Introduction to Deep Learning

Notes

Human Visual System

- The human visual system is one of the wonders of the world. Consider the following sequence of handwritten digits:

- Most people effortlessly recognize those digits as 504192
- In **each hemisphere of our brain**, humans have a **primary visual cortex**, also known as **V1**, **containing 140 million neurons**, **with tens of billions of connections between them**
- And yet human vision involves not just V1, but an entire series of visual cortices - V2, V3, V4, and V5 - doing progressively more complex image processing
- We carry in our heads a supercomputer, tuned by evolution over hundreds of millions of years, and superbly adapted to understand the visual world
- Recognizing handwritten digits isn't easy. Rather, we humans are stupendously, astoundingly good at making sense of what our eyes show us

Challenges in visual pattern recognition

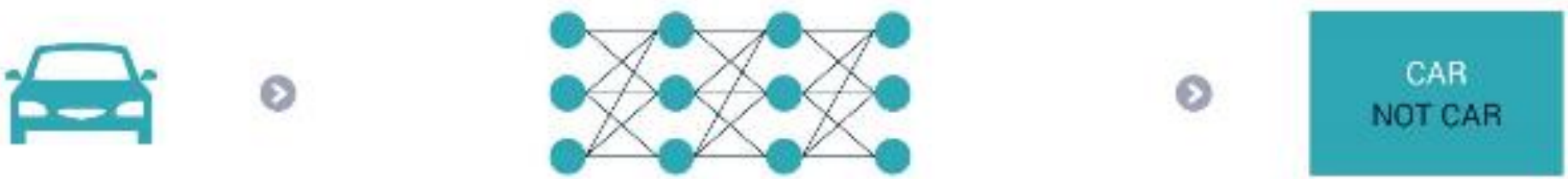
- The difficulty of visual pattern recognition becomes apparent if you attempt to write a computer program to recognize digits like those above
- What seems easy when we do it ourselves suddenly becomes extremely difficult
- Simple intuitions about how we recognize shapes - "a 9 has a loop at the top, and a vertical stroke in the bottom right" - turn out to be not so simple to express algorithmically
- When you try to make such rules precise, you quickly get lost in a morass of exceptions and caveats and special cases
- It seems hopeless.

Machine Learning vs Deep Learning

Machine Learning

Input → Feature extraction → Classification → Output

Deep Learning



Input → Feature extraction + Classification → Output

ML vs DL

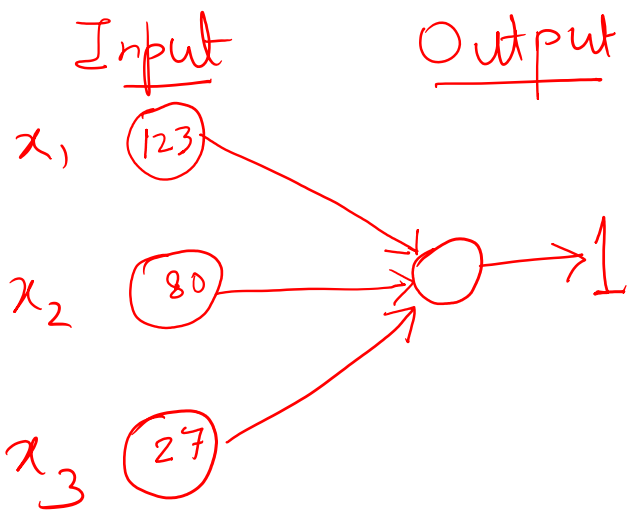
- ML
  - Flow Input -> Feature Engineering -> Classification(Model) -> Output
  - Feature Engineering
    - EDA
    - Feature Elimination
      - Missing Value Ratio
      - Low Variance Filter
      - Random Forest Classifier
      - High Co-relation filter
    - Feature Extraction
      - Extracting the variability into a fewer variables
      - PCA works on co-related variables and creates un-corelated variables
- DL
  - Flow Input -> (Feature Engineering + Classification) -> Output
  - DL handles Feature Engineering by itself…you don’t need to do that..

