Keynote

intro.

- Operations per second → OPS
  floating point opn's / second → FLOPS
- Benchmark → how many ops can it do.

Applications

1. Day-to-Day life
   a) Weather forecasting and early warnings
   b) Improving healthcare
      ↳ Bioinformatics
      Drug discovery, Drug mol. simul"s by HPC.
2. Science
   - invest. e⁻ structure
   - Air quality monitoring
   - Computational Fluid Dynamics.
3. Nano-Science
4. N-Cap rating

Basics of HPC

Parallel Computing

Principle :- Large problems, → divide into smaller problems
↓ solve concurrently.

Parallel Processing - Simultaneous processing. Each small task is assigned to a single processing unit.

Distributed Computing - Using multiple, isolated processing elements.

- Mostly used in numerical simulations, modelling and data processing.

Architecture of a supercomputer

Cluster - coll. of compute nodes

Cluster → connected by switch fabric
↓
high bandwidth, low latency

Compute nodes → microprocessor + hardware - monitor - mouse etc.
↓ data

User gets IP of login nodes

Login nodes

Parallel File System (Beowulf clusters) TCP/IP protocols

Primary interconnect

Network

Tape lib. / backup stora

Switch fabric

Compute nodes

Storage accelerator

Parallel File System
↓
- mag. media

Read
- PGAS
- Intelligent n/w card
- TCP/IP protocols
- wakewulf clusters
- DMA
- Param nets
- RDMA
- SSD

- DgX3
- SSH
- GPU
- GPGPUs
- Scheduling
- Boot Service / Login

SSD vs Hard disk
→ speed is faster but cost/bit > than hard disk.
→ compute nodes that are acc.

Process Scheduling

- Multiprogramming to ensure complete CPU utilizⁿ.
- Some processes running at all times.

- Time sharing → switching CPU among processes st. users can interact with all program.
- eg. Downloading files, writing to files, listening to music all happens at the same time acc. to us, but they are switched so freq. so as to maintain continuum.

↓

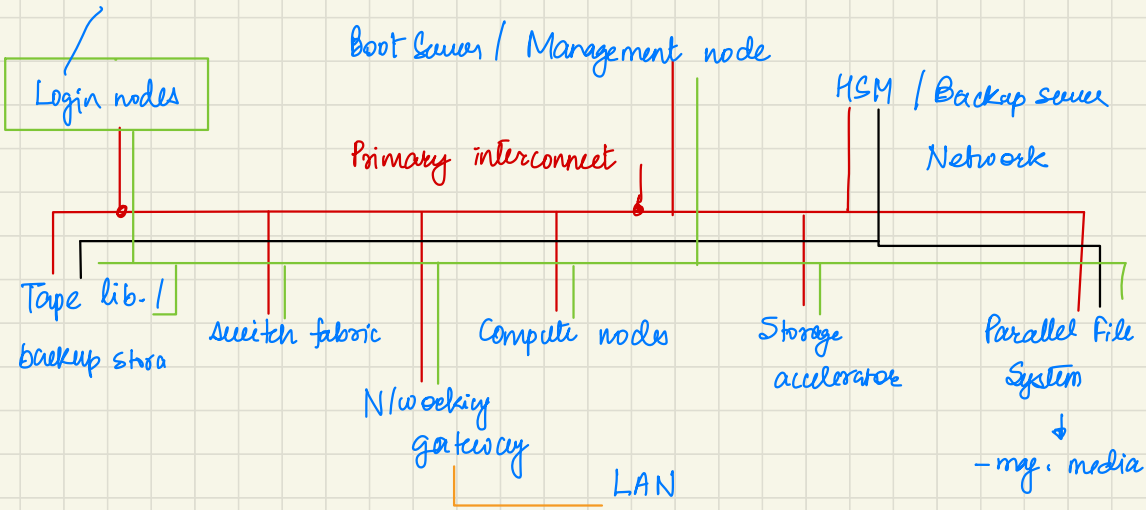
Process scheduler

- select av. proces and assign CPU core to execute
- select priority among processes and selects which instructns. can be executed at one time
- Rescheduling diff. processes.
- Scheduling queues

CPUs vs. GPUs

→ CPUs are powerful, but have limited parallel capability (32 or 64 cores)

→ GPUs have 4000 cores → limited memory/core but can do work independant and fast.



