Machine Learning and Data Science

CS277

Instructor Details

- Name: Dr. Samrat Mondal
- Department: CSE
- Office: 405, 4th Floor, Block 3
- Email: samrat@iitp.ac.in

Class Timings & Classroom

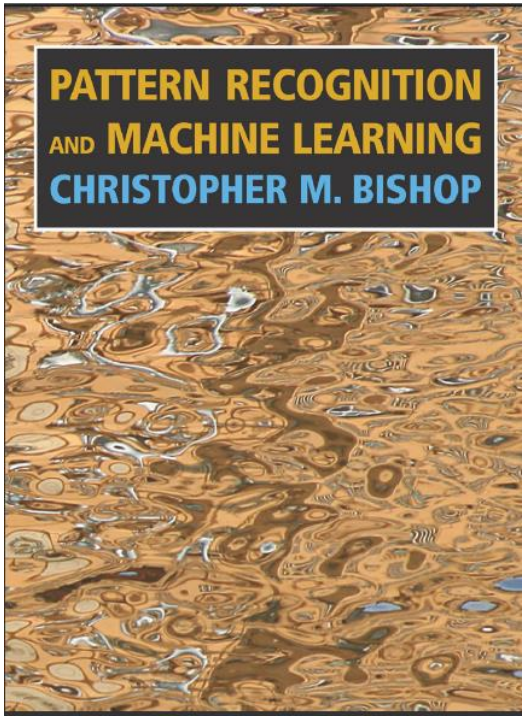
- Wed: 3 pm to 4 pm, R409
- Thur: 4 pm to 5 pm, R409
- Fri: 5 pm to 6 pm, R409

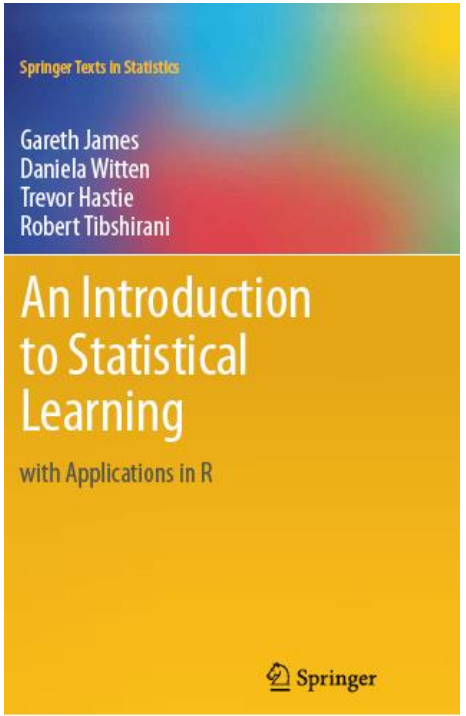
TA Names and Emails:

- Medhasree Ghosh: medhasree_2121cs05@iitp.ac.in
- Muskan: muskan_2211cs32@iitp.ac.in

Course Page: http://10.12.10.9/~samrat/CS277/CS277_2024/

Reference Books





Evaluation Policy

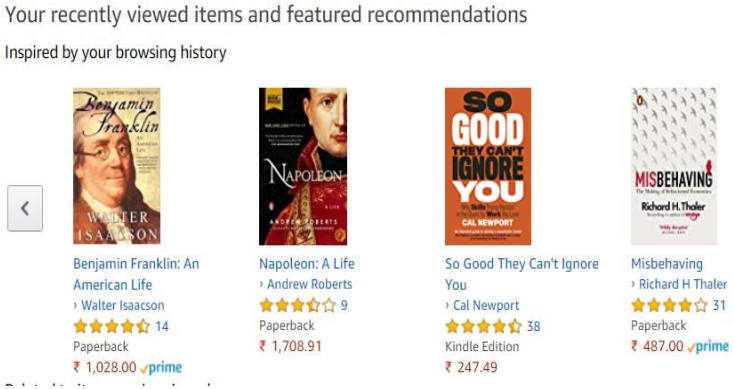
- Quizes/Assignments/Internal Assessment: 30%
- MidSem:30%
- EndSem:40%

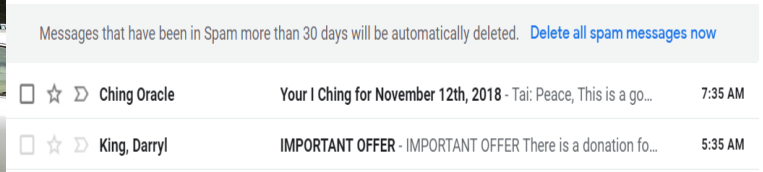
Important Note: Students are expected to adhere to ethical standards throughout their course attendance and examination participation. Engaging in any form of academic misconduct, such as malpractice, proxy attendance, or providing false information to gain advantages, will result in severe penalties.

Acknowledgement

- Some of the materials of this course are borrowed from the “Machine Learning for Engineering and Science Applications” course materials used by Prof. Balaji Srinivasan and Prof. Ganapathy of IIT Madras.

What makes these possible?



