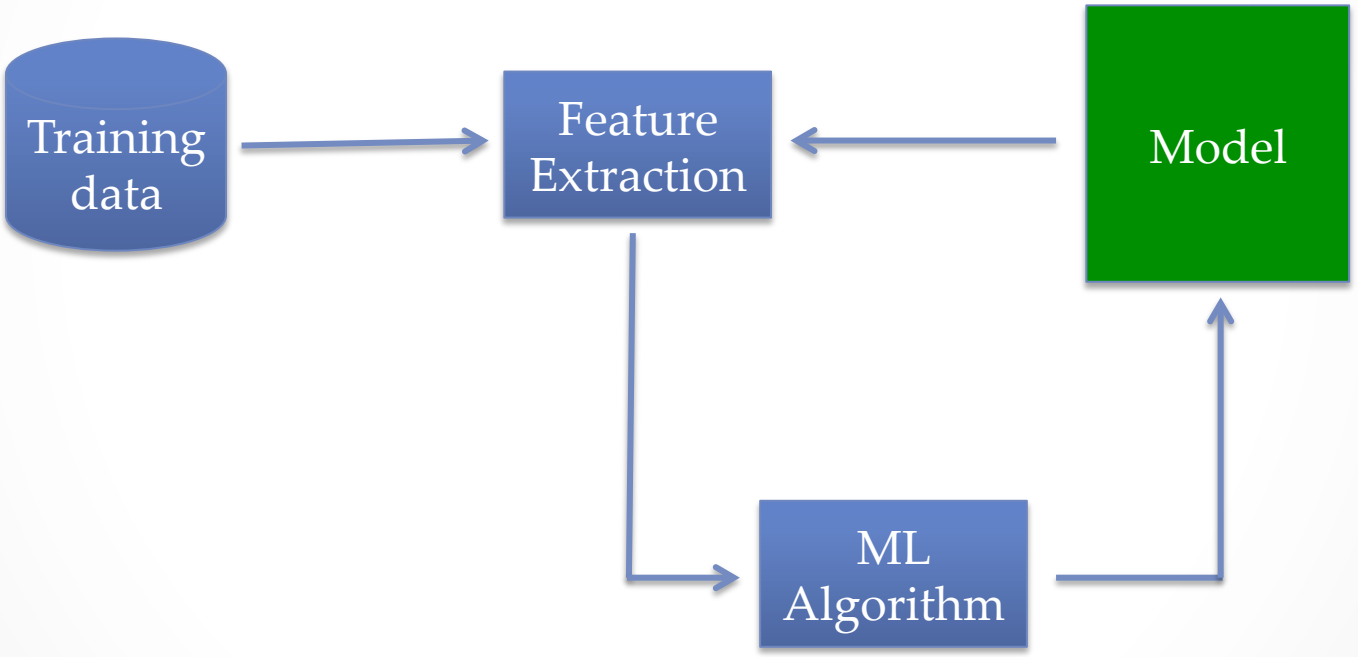
- Bias
  - *Bias of estimator B = E[B] – B'*
    - *E[B] => expected value*
    - *B' => actual value*

Evaluating performance

- Variance
  - Measure of variability of predictions if the learning process is repeated multiple times with fluctuations in training set.
    - ***Variance = E[(B- E[B])^2]***

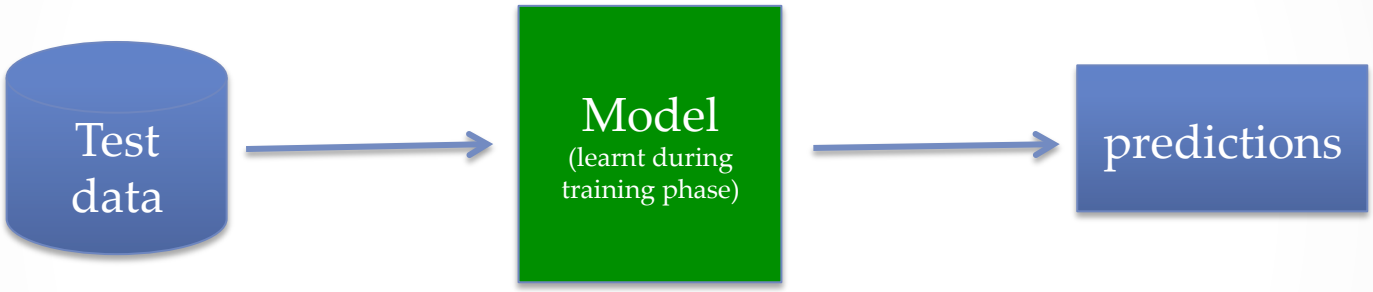
General ML approach

- Training Phase:



General ML approach

- Testing Phase:

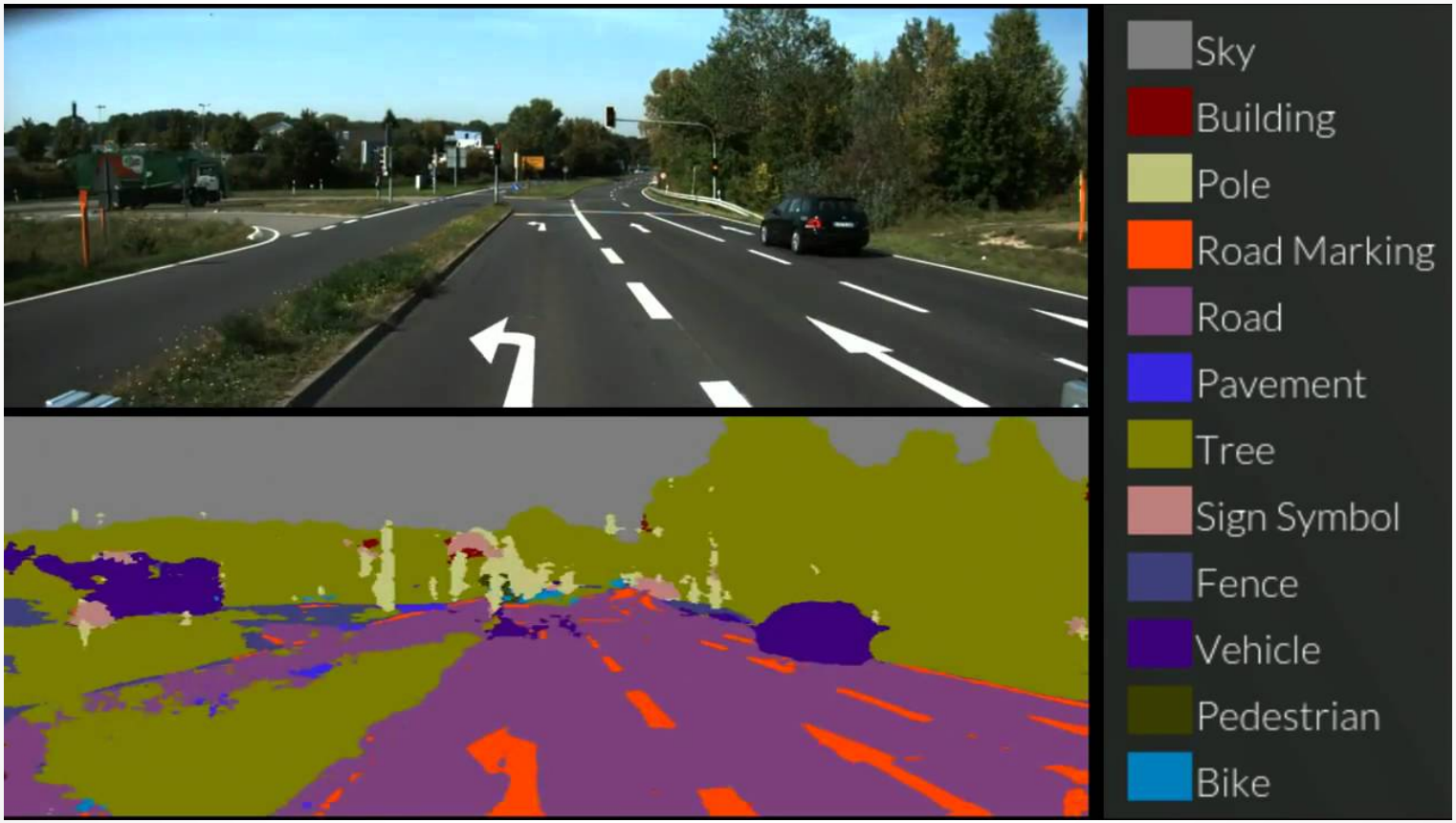


Deep Learning

- ?

AI vs Machine Learning vs Deep Learning

Self Driving Cars

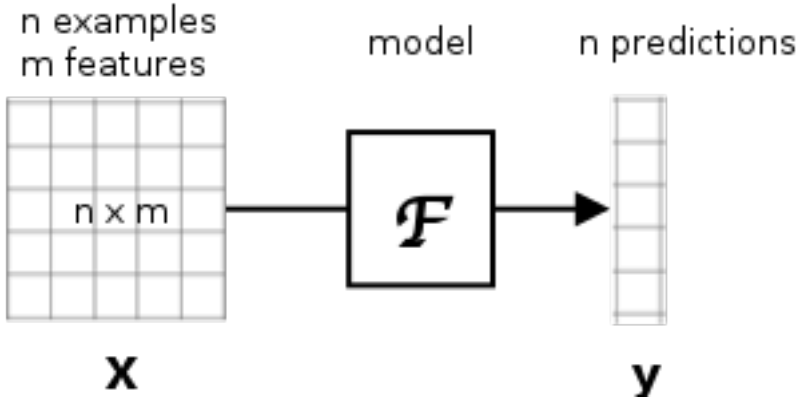


Source: https://www.youtube.com/watch?v=kMMbW96nMW8

Need for Deep Learning

- Pros
  - Automatic Feature Selection.
  - Ability to re-use basic building blocks to compose models tailored to different applications.
- Cons
  - Tendency to over fit.
  - Requires lots of data.