- Boot Server → allow these compute nodes to bootup in cfg that
  → one patch applies to all nodes

Software Subsystem

① - Every nodes has OS (Cent OS 7.x) in Param Servers.

② - Middleware → Cluster Monitoring → monitor clusters
  → Provisioning →
  → file system
  → Res. manage

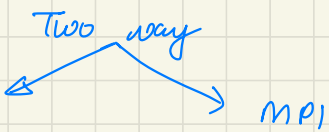
FI System { NFS maintains home area
Lustree 11[al] file system → Scratch

- Job submitted in queues. Priority queues
  - ↳ maintained by SLURM → managed

- Shell Scripting

(3) Programming tools
↓
- Perf. mon.
- Dev. tools
- App. libs

- Pragmas
- MPI → mech. by which we use a lib.



Open MP,
Pragma

- hypervisor in Cloud
- Debugging in Dedicated Mode.

- SIMD - concept as a part of Instn. Set.

Rack Mountable Server

- One

Parallelism in Arch.

i) Abstn. layers

black box → hardware ← top software

Parallel Prog. frame-work → Appln. Program

Multi-core, multi-processors → OS

SIMD → Instn. Set Arch (ISA)

Superscalar → Micro-architecture

Pipelined → Exec. units

} Software

} Hardware

↑

Parallelism present in every layer.

- multithreading

Superscalar

- Div. → 20 clock cycles for exec.

  2 diff. ALUs.

SIMD

- AVX → Intel and AMD introd in x86.
- Intel introduced AVX-512

  512 bit reg. partitioned into mult. 32 bit data

- but single instn. operated on all 32 bits

  a(31:0) + b(31:0)

  a(63:32) + b(63:32)

  ⋮

  } Same instn. but simd.

- Increase processor throughput

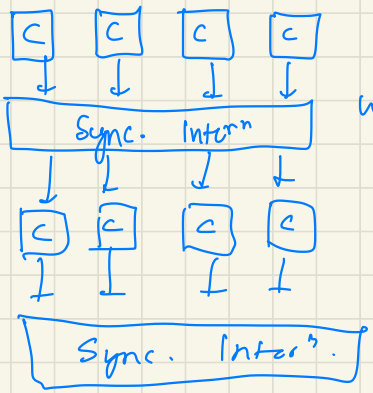
- Streaming SIMD Execution (SSE)

Multi-core Multi-processor OS

- Develop parallel-aware pgms

Parallelizn. paradigms

- Comm. among process is largely unnecessary
- Phase II[d] method



Wait till slowest task is performed.

- helpful in checkpointing
- Lag. Processor fails → int. memory → auto recovery.

→ for long running pgms.

Divide & Conquer

- Divide workloads and then merge

Pipeline data to increase throughput

Process Farm Method - only 1 time subdivision

Domain decomposition -

Desirable Attributes

- Concurrency
- Scalabel
- Data locality
- Modularity

Writing II^{al} codes

srun hostname
hostname
sinfo
srun -- node =2

Analyse the time complexity of parallel algorithms.

- Master method
  - div into a subprob. of size n/2.
  - f(n) → time to div and merge n/2.

$$T(n) = a\,T(n/b) + f(n)$$

- Compare $n^{\log_b a}$ with f(n)

Suppose $n^{\log_b a} >> f(n)$

$$T(n) = \Theta(n^{\log_b a})$$

eg. Matrix mult. by block mat. mult.

(3) $n^{\log_b a} << f(n)$

$$T(n) = O(n^2)$$

$C_{ij} = \sum a_{ij} \times b_{ij}$

→ sync. → synchronizn. points

Span of matrix multiplic^n. → $M_\infty(n) = M_\infty(n/2) + \Theta(1)$

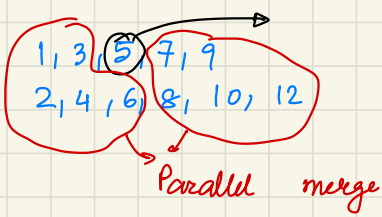
$= M_\infty(n/2) + \Theta(\lg n)$

$= \Theta(\lg^2 n)$

- Estimate amt. of parallelism present in an algorithm
  - Time for single thread exec. → $O(n^3)$ = work
  - Time for multi-threaded exec. → $O(\lg^2 n)$ = Span