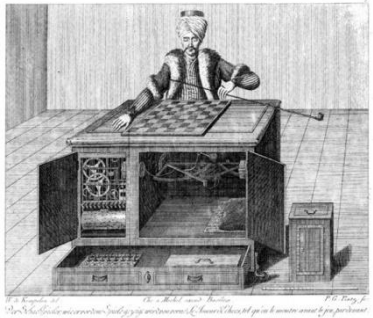


Simplistic Definition -- Machine Learning aims to replicate activities requiring human cognition

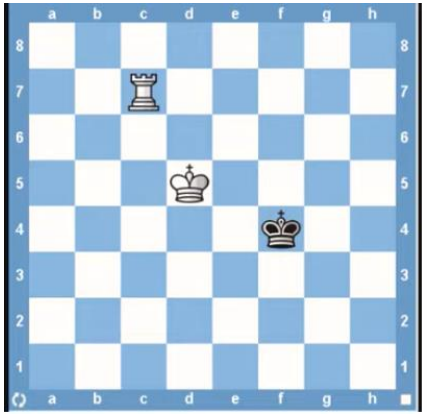
History of Artificial Intelligence

Prehistory (Till 1900)

- 11th century – Robots that could replicate human speech and motion (Raja Bhoja)
- Realistic automatons in several parts of the world
- Liebnitz – All human ideas are combinations of a few thoughts
- 1837 – Charles Babbage – Analytical Engine



Birth of the field of Artificial Intelligence (1900-1960)



1914 First chess playing machine (KR-K endings)

1925 Radio-controlled driverless car (Francis Houdina)

1940s Pitts and McCulloch – First Artificial Neuron

Alan Turing – Theory of Computation, Imitation Game

Shannon (Information Theory)

1950 Wiener – Cybernetics

1951 Minsky– (SNARC) First Neural Net Machine

1955 Simon and Newell – Logic Theorist – Theorem Proving machine

1956 Dartmouth conference – **The term "A.I" is coined** with the aim to build thinking machines.
Formal birth of Artificial Intelligence

1957 Rosenblatt – Perceptron – Two-layer Artificial Neural Network

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed. – Arthur Samuel, 1959

The golden years (1960-74)

- Appearance of Expert Systems – Explicit, rule based, programs
  - Playing Chess
  - Helping in constructing organic chemistry models
  - Solving word problems in algebra
  - Understanding natural language
  - General purpose mobile robot
  - Identifying infections and recommending antibiotics
- Theoretical Progress – Backpropagation – 1969 (Bryson and Ho)

Optimism

"Machines will be capable, within twenty years, of doing any work a man can do"

-- H.A. Simon (1965)

"In from three to eight years we will have a machine with the general intelligence of an average human being." – Marvin Minsky (1970)

…..etc, etc

But, we should always remember that, after great times…..

The first A.I. winter (1974- 80)

Problems with A.I

- Results were primarily for toy problems
- Low computational power
- Combinatorial explosion
- Commonsense is nearly impossible to program!
- Minsky's book – *Perceptrons* showed limitations of simple neural networks
- Most importantly, loss of government funding in A.I

A new seasonal cycle (1980-2000)

Boom – Spring -- (1980-87)

- Expert Systems used in businesses with specialized hardware
- First driverless car
- Hopfield networks, popularization of backpropagation
- Minsky (1984) – “Winter is coming!”

Bust – 2nd A.I Winter – (1987-93)

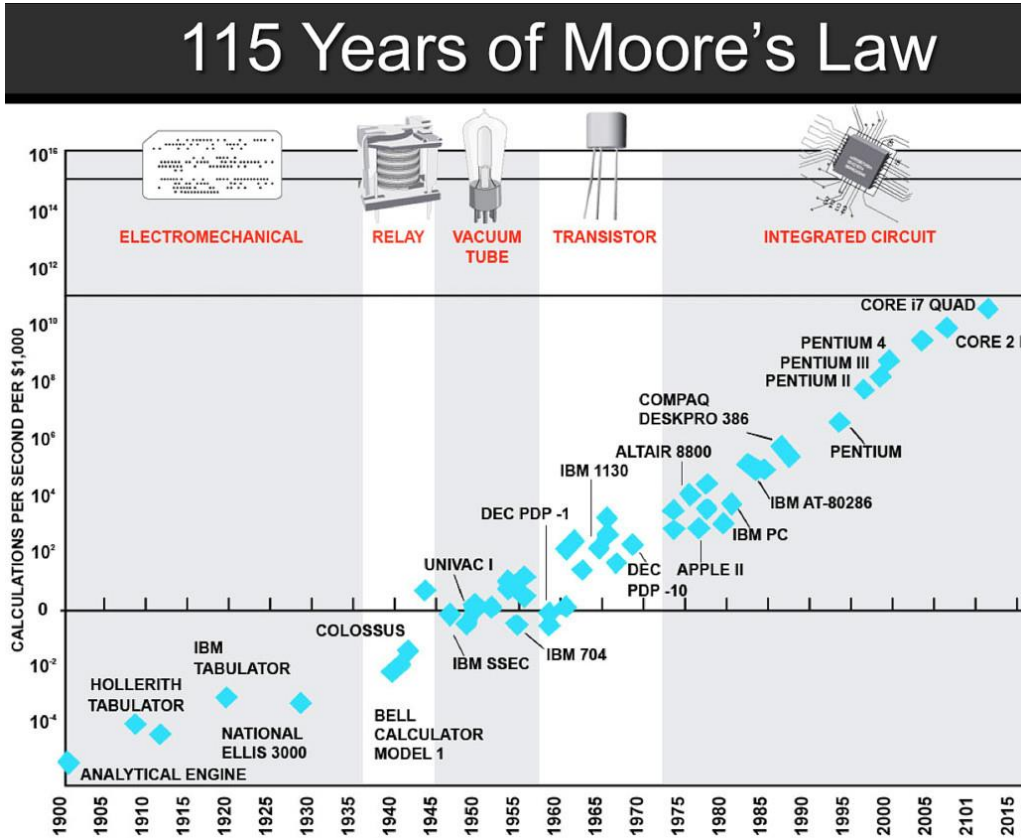
- Popularity of the PC.
- Disappointment in lack of spectacular results
- Brutal funding cuts