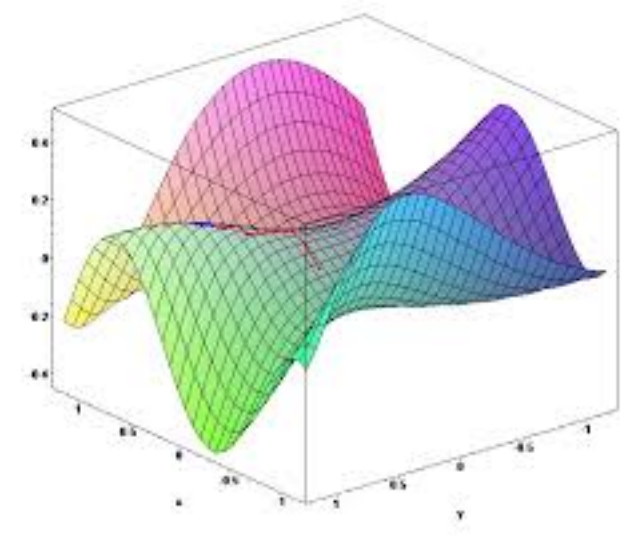
Motivating Example

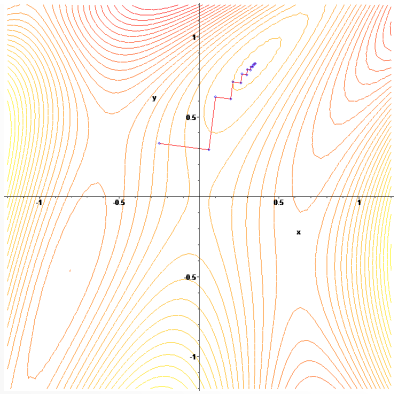


- Typical Classification Problem – given n training examples each with m features, we need to find a mapping **F** that produces the "best" predictions $\hat{y}$.
- "Best" == minimize difference between $\hat{y}$ and label y.
- Loss function quantifies notion of "best".

$$L = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Loss Function





- Visualization of Loss function for a system with 2 features.
- Convex functions can be solved analytically.
- Real life functions rarely convex.
- Gradient Descent is a numerical optimization method to find optimal points in the space.
- Guaranteed to find the global optima for convex functions.
- Guaranteed to find local optima for non-convex functions.
- Local optima usually "good enough" for high dimensional space.

Pictures: en.wikipedia.org/Creative Commons

Linear Model

- Matrix Formulation:

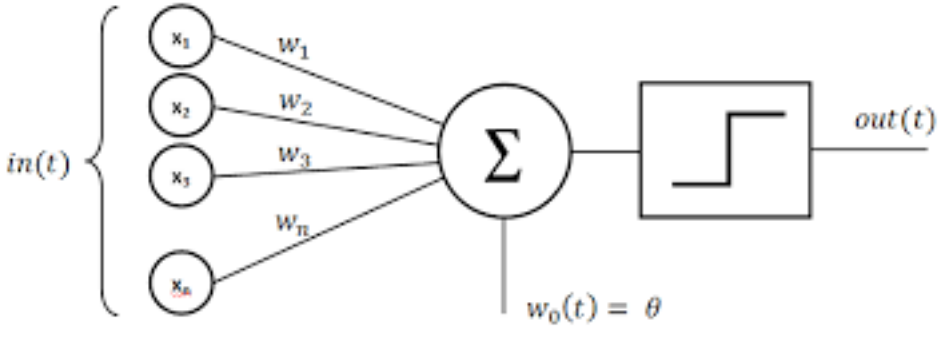
$$y = XW + b$$

where:

- y is a (n, 1) vector of predicted values.
- X is the (n, m) input matrix
- W is the (m, 1) weight vector – corresponds to the final coefficients of the model.
- b is a constant – corresponds to the intercept of the model.