Amount of parallelism = $\frac{O(\lg^2 n)}{O(n^3)} = \frac{\text{work}}{\text{Span}}$

Parallel Merge of Merge Sort



- Span of P_Merge

Work

Span $T_\infty(n) = T_\infty(n/2) + \Theta(\lg^2 n)$

- Cilk code
- Recursive construction

Ethernet

- Widely used LAN technology
- Operates in data-link layer and PHY layer
- Supports data b/w of 10, 100, 1000, 10,000, 40,000 and 100,000 Mbps (100Gbps)

$$T_\infty(n) = 5\,T_\infty(n/3) + O(1)$$

$$n^{\log_b a} = n^{\log_3 5}$$

$$\Theta(n^{\log_3 5})$$

$$\frac{T_1(n)}{T_\infty(n)} = \text{Parallelism} = \frac{O(n^2)}{O(n^{\log_3 5})} = O\left(n^{2-\log_3 5}\right)$$

- logical core, physical core
- 2 threads / more in one core.
- I Cache, D Cache — Instrn. Cache, Data Cache.
- L1, L2 Cache local to chip
- L3 Cache common to all core

Write pgms using shared mem. concepts → OpenMp

Processor – logical address (generate ↑) to physical address (conv. by MMU)

then proc. checks whether data is in L1 Cache

- data stored in main mem. in page mem. (4Kb on Sir's system)
- Blocks (64) gen. in mem.
- Inf. needed → entire block sent to cache.

- DMA is responsible for transfer data from secondary mem. to main mem.

Inclusive Cache | Non-Inclusive Cache

((L1 in) L2 in) L3

- Compiler directive → #pragma omp parallel → after this parallel starts.

```
omp_get_max_threads();
omp_set_num_threads(4);
```

master thread → only 1

#pragma omp barrier → Sync. point
all threads running should complete exec^n.

- Find area of a square by subdividing into various sub squares.
  → false sharing

Parallel Computers Memory Architecture

Shared memory (UMA)

Non-uniform Shared Mem. (NUMA)

Distributed memory

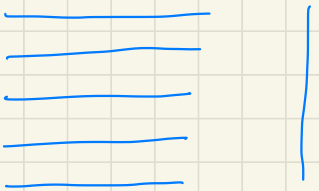
Hybrid mem NUMA + Distr. mem.

MPI

- MPI employs processes to do parallel tasks. Any process contains of a separate address space.
- A process may have multiple threads.

1. Prefix sum problem - Brielock prefix selection
2. Merge sort problem
3. Matrix Addn.
4. Matrix Multiplicn.

① Matrix addition.



Physics Assisted digital Twins for Complex Systems

- Explicit large Time stepper
- Tracking slow manifold
- PDE solving.

$$\frac{\partial^2 p}{\partial x_i^2} = \frac{\partial}{\partial x_i}\left(u_j \frac{\partial u_i}{\partial x_i}\right)$$

↳ Small scale flow → non-linearities
↳ → Reynold's number

- Parallelize non-linear flow.
- Cost of DNS $\sim Re^3$.
  ↳ compute
- Pipe 10^3 - Re Compute power - tera flop
  Automobile - $10^{12} \times 100$
- Reynolds - Assisted Navier stokes (int. particles)

- Large eddy Simul^n. (LES)

- Large scale energy systems → complex systems.

- Real physical systems 3% eff → 100b tCO_2 savings

Semiconductor manufacturing

- Plasma chem., multiscale process.
- comp. simul^{ns} to model real bodies

Cardiovascular diagonasis protocol

- Cheaper more reliable devices,
- pulmonary artery, lipid deposition of plaque.

- ECG and TMT → but very little info.
- CT-Scan → structural info. not func. info.
  - how it looks
  - measurements eg pressure

- Angiogram → invasive but risky
  - Invasive FFR → measure pressure ratio before and after block.

Take CT Scan images and try to model pressures in diff. parts of heart.