Consolidation – Summer – (1994-2000)

- 1997 Deep Blue beats Kasparov in chess
- Theory – Including probability, information theory, optimization, etc
- Moore’s Law – Rapid growth of processing power

Moore’s Law : Number of transistors doubles every two years

Moore's Law



Steve Jurvetson -- https://c1.staticflickr.com/2/1640/25046013104_68059057ab_b.jpg

The quiet years (2000-12)

- Google is born.
- Internet Boom
- Shifted emphasis to big data – statistical techniques

- Birth of Graphical Processing Units (GPUs)
- Good results in specific problems using deep networks

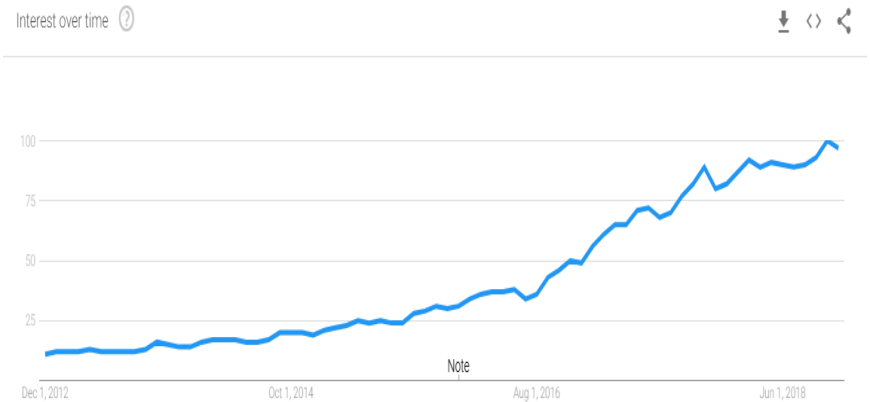
- Research focused on specific outcomes rather than general, all purpose, A.I
- 2005 – Autonomous driving for 135 miles in desert
- IBM's Watson beat the *Jeopardy* champions

And after a long, quiet consolidation …..

The A.I. Spring (2012 – ??)

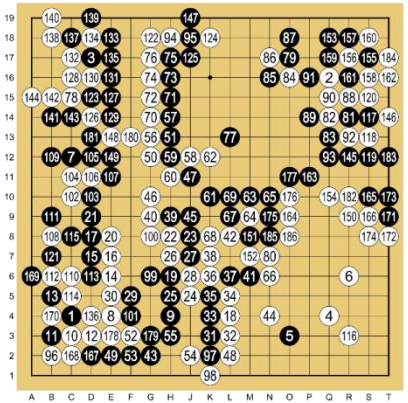
- Lots of private funding
  - Google, IBM, Facebook, Microsoft….
- Rapid development in computational power
- Rapid growth of data
  - Data mining
  - Voluntary, distributed work (Amazon Turk, Captchas, Games)

Growth of Machine Learning/A.I



2012 Convolutional Neural Networks perform extremely well on image recognition challenge (ImageNet)

Machines vs Humans


Lee Sedol (B) vs AlphaGo (W) - Game 1

What is different this time?

What is different this time

1. Better technology
   - Exponentially huge computational power

2. (Really, really) Big Data

3. Democratization of resources
   - Software and Hardware

4. Better algorithms

What is different this time

1. Better technology
   - Exponentially huge computational power

2. (Really, really) Big Data

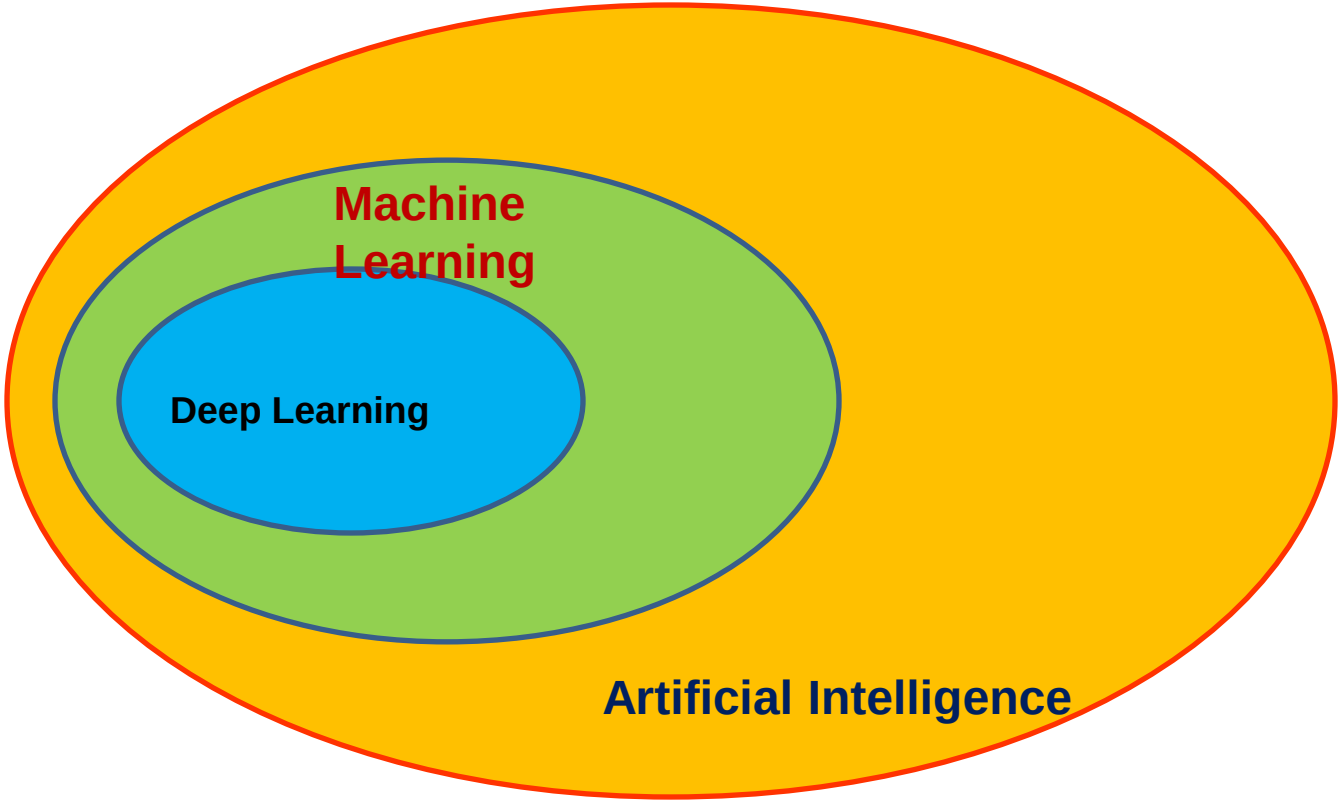
3. Democratization of resources
   - Software and Hardware