Perceptron

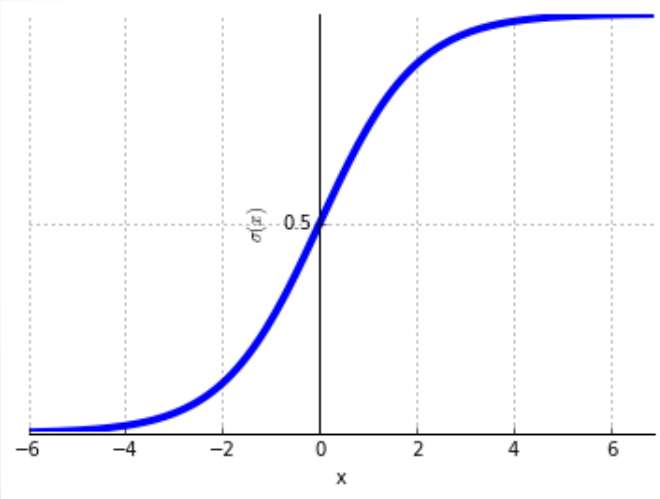
- The Perceptron is an early Neural Network model.
- Each Perceptron takes multiple inputs and outputs a binary value (0 or 1).
- Uses step function to compute binary output.
- Network of Connected Perceptrons make up model.



Picture: commons.wikimedia.org/Creative Commons

Sigmoid Neurons

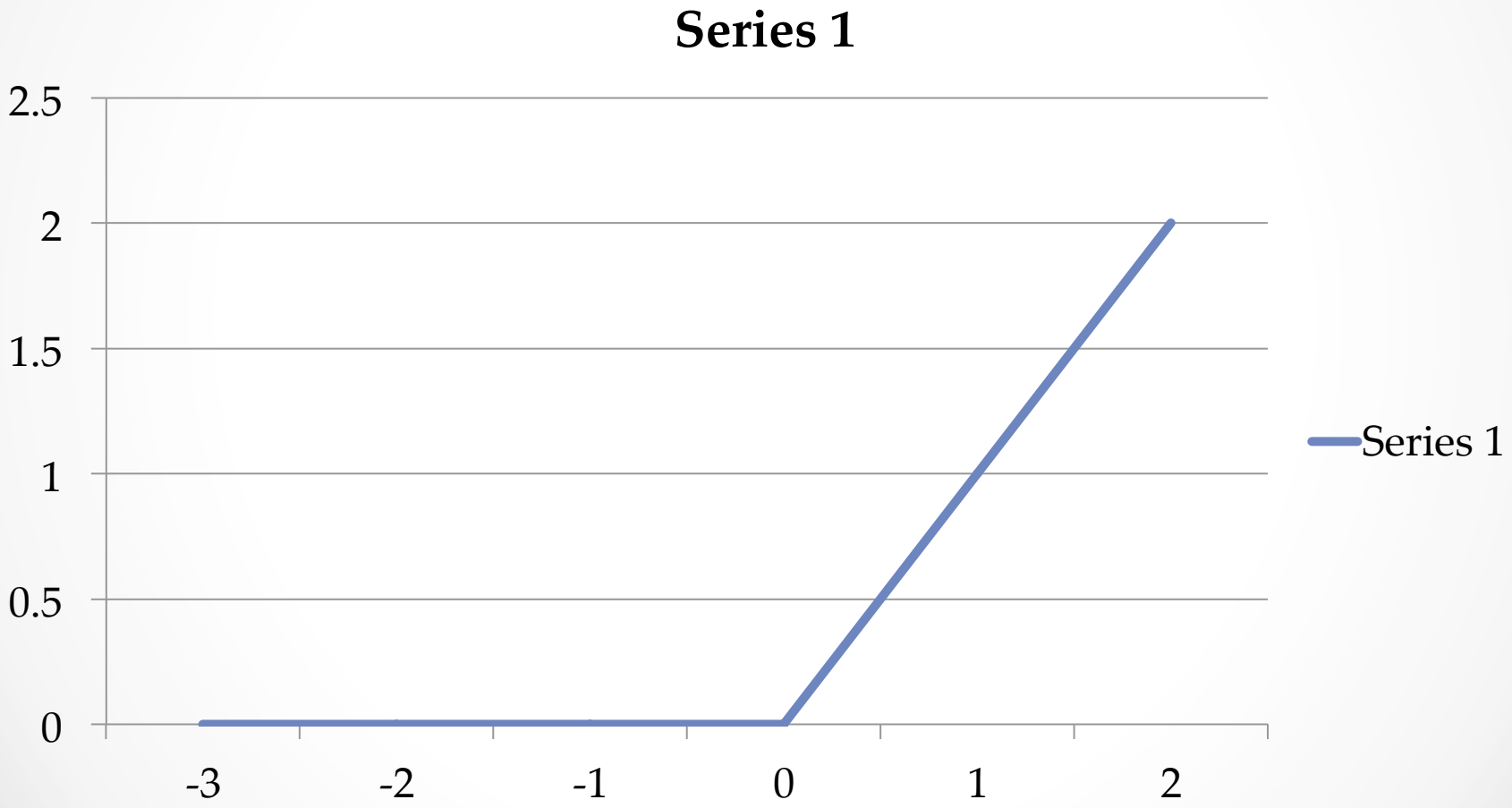
- Similar to Perceptron neurons, but can output a continuous value between 0 and 1.
- Output is differentiable, so back-propagation is possible.
- Called Activation Function in the literature.
- Other Activation Functions in use now as well.