Computational Physics

- Numerical methods
  - math. schemes to solve problems
  - Int. or diff.

- Root finding
- Optimizn.
- Linear algebra – ideal value finding
- DE – Navier Stokes,

- Numerical simulns.
  - Perform sim. inside the computer
  - math. models / eqns.
  - Modelling and mimicking

- Nucleation problem
  - don't perturbate it → supercooled state, metastable state.
    → finite perb.

- microstates given a state (n, V, T) → ensemble theory
- fix accessible microstates.
- average all → ensemble average given a microstate.

- Monte - Carlo simulatn. → generate eqlb. configurns by drawing out from a distr.
  use molecular dynamics
- (N, V, E) ensemble to define a state.

Intermn. Potential

- Lennard Jones potn. – short range potn.
- Soft repulsive, short attr^n.

scale for distance

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right]$$

scale for energy

- Fix bdry points to declare 0, due to asymp.
- Chopping → non diff. forces → non-realistic → use quadratic.

Implemⁿ.

- Initializⁿ.

for a solid
- → posⁿ. – say simple cubic
- – define a box length, gen. periodic bdry.
- –

- for a liquid – generate a random (→ uniform) no. for a positn.
  – non-crystalline systems
- But if high den. particles come closer → force becomes repulsive → start with Fcl (high. pckg frac.) and increase vel. till it melts.
- Just ensure no two part. overlap each other or $< \sigma$.

Velocity initializn.

- follows Maxwell Boltzmann distrbn.

$$f(v) = \left(\frac{m}{2\pi K_B T}\right)^{\frac{1}{2}} e^{\left(-\frac{mv^2}{2K_B T}\right)}$$

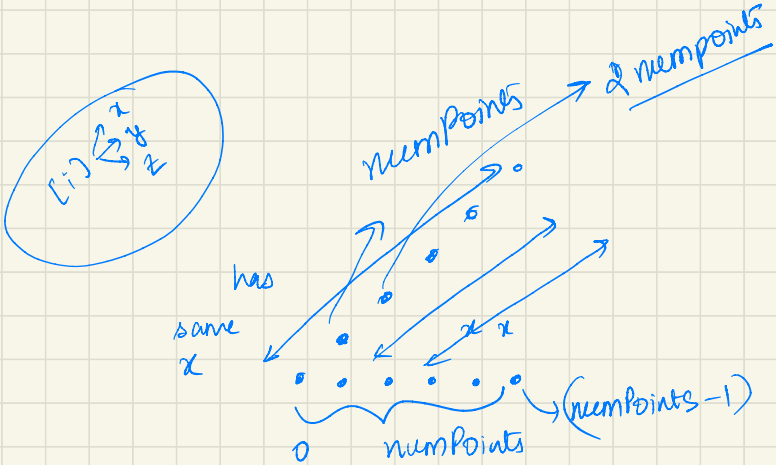
↑ Gaussian distrbn.

variance $K_B T$

- Generally periodical boundaries, due to

Radial distrbn. and Co-ordin. number

No. of particles	Temperature	Rho	Time
4.0	1.0	0.1	0.17
4.0	16.0	0.5	0.20
16.0	1.5	0.5	2.14
32.0	1.5	0.5	7.13
64.0	1.5	0.5	17.21
128.0			35.04
256.0			82.82
512.0			212.02
1024.0			618.76



→

Real systems in experiment ~ Avogadro no.

→ capture inherent corrln. length

Unit of energy - ϵ (depth of potn.
length = σ (dia) of part.)
mass = m (mass of part.)

Velocity - Verle Algo

in simuln.

$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2$

RK4 method

2nd order RK

Kind of Taylor series.

$x(t+h) = x(t) + k_2 h$

2nd order Range - Kutta

$k_1 = f_1(x, t)$

$k_2 = f\left(x\left(t+\frac{h}{2}\right), t+\frac{h}{2}\right)$

→ Energy not conserved

RK4 does not

Verlet algo. { Use here

$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2$

$v(t+\Delta t) = v(t) + \frac{f(t+\Delta t) + f(t)}{2m}\Delta t$

Profiling and Optimization

- Algorithmic Analysis → which part takes most time.
  Optimizn. → reduce that time.