4. Better algorithms

Some Common Terms

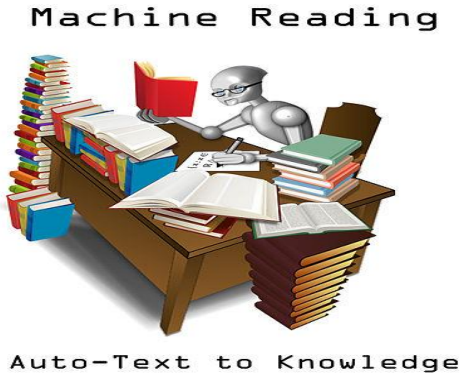
- Artificial Intelligence – Any method that tries to replicate the *results* of some aspect of human cognition
- Machine Learning – Programs that perform better with experience.
- Artificial Neural Networks (ANN) – A Machine Learning algorithm
- Deep Learning – A type of ANN
- Big Data – Using data to find unobvious patterns

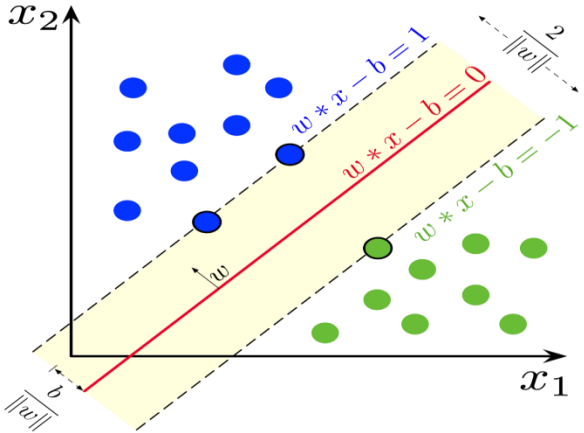
The AI Venn diagram



Adapted from *Deep Learning*, Goodfellow et al (2016)

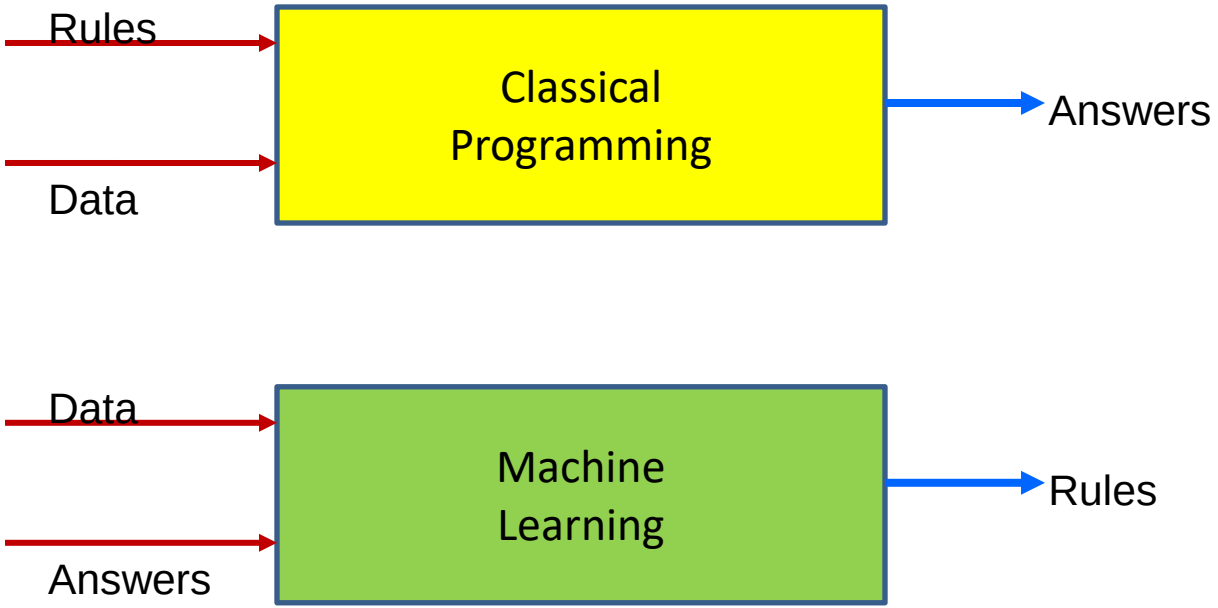
What is Machine Learning?