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

$$\hat{y} = \sigma(XW + b)$$

RELUs

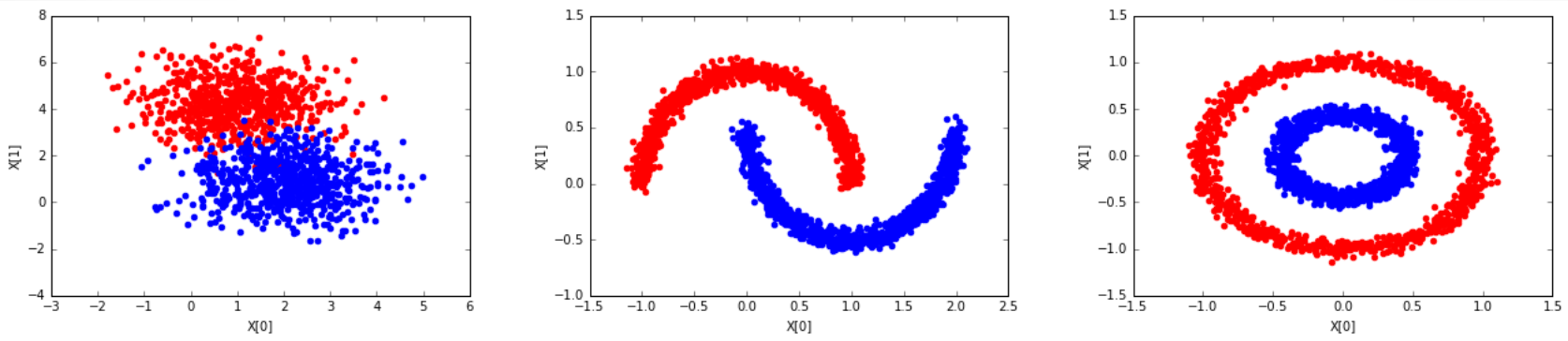


Popular Toolkits

- Comprehensive List in the PyImageSearch blog My Top 9 Favorite Python Deep Learning Libraries from Adrian Rosenbrock.
  - Caffe – written in C++, network architectures defined using JSON, has Python interface for training and running networks. Very mature and has many prebuilt models.
  - Keras – Python wrapper that exposes very minimal and easy to use interface. Useful if you are composing your architectures from existing models.
  - Tensorflow – written in C++ but Python API gives you full control over the network. Useful if you are building new architectures (or those not available already)

Handling Non Linear Data

- Demo - universality of Deep Learning techniques.
- Build and run progressively more complex networks to handle datasets like these.

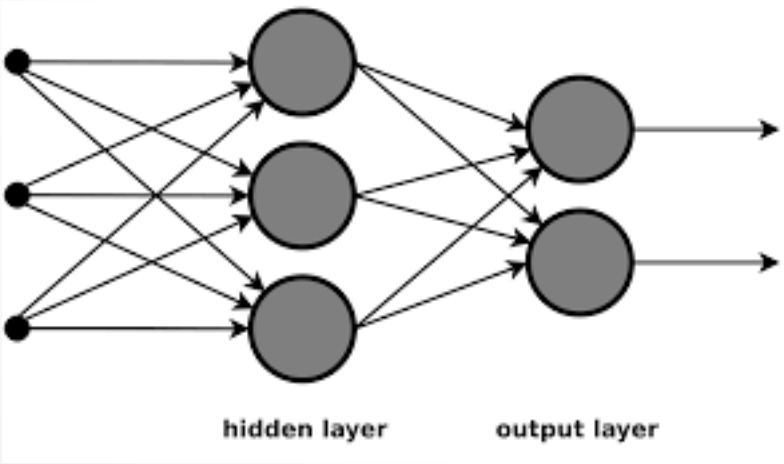


Building Blocks

- Fully Connected Networks
  - First network model to become popular in Deep Learning.
  - Has been applied to classification, generating embeddings for natural language vocabularies, etc.
- Convolutional Neural Networks
  - Exploits the geometry of the input.
  - Applied to computer vision (image and video), classification and natural language processing.
- Recurrent Neural Networks
  - Exploits the temporal nature of the input.
  - Applied to classification, audio and speech applications, natural language processing, machine translation, image captioning, etc.

Fully Connected Networks

- Inspired by biology – axons.
- Every node in a layer is connected to every node in the layer following it.
- Input to a layer is output of previous layer.
- Weight vector shared within each layer.
- Gradient of loss propagate backward to update weights in each layer.