- Disable the I/O, eg. outputting thermodynamic values.
- Disable the optimizn. flags
- Fix the time (max. timesteps) and timestep
- Vary the parameters - no. of particles, density cores.

- time -p
  → sys time, user time, clock time

Profiling tools

- Serial code opt.

```
gcc -pg -no-pie -fno-builtin program.c
```

- Run the code → out → gmon.out

- gprof gmon.out

Optimizn. → Nearest neigh. list

- Finding neighbour list, how often update neigh. list.

- Compute neighbour list $n = 2.5 + \sigma$ (↑ skin.)

- If the dist. travelled $|r(t) - r(t+\Delta t)| = \Delta r$

- if $\max_{ri}(r_i) > \frac{\text{skin dist.}}{2}$, update the neighbour list.

Fast Parallel Algorithms for Short-Range Molecular Dynamics

- Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS)

Atom decomp. algo - N/P atoms - each processor

- Commn. overhead → problematic
  ↳ $(\log_2 P)$ → optimal

P → cores.

- Neighbour list is shared to all the processor.
- update shared only with all the cores.

- Force decomposition
- Spatial decomposition

Advanced Inputs

- Radial Correlation → prob. at what distance of finding another particle

Assignments

1. Implement using neighbourlist.

1. Matrix Multiplication

a. Create 2 Matrix of size (n x n).
b. Create a pointer to the 3rd matrix.
c. Scatter the inf. to sub processes from the root (0).
d. Gather the inf. at the root and concatenate the result.

Simp. 1D matrix (1 x n) and (n x 1)

↓

n processes → buff of size 2

a b

Returns a * b

Accelerator architectures and Computn. Off loading

T_{Exe} = Inston. / program x CPI x T_{cycle}

CPI :- cycles per inston.

Inston. / program :- choose some intelligent ISA arch. / compiler

CPI :- about 0.5 for superscalar arch. + pipelining

Ahmdal's law

Speedup → bottlenecked by serial portion of code.

f :- fraction parallelized

(1-f) :- serial code

$$\text{Speedup} = \frac{1}{(1-f)+(f/s)}$$

S :- speedup $= \frac{\text{new time}}{\text{old time}}$

$\max\limits_{s \to \infty} (\text{Speedup}) = \frac{1}{1-f}$ → serial code

if $f = 2\%$ then even on inf. resources,

$\max\limits_{s \to \infty} (\text{Speedup}) = 2$

→ diminishing return on increasing resources.

$\text{Efficiency} = \frac{\text{Speedup } (S(n) = T(1)/T(n))}{n}$

$\text{Eff. } (E(n)) = \frac{T(1)}{n\,T(n)}$

Gustafson's law

Π :- fon. of parallel part.

S → serial code, execn. time

n :- no. of processors

Πn :- total

$S_s(n) = (S + \Pi n)\,/\,(S + \Pi)$

→ Scaling up the problem size as you increase the parallizatn. resources.
→ Serializ^n. does not scale up.

Specialized Hardware Resources - Accelerators

- Higher efficiency to tailor to a specific purpose.

Why Domain Specific Architechtures

- Ineff. of high level languages
  ↳ C + parallel loops + memory optimizn. + SIMD instrns.

- optimizn. towards specific applicns.

Domain Specific Architechtures (DSA)
Domain Specific Languages (DSL)

Best combn. → (DSA + DSL + Automatn.)

How to design domain Specific Accelerators

- Eff. memory design
- Parallelism
- Redn of overhead
- Specialized Ops. – over data type

Parallelism

- make few global memory
- make few PE - cross data movement.
-

Eff. mem. system

- Specialized mem. system.
- reduce point-to-point commn.
- Arithmatic opern is free, Memory is expensive.

Add → 3 memory op. → high mem. access.

- Storing key data struct. on local cache memory
- do data compression eg. storing in sparse matrix.
- gen. DL workloads → matrix

Sparse matrix → reduce some of the elements to 0.

- data reuse - Once stored in local memory from global memory, reuse it as much.

eg. mat1 → r1 r2 r3 ⋮ rn

c1 c2 c3 .. cn

store colomn matrix as row matrix → contiguous allocn. and reuse it.

CGRA → point-to point commn. b/w diff. processors.