- Simple Definition -- Using Data to answer questions
- Study of computer algorithms
  - that improve automatically
  - through experience.
- Formally, A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

The Machine Learning Paradigm



When is Machine Learning useful?

- When experts are unable to explain their expertise
  - Image recognition
  - Speech recognition
  - Driving a car

- When Human expertise does not exist
  - Hazardous environments -- Navigating on Mars

- Solution needs to be adapted to particular cases
  - User biometrics
  - Patient specific treatments

A fundamental “trick” in most of ML

- All problems are data, all solutions are functions/maps

- Cognitive tasks -- Humans get sensory inputs as qualia
  - We must convert these qualitative inputs into numbers – Input Vectors
  - Similarly, outputs that humans give must also be converted into numbers – Output/Target vectors

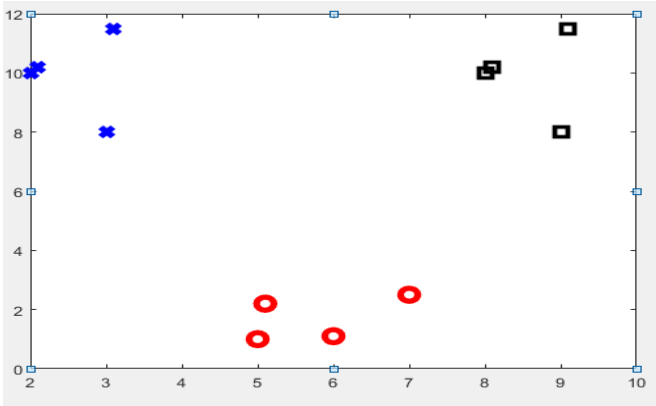
- Determining appropriate inputs and outputs for a machine learning task is an essential part of the process

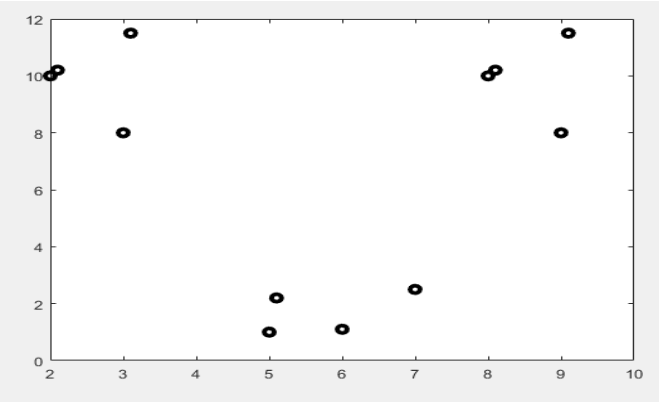
- Often the “Learning Task” is learning the mapping from input to output.

Types of learning approaches

- Supervised Learning
  - Data labeled by human experts
  - Labeling images
  - Speech recognition
  - OCR

- Unsupervised Learning
  - Unlabeled data
  - Grouping customers
  - Detecting new diseases
  - Anomaly detection





Other types of learning approaches

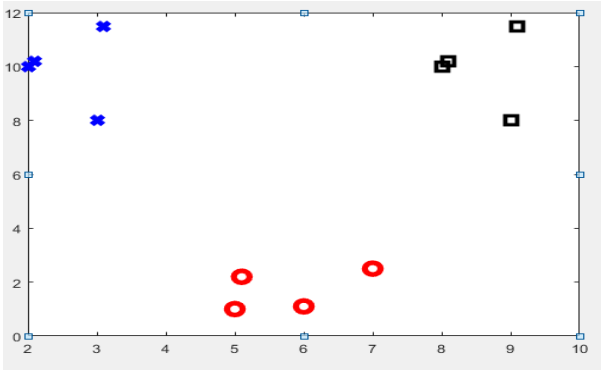
- Generative approaches
  - Creating new data that is “like” given data
  - Generally included in unsupervised learning
- Semi-supervised learning
  - Small amount of labeled data available along with unlabeled data
- Self-supervised learning
  - Implicit labels are extracted from data using heuristics
- Reinforcement learning
  - Actions are chosen based on rewards. Example : Chess, Games, etc
  - Feedback is far removed temporally from action

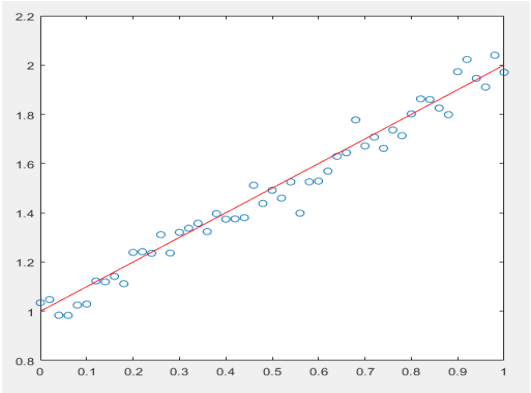
The distinction between the various types of learning is often blurred