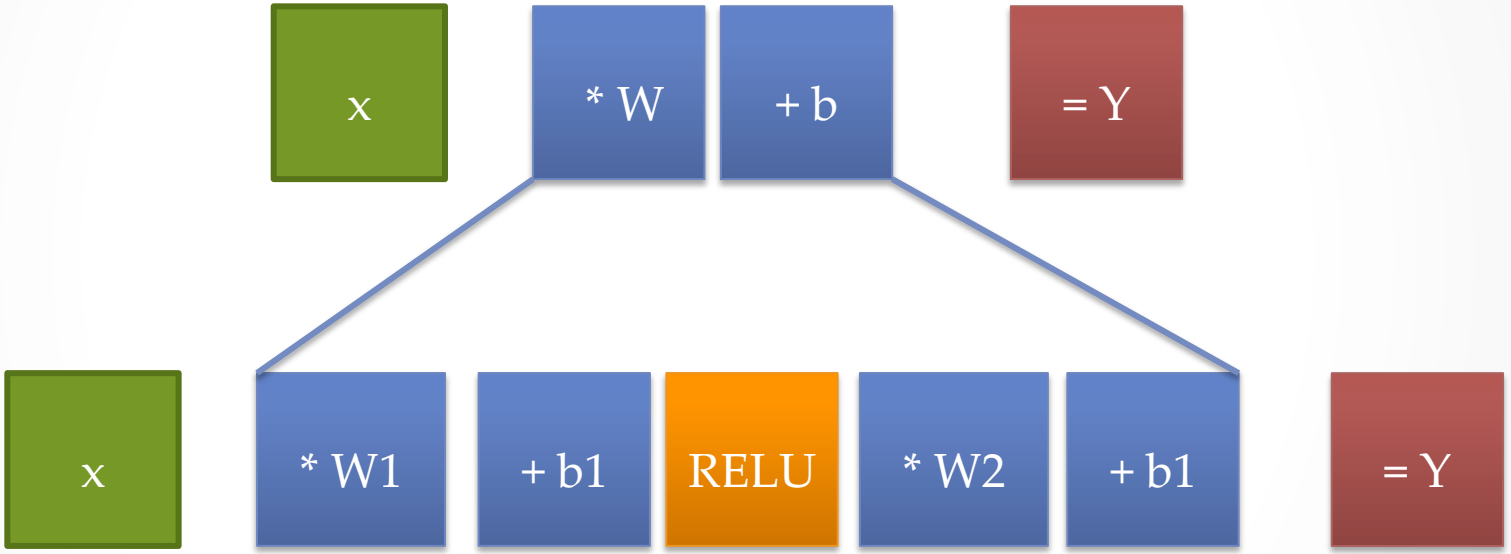
$$X^L = \sigma(W^{L-1}X^{L-1} + b^{L-1})$$