Reduction of overhead

- Sp. hardware
- out-of-order execution reduces 99% overhead.

- segn. code to CPU and parallel code to sp. hardware like CGRAs.
- pragma call to compiler

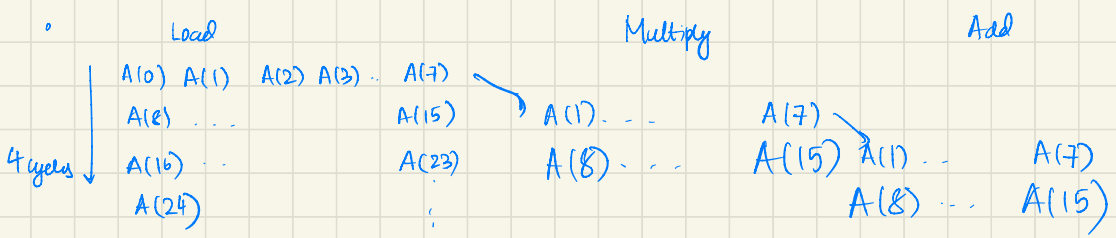
- only commn. to the CPU is whether done or not.

- CPU → hosts, gen. purpose computing
- minimize memory footprint.
- data comp., sparsity
- lots of fasts, on-chip memories.

- part of execution pipeline → tightly coupled / loosely coupled.

Single instruction multiple data:
- Array processors → different cases
- Vector processors → specialized hardware for performing its own op?

- Memory banks → to indep. access data.
- Lane wise execution manner.
- Vector Instruction Level Parallelism



VLIW arch. and ELI-512.

Instruction stream

- LD VR A(3:0)

- disadvantages in Array / Vector processors → data dependancies.
- VLIW → Compiler sees the code and creates the instns.

Automatic Code Vectorizn.

- Scalar Sequ. instn. (each step is indep.)
- Compiler sees this and does code vectorizn.

Vector instrn. ← load A(1) load A(2) load A(3) ...
← load B(1) load B(2) load B(3)

- Vector masks -

SIMD

- acts on multiple data at once
- Short arithmatic ops
- Vector instrns.
- must modify ALU to eliminate carry values.

SIMD in modern accelerators

- Cerebras WSEngine
  - neural layers mapped to diff. layers
  - MIMD machine → each tile in a die in a wafer
  - distributed memory, not shared
  - 4 way - SIMD for 16
  - 48 - KB local SRAM
  - GPUs are SIMD machines

- VLIW
- thread → can threads
- stalls in pipeline

SPMP

- A[i] + B[i] = C[i]
  - SIMD extrn. (either vector or array)
  - Thread for each iterⁿ.
  - Diff. b/w SIMD machine or MIMD machine

- Vector code
- forced
-

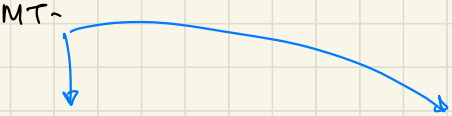
GPU is a SIMD (SIMT) Machine

- SIMT → prog. using threads
- thread → executes same code but diff. data

- Multiple threads → warp (wavefront) grouping threads

- SIMD opn. formed by hardware → warp
- Warp → set of threads perform same instrn.

- SIMD - each instrn.
- SIMT -



MIMD processing

- Fine Grained MThr. of warps

- Group scalar thread into warps
- branch div.
- SIMD utilizn.

- Dynamic Warp Formatn.

An Example GPU

- Tensor cores (for deep learning)
- PCI Express for connecting to CPU
- Fp16 or Fp 32 cores
- Tensor Core Architecture

- 4x4 matrix mult. → 1 cycle

NVIDIA A100

- 108 cores, tensor cores, 64 func. units per core
- New floating point data type (TF 32)
- Tensor cores have sparse mat. supp.
- Red. mem - comp by calculn. in that core itself.