Seven Steps in Machine Learning

1. Gathering Data
   - Deciding what “data” means is part of the problem
2. Preparing Data
   - Ensuring that there is no bias
3. Choosing a Model/Algorithm
   - Examples – Random Forest, ANNs, Hidden Markov Models,etc
4. Training
   - Using data to determine model parameters
5. Evaluation – How well did we do?
6. Hyperparameter Tuning
7. Prediction

Two problems in Supervised Learning





Classification	Regression
Split it	Fit it
Discrete or Categorical data.	Real number data
Has category associated	Has associated number
Example : Tumour classification	Example : Prediction of stock market

Mathematical ideas we will be using in this course

- Linear Algebra
  - Machine Learning involves mapping
    - From vectors to vectors
    - That is, matrix based transformations – Linear Algebra

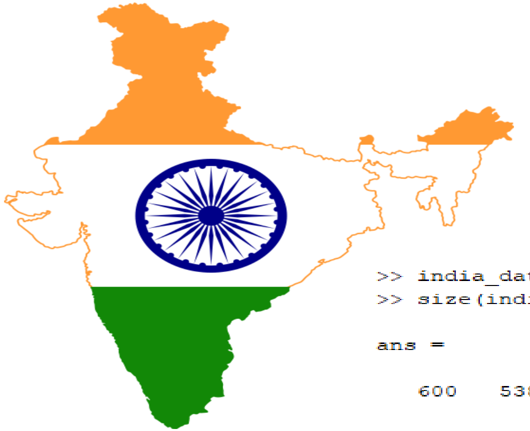
- Probability
  - Data and results have uncertainty built into them
  - Conditional probability is a very important component of ML

- Optimization
  - For a given set of data what is the “best” model?
  - Many ML models finally reduce to solving some optimization problem

Why linear algebra is useful

- In many Machine Learning algorithms, the input and the output are both represented as vectors
- By vectors we simply mean a collection of numbers
- Part of the problem is to convert a seemingly qualitative input (such as a picture, sound, colour, etc) into a number
- Let us see an example….

From image to vector



```
>> india_data=imread('india.png');
>> size(india_data)

ans =

   600   538     3
```

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	240
255	255	255	255	255	255	255	255	255	255	231	80
255	255	255	255	255	255	255	255	255	242	78	51
255	255	255	255	255	255	255	255	255	131	51	51
255	255	255	255	255	255	255	255	225	54	51	51
82	140	209	254	255	255	255	255	108	51	51	51
51	51	51	89	180	227	209	120	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51

Notation

Scalar : Single number.

Example : *Let* $\alpha \in \mathbb{R}$, *be the learning rate*

Let $n \in \mathbb{N}$, *be the number of hyperparameters*

Vector : In ML, array of numbers.

Example : *Let* $\vec{x} \in \mathbb{R}^n$, *be the input vector.*

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

Matrix : In ML, 2-D array of numbers.

Example : *Let* $W = \mathbb{R}^{m \times n}$ *be the matrix of weights*

Tensors : In ML, array of numbers with dimensions greater than 2

Example : $A_{i,j,k}$

Scalars, Vectors, Matrices, Tensors

Scalar (0^{th} order tensor) $\alpha = 3$

Vector (1^{st} order tensor)

$$\vec{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Dimension of the example vector is?

Matrices, Tensors

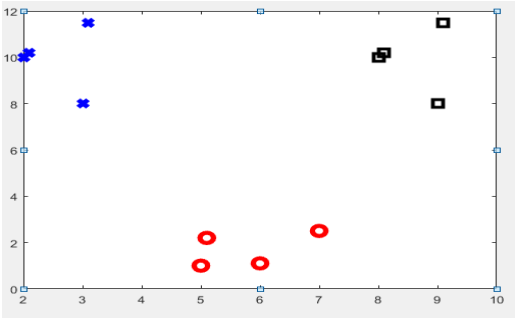
Matrix (2nd order tensor)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 5 \end{bmatrix}$$

Tensors (3rd and higher order tensors)

- Example: Colour images, Video data

Implications of tensor representation



- We represent both vectors and transformations as tensors
  - Trasformations between vectors -> vectors are naturally represented as matrices
- Could be high dimensional representations
  - Need algorithms that work well in high dimensions
- Lets us go back and forth between images and numbers
  - Very useful for engineering applications

Tensor operations

- Addition, Broadcasting
- Multiplication
  - Matrix Product, Dot Product, Hadamard Product (Elementwise multiply)
- Transpose
- Inverse

Addition

- Normal Matrix addition: $A_{ij} + B_{ij} = C_{ij}$

```
>> A = [ 1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>> B = [ 1 5 6; 1 7 8]

B =

     1     5     6
     1     7     8

>> C = A + B

C =

     2     7     9
     5    12    14
```

- Broadcasting: $A_{ij} + b_j = C_{ij}$
  - Adding a vector to a matrix by repeating the vector
  - Done automatically in MATLAB and Numpy

```
>> A

A =

     1     2     3
     4     5     6

>> b = [1 1 1];
>> A + b

ans =

     2     3     4
     5     6     7
```

Multiplication

- Matrix Product : $C = AB$
  - $C_{ij} = \sum_k A_{ik} B_{kj}$
  - Sizes must match

```
>> A

A =

     1     2     3
     4     5     6

>> b

b =

     3
     4
     5

>> C = A*b

C =

    26
    62
```