$$\frac{\partial L}{\partial W^L} = X^L \frac{\partial L}{\partial X^{L+1}}$$

Picture: commons.wikimedia.org/Creative Commons

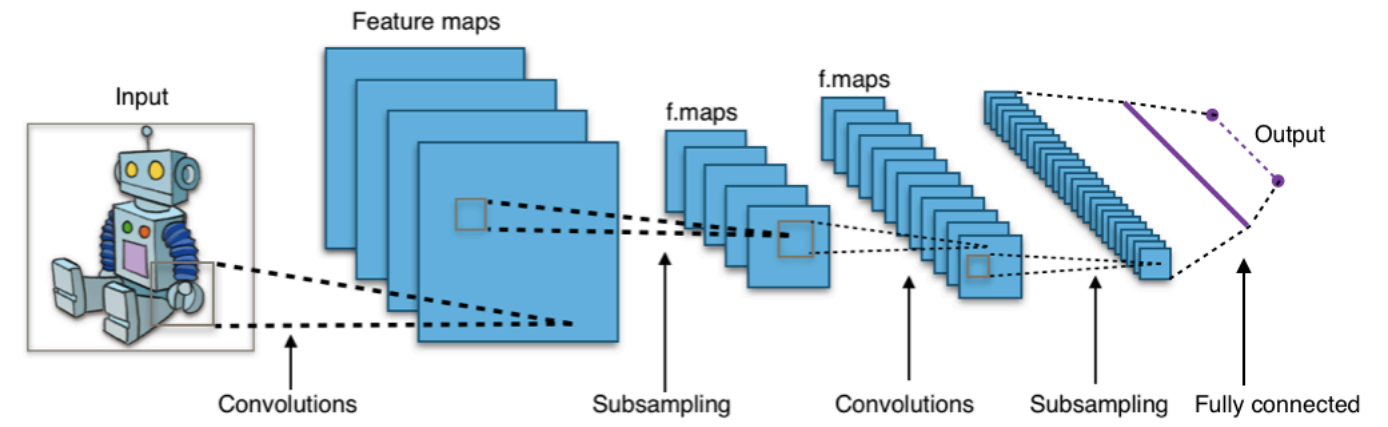
Double Layer NN

$$X * W + b = Y$$



Convolutional Neural Networks

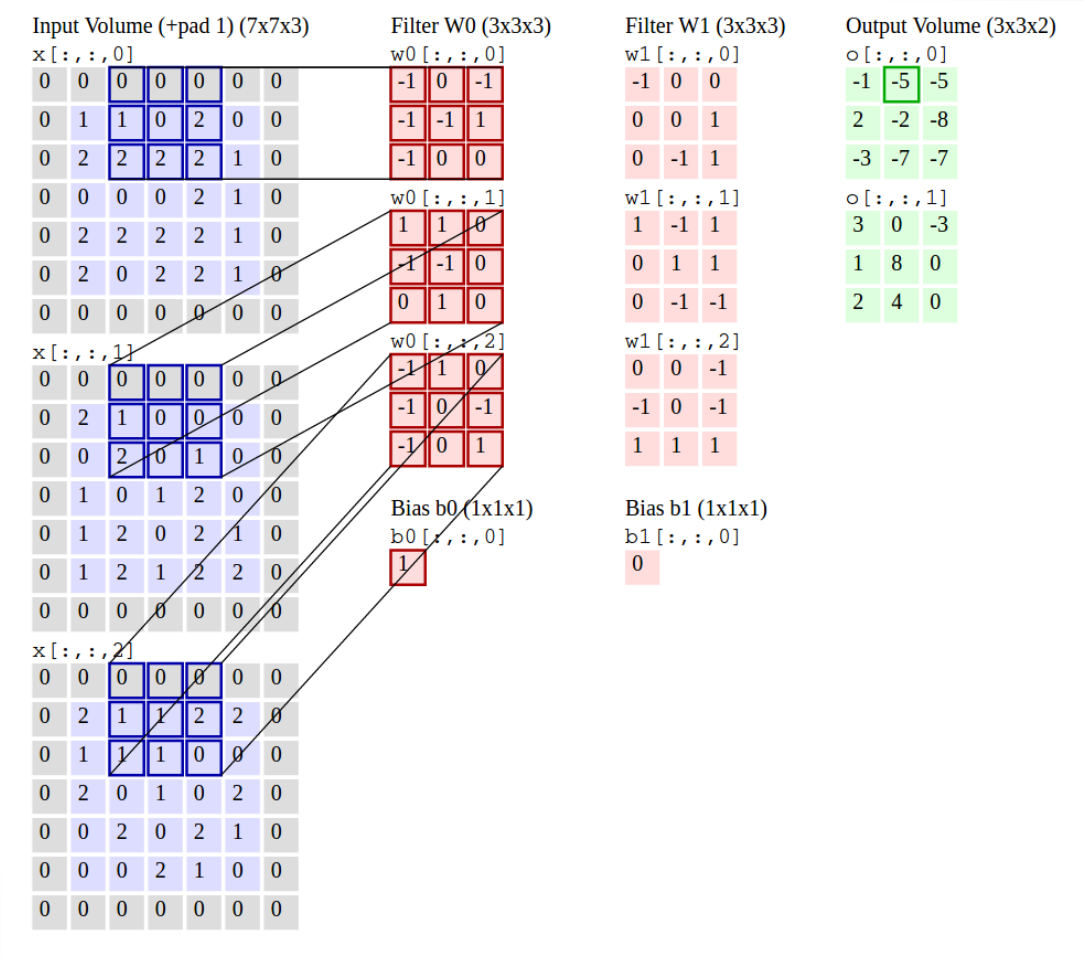
- Inspired by biology – visual cortex.
- Input is broken up into multiple small regions and processed by individual small networks.
- Alternating sequence of convolution and pooling.
- Convolution filter weights shared across image.



Picture: en.wikipedia.org/Creative Commons

CNN (cont'd)

- Convolution



CNN (cont'd)

- Pooling – max pooling over (2, 2) regions

$$\begin{bmatrix} 1 & 0 & 2 & 3 \\ 4 & 6 & 6 & 8 \\ 3 & 1 & 1 & 0 \\ 1 & 2 & 2 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 6 & 8 \\ 3 & 4 \end{bmatrix}$$

- Alternate layers of convolution and pooling result in production of "higher order features".
- FCN Layers classify higher order features into scores and classes with Softmax.

CNN Demo

- http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

- **More resources on CNN**
  - **http://cs231n.github.io/convolutional-networks/**
  - **http://deeplearning.net/tutorial/lenet.html**