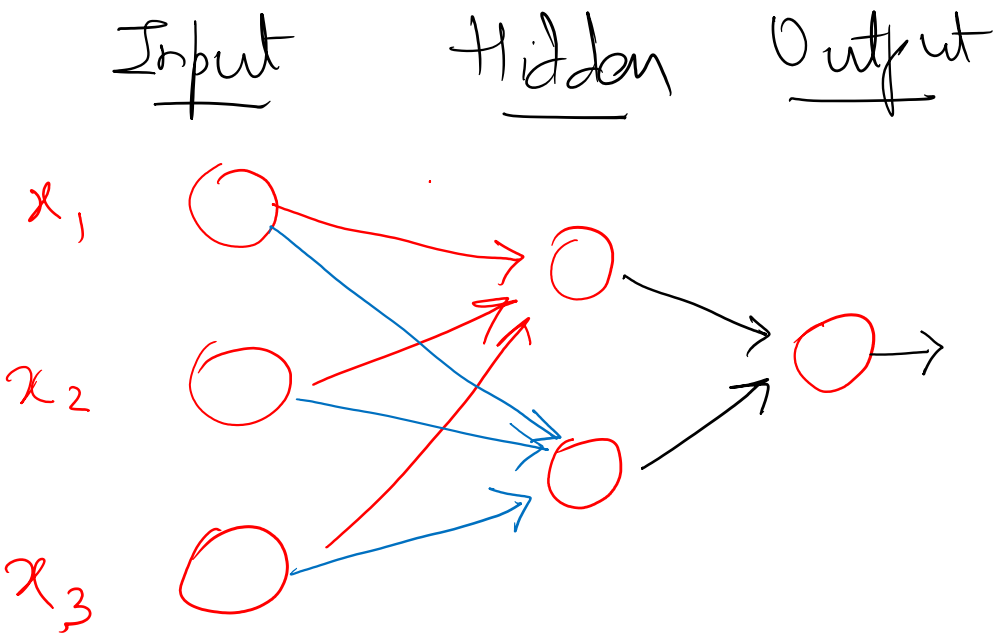
ML vs DL

- ML
  - Mostly used with structured data
- DL
  - Mostly used with unstructured data – images, videos, text

Neural Network Representation

	X			Y
	x_1	x_2	x_3	
1	123	80	27	1
2	.	.	.	0
⋮				⋮

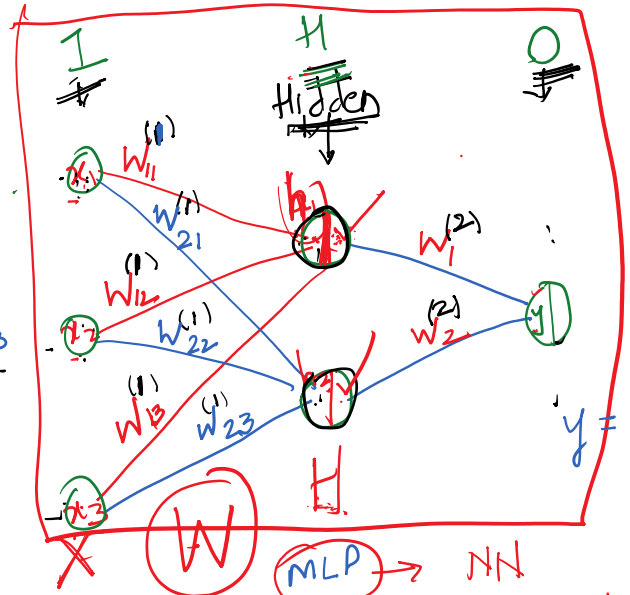




Perceptron & Neural Networks

- The **perceptron** is an algorithm for supervised learning of binary classifiers
- A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class
- It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector
- ***Perceptron is a single layer neural network*** *and a multi-layer perceptron is called Neural Networks.*

DNN



NN
Neural

$h_1 = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3$

$h_2 = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3$

$y = w_1^{(2)} h_1 + w_2^{(2)} h_2$

$h_1, h_2 \dots$

① $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

② $H^{(1)} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

$$*X = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{11}x_1 + w_{12}x_2 + w_3 x_3 \\ w_{21}x + w_{22}x_2 + w_{23}x_3 \end{bmatrix}_{2\times 1} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = H$$

(W: 2×3; X: 3×1)