- Hadamard Product: $C = A \odot B$
  - Elementwise multiplication
  - A, B and C must be of the same size

```
>> A

A =

     1     2     3
     4     5     6

>> B

B =

     1     5     6
     1     7     8

>> A.*B

ans =

     1    10    18
     4    35    48
```

Multiplication (contd)

- Dot Product : $\vec{a} \cdot \vec{b} = \alpha$
  - $\alpha = \sum_i a_i b_i$
  - Can also be written as $\alpha = a^T b = b^T a$

```
>> a = [ 1 2 3]'

a =

     1
     2
     3

>> b = [4 5 6]'

b =

     4
     5
     6

>> alpha = dot(a,b)

alpha =

    32
```

```
>> alpha1 = a'*b

alpha1 =

    32
```

```
>> alpha2 = b'*a

alpha2 =

    32
```

Used very often. Ensure you can switch between vector and matrix notations

Transpose and Inverse

- Transpose : $B = A^T$
  - $B_{ij} = A_{ji}$

```
>> A

A =

     1     2     3
     4     5     6

>> B = A'

B =

     1     4
     2     5
     3     6
```

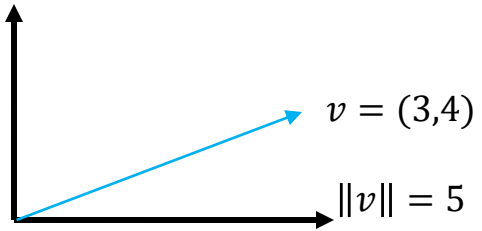
```
>> A = rand(3)

A =

    0.7814    0.5567    0.7802
    0.2880    0.3965    0.3376
    0.6925    0.0616    0.6079

>> inv(A)

ans =

   50.2178  -66.1993  -27.6885
   13.3927  -14.8948   -8.9171
  -58.5694   76.9290   34.0937

>> A*inv(A)

ans =

    1.0000    0.0000   -0.0000
    0.0000    1.0000   -0.0000
    0.0000    0.0000    1.0000
```

- Inverse: $I = A^{-1}A = AA^{-1}$
  - Above definition is for a square matrix
  - Not all square matrices have an inverse
  - For non-square cases we can define something called the pseudoinverse

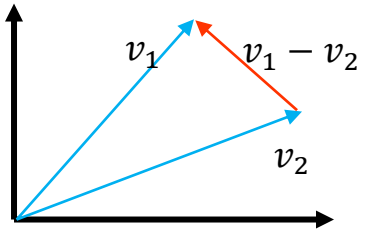
The reason to use norms

- Machine Learning uses tensors as the basic units of representation
  - Vectors, Matrices, etc..

- Two reasons to use norms
  1. To estimate how “big” a vector/tensor is



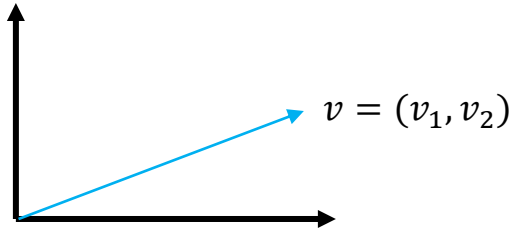
  2. To estimate “how close” one tensor is to another
     - That is how “big” the *difference between two tensors* is
     - Example : How close is one image to another?

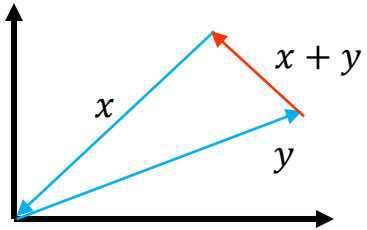


Norm is the generalization of the notion of “length” to vectors, matrices and tensors

Definition of a norm

Norms are a way of measuring the “length” of vectors, matrices, etc



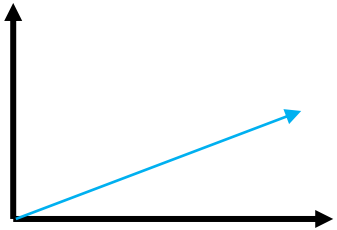


Mathematically, a norm is any function $\boldsymbol{f}$ that satisfies

- $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = \boldsymbol{0}$
- $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (Triangle Inequality)
- $\forall \alpha \in \mathbb{R},\ f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$ (Linearity)

Some standard norms

- $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = \boldsymbol{0}$
- $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (Triangle Inequality)
- $\forall \alpha \in \mathbb{R},\ f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$ (Linearity)



```
>> v = [-5,3,2]'

v =

    -5
     3
     2
```

Vector Norms

1. Euclidean Norm : $\|v\|_2 = (v_1^2 + v_2^2 + v_3^2 \ldots\ldots + v_n^2)^{\frac{1}{2}}$
   - Also called the 2-norm or the L^2 norm
   - Corresponds to our usual notion of distance
2. 1-norm : $\|v\|_1 = |v_1| + |v_2| + \cdots |v_n|$
3. p-Norm : $\|v\|_p = (|v_1|^p + |v_2|^p + \cdots + |v_n|^p)^{\frac{1}{p}}$
4. ∞-Norm : $\|v\|_\infty = \max(|v_1|, |v_2|, \ldots, |v_n|)$

```
>> norm(v,2)

ans =

    6.1644
```

```
>> norm(v,1)

ans =

    10
```

```
>> norm(v,inf)

ans =

     5
```