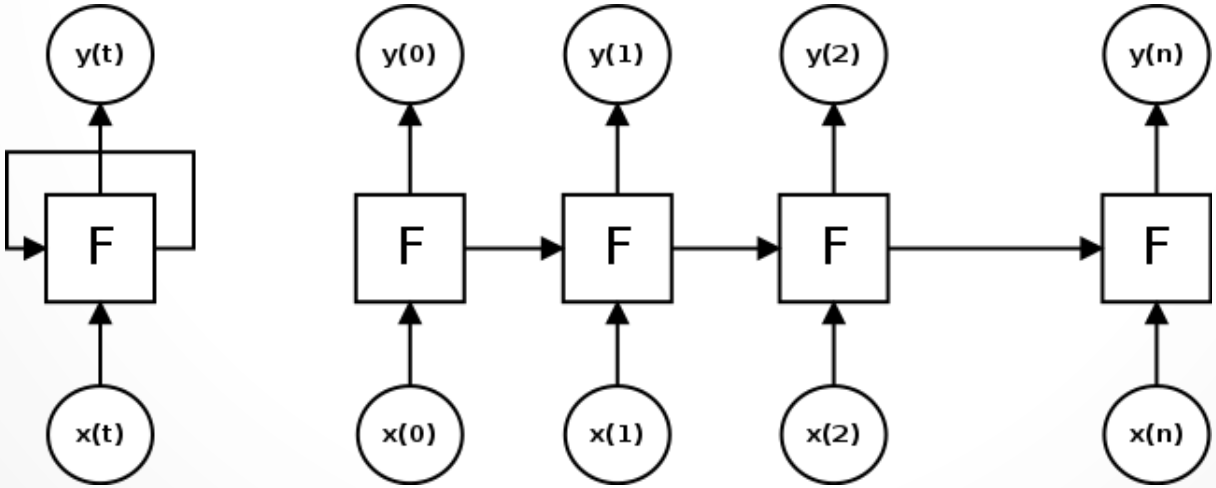
Tensorflow Lab

- https://github.com/abhi21/tf-tutorial

Recurrent Neural Network

- Biological inspiration: memory.
- Input at time step is combined with input from all previous time steps.
- Weights shared between all networks in chain.

$$y_t = \sigma(W X_t + U y_{t-1})$$

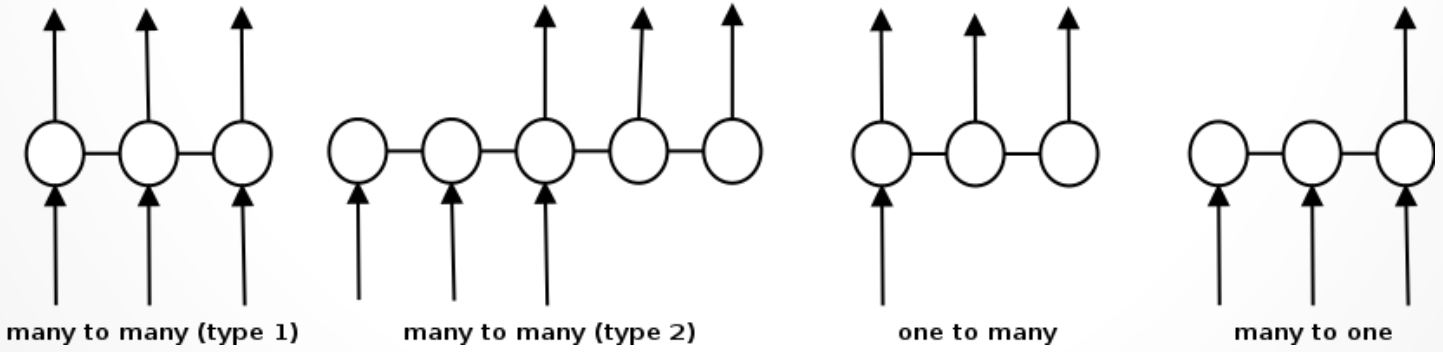


RNN (cont'd)

- RNN have vanishing gradient problem because of inputs from many time steps back.
- LSTM (Long Short Term Memory) – designed to fix the vanishing gradient problem, but performance intensive.
- GRU (Gated Recurrent Unit) – simplifies LSTM, results in better performance.
- Good discussion on RNNs can be found here:
  - Understanding LSTM Networks by Christopher Olah
  - Recurrent Neural Network Tutorial, Parts 1-4 on the WildML blog.

RNN (cont'd)

- Demo #1 – generative language model (example of many to many type 1).
- Demo #2 – machine translation example (example of many to many type 2).
- Demo #3 – sentiment classification (example of many to one architecture).



Conclusion

Learn More!

Note: many concepts were not covered!

- Machine Learning (Tom M. Mitchell)
- Pattern Recognition and Machine Learning (Christopher Bishop)

- MOOCs:
  - Machine Learning – covers almost all the important concepts in Machine Learning
  - Deep Learning on Udacity – good coverage of the basics of Deep Learning and Tensorflow.
  - CS224d on Stanford – Deep Learning for Natural Language Processing, taught by Richard Socher.
  - CS231n on Stanford – Convolutional Neural Networks for Visual Recognition.

- Deep Learning Enthusiasts meetup group

References

- Google Trends

- notMNIST Dataset: http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html
- http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf
- https://classroom.udacity.com/courses/ud730/
- http://cs231n.github.io