- NVIDIA nsight CLI profiler
  → what benifits
  → exp. varying block sizes & dim^{n}s

- CUDA programming

Artificial Intelligence

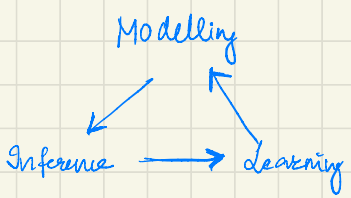
AI → defn.
- Represent^n. that will play / support models targeted to thinking, percept^n/ action

- All about making models →
- constraints →

Also AI →
- model with constraints + represent^n. that support.
- algorithm

model → constraints → represent^n. → algorithms → soln.

- Problems → solutions diff. ways.



Modelling - simplific^n., precise math. expression.

Inference - asking qns., and get the soln.

Learning - (data + model with param)

parameters are learnt from the data.

Data → model

+

ask qns. on data that hasn't been trained.

- leap of faith on the model. - generalizn of machine learning

Reflex based intelligence

eg linear classifiers, deep neural nets

→ fully feed-forward.

State based models eg.

- search based problems
- Markov decision based prob. (Blackjack)
- Adverserial games

Variable based models

eg. Constraint satisfn. problems

Bayesian networks

Optimizn. - discrete optm.

→ $\min_{p \in \text{paths}}$ Distance(p)

Algorithmic app - dynamic programming

iiy a cont. optimizn → $\min_{w \in R^d}$ Train. error

Algorithmic opt. - gradient descent.

a) Compute edit distance

Gen. principle

"cat" "dog"

↓

i) Complete present

ii) Substrig present

iii) Insertn. needed before / above

iv) Insertn. needed in between.

v) Deletion of some characters

vi) Deletion of some charact. + insertn of some char.

ab cd

4 10 → (1)

= =

(1) a) extract common characters

b) delete from both ends

insertn. + delm. (subs.) also

= + 9 + 3 ↙ least cost.

General principle - Reduce the problem.

- dynamic programming
  - reduce the problem size
  - match the problem

dynamic progran

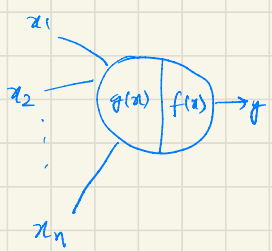
Introduction to Deep learning

→ Mc Cullogh Pitts neuron

$$g(x) = \sum_{i=0}^{n} x_i$$ ← boolean values

$$f(g(x)) = \begin{cases} 1 & g(x) \geq \theta \\ 0 & g(x) < \theta \end{cases}$$

$f(g(x))$ = output = y ↳ boolean



→ feedforward - not feedback loop provided.

Perceptrons

- generalized version of MP neurons
-

$$g(x) = \sum_{i=1}^{n} (w_i x_i)$$

→ linear eqn. in n dimension (line) ↗

draw a straight line boundary b/w P and N.

- weights to provide some preference
- inputs can be integer values as well.
- Perceptron learning Algorithm