- Matrices : Frobenius norm $A_F = \left(\sum_{i,j} A_{ij}^2\right)^{\frac{1}{2}}$

```
>> A = [1 2;2 0]

A =

     1     2
     2     0
```

```
>> norm(A,'fro')

ans =

    3
```

Linear combination

Linear combination : A linear combination of the set of vectors $\{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}\}$ is given by multiplying each vector by a corresponding scalar coefficient and adding the results

$$\sum_i \alpha_i \boldsymbol{v}^{(i)}$$

Example : $\boldsymbol{v}_1 = \begin{bmatrix}1\\2\\3\end{bmatrix}$, $\boldsymbol{v}_2 = \begin{bmatrix}2\\0\\3\end{bmatrix}$

$$\boldsymbol{v}_3 = v_1 + 2v_2 = \begin{bmatrix}5\\2\\9\end{bmatrix}$$

Note: $\boldsymbol{v}_3 = [\boldsymbol{v}_1\ \boldsymbol{v}_2]\begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}1 & 2\\2 & 0\\3 & 3\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}$

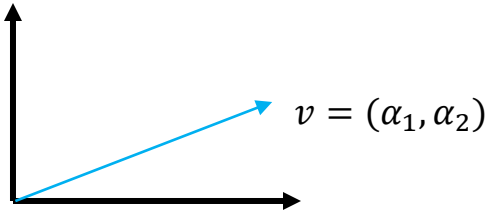
Matrix multiplications can be interpreted in terms of linear combinations of columns

```
>> V = [1 2; 2 0; 3 3]

V =

     1     2
     2     0
     3     3

  >> V*A

  ans =

       5    11
       2     6
       9    21
```

```
>> A = [1 3;2 4]

A =

     1     3
     2     4
```

Span

Span : The span of a set of vectors is the set of all vectors obtainable by a linear combination of the original vectors.



Example : The span of the coordinate vectors $v_1 = (1,0),\ v_2 = (0,1)$ is?

Ans :

The span of all the columns of a matrix is called the column space

Note: The equation $A\boldsymbol{x} = \boldsymbol{b}$ has a solution only if $\boldsymbol{b}$ lies in the column space of *A*

Linear independence

Linear independence : A set of vectors is linearly independent if none of these vectors can be written as a linear combination of the other vectors.

Example: $\{v_1 = (1,0), v_2 = (0,1)\}$ linearly independent

$\{v_1 = (1,0), v_2 = (0,1), v_3 = (3,4)\}$ linearly dependent

Mathematically, $S = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ is linearly independent if and only if the linear combination $\alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + \cdots \alpha_k \boldsymbol{v}_k = 0$ means that all the $\alpha_i = 0$

Why matrix decomposition?

- Matrices transform one vector to another
- We will deal with high dimensional vectors and tensors
  - Recall images as an example of high dimensional vectors
- As with the prime factorization of numbers, it is useful to understand "components" of a matrix
- Also useful to get smaller set of representative numbers
  - Example : Norms, Trace, Determinant, Eigenvalues, Singular Values

Trace of a Matrix

- The trace of a matrix is given by the sum of its diagonal elements

$$Tr(A) = \sum_i A_{ii}$$

Some properties

$$Tr(A + B) = Tr(A) + Tr(B)$$

$$Tr(AB) = Tr(BA)$$

$$Tr(A) = Tr(A^T)$$

Determinant of a matrix

- Easiest to define recursively

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

- Laplace expansion : For a *nxn* matrix A

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} A_{ij} \det(a_{ij})$$

- Physically represents volume formed by column vectors

Invertibility of a matrix

- A square matrix is invertible if and only if $\det(A) \neq 0$

- Note that this automatically means that the columns of A have to be linearly independent

Special Matrices and vectors

- Diagonal Matrix – Only diagonal entries are non-zero

$$D_{ij} = 0 \text{ if } i \neq j$$

- Symmetric matrix – Matrix is equal to its transpose

$$A = A^T$$

- Unit vector – Vector with unit "length"

$$\|v\| = 1$$

- Orthogonal vectors – Mutually perpendicular

$$\boldsymbol{x}^T \boldsymbol{y} = 0$$

- Orthogonal matrix – Transpose is equal to inverse

$$A^T = A^{-1}$$
$$\Rightarrow A^T A = A A^T = I$$

Orthogonal matrices represent rotational operations which preserve volume

Eigen Decomposition

- Extremely useful for square symmetric matrices
  - Used for other matrices as well

- Physical meaning
  - Every real matrix can be thought of as a combination of rotation and stretching
  - Eigenvectors for a matrix are those special vectors that only stretch under the action of the matrix
  - Eigen values are the factor by which eigenvectors stretch

$$A\boldsymbol{v} = \lambda\boldsymbol{v}$$

Eigen Decomposition (contd)

- Say A has n linearly independent eigenvectors

$$\{\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \dots, \boldsymbol{v}^{(n)}\}$$

- Concatenate all the vectors (as columns) and make a eigenvector matrix V

$$V = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \dots, \boldsymbol{v}^{(n)}]$$

- If we concatenate the corresponding eigenvalues into a diagonal matrix

$$\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_n)$$

- Then, the eigen decomposition (factorization) of A is

$$A = V\Lambda V^{-1}$$

Physical interpretation of the eigen decomposition

- All matrices can be thought of as rotating and stretching vectors
- Eigenvectors have pure stretch
- Real symmetric matrices have real eigenvectors and real eigen values

$$A = Q\Lambda Q^T \text{ where } Q^T = Q^{-1}$$

- Eigen Decomposition may not be unique