$H = W^{(1)} * X$

$Y = W^{(2)} * H$

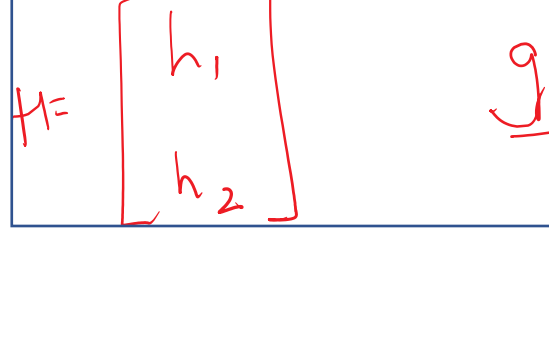
$H^{(1)} = W^{(1)} * X$

$H^{(2)} = W^{(2)} * H^{(1)}$

$Y = W^{(3)} * H^{(2)}$

$H^{(k)} = W^{(k)} * H^{(k-1)}$

where $H^{(0)} = X$

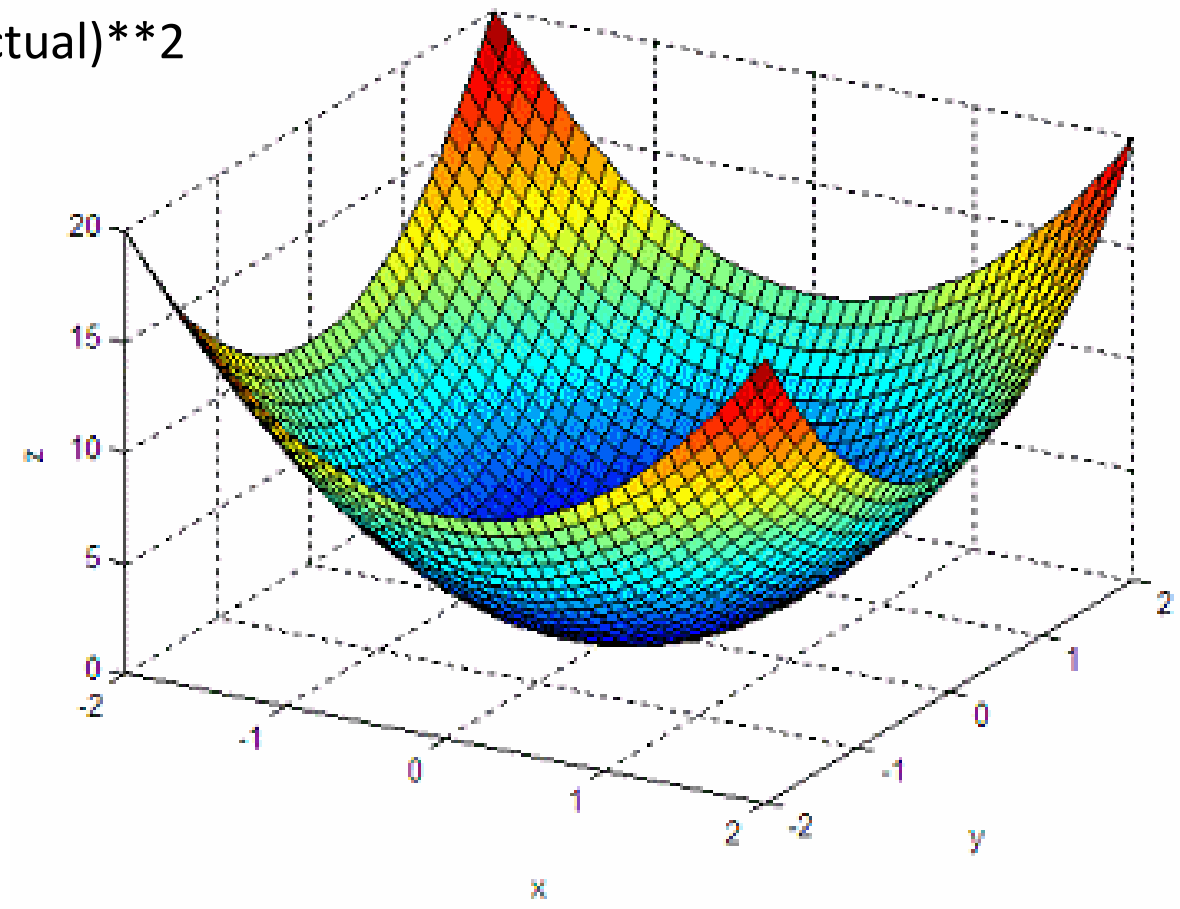


Forward and Backward propagation

- Go through the Perceptron stages: Input, Summation, Activation and calculate the Loss (based on the Target)
- This is forward propagation
- Based on the Loss, adjust the only thing you can…the Weights/Bias
- This is backward propagation
- Multiple such forward and backward passes need to reduce the Loss
- This is the learning process

Gradient Descent

- Y = m1x1 + m2x2 + c
- Loss v/s m1, m2, c
- Loss = (Pred-Actual) = ((m1x1 + m2x2 + c) – Actual)**2
- Find m1, m2, c such that Loss is minimum
- Use Gradient descent for this
- Derivative to find minima:



Representation Learning

- Representation learning is learning representations of input data typically by transforming it or extracting features from it*(by some means)*, that makes it easier to perform a task like classification or prediction.