  if $x \in P$ & $w \cdot x < 0$ $\quad w = w + x$

  if $x \in N$ & $w \cdot x > 0$ $\quad w = w - x$

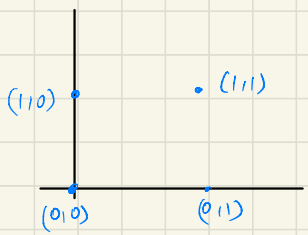
XOR conundrum

→ $w_0 < 0$

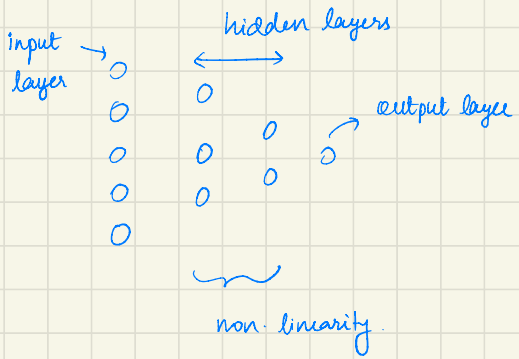
$w_1 > -w_0$

$w_2 > -w_0$

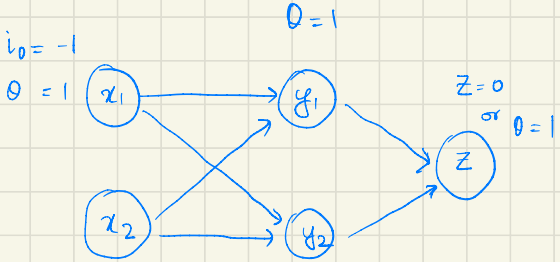
$w_1 + w_2 < -w_0$



Multi-Layer Perceptrons



Multi-layer Perceptrons



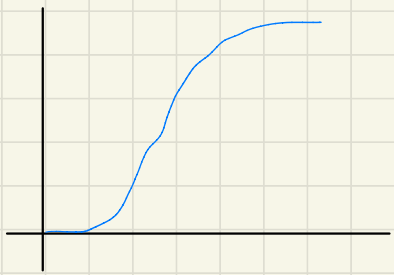
Theorem - Any boolean fxn. of n i/ps can be exactly represented with 1 input layers with n neurons + 1 hidden layer (2^n neu)

+ 1 output layer (1 neuron).

Need for activ^n functions

Sigmoid neurons → smoother trans^n.

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



- Output is no longer a binary values but
- eg. ReLU :- $y = \max(0, z)$
  Leaky ReLU :- $y = \max(\alpha z, z)$
  Exp. linear decay (ELU) $y = \max(\alpha(e^z - 1), z)$ $\alpha > 0$
  Tanh.

all these fns need to be differentiable

Feedforward Neural Networks

- multi-layer perceptron (multiple layer perceptron).
- f^* fn. approx. poss.
- neurons are formed by a directed acyclic graph.

Gradient Descent

- Loss_{MSE} =

$$= \frac{1}{M} \sum_{i=1}^{M} \left(f^*(x) - f(x_i \theta) \right)^2$$

mean-square error.

- $\min \left(f(x) = x^2 \right)$
- if neg. slope go ahead, $x + \frac{df}{dx}$
- if pos. slope go behind, $x - (-1)\frac{df}{dx}$

Neg. gradient

Why? $L(\theta) \rightarrow$ loss gradient

- Random vector, calculate
- $\theta_i^{next} = \theta_i^{curr} - \eta \left(\frac{\delta L}{\delta \theta_i^{curr}} \right)$

η: Learning rate

L: (loss function)

- η :- hyperparameter here
- Compute gradient of loss function

n_i layers in the network, $l = 1, 2, 3, 4, \ldots, n$.

$z \quad a = f(z)$

f:- activ^n functn.

$a_1^{(1)}$ $a_2^{(1)}$ $a_2^{(1)}$ $a_2^{(1)}$

Backward pass

→ Trying to compute

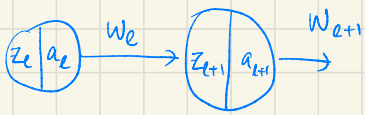
$$\frac{\partial L}{\partial w_1} = \left(\frac{\partial L}{\partial a_i} \cdot \frac{\partial a_i}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}\right)$$

$\nabla_{w^{(l)}} L$

δ^{l+1}

$\hookrightarrow f(z) = (a^l)^T$

$$\frac{\partial L}{\partial w} =$$



use backpropagⁿ. to compute $\nabla_{\theta^l} L(\theta; x; y)$

θ^l → w^l, b^l

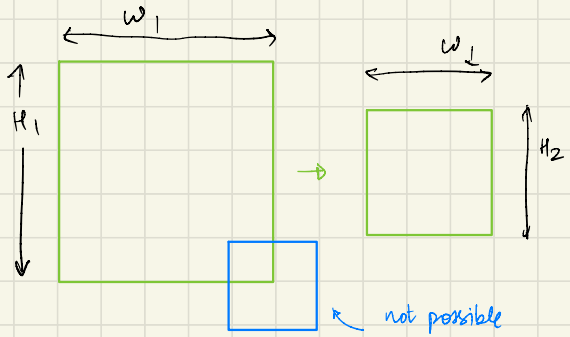
Update the

$$w^l = w^l - \eta \left[\frac{1}{M} \Delta w^l\right]$$

- Consider a simple FFNN (or MLP)
  - pytorch, tensorflow, Keras

Convolutional Neural Networks

- $$Y(i,j) = \sum_{u=-K}^{K} \sum_{v=-K}^{K} w(u,v)\, X(i-u, j-v)$$
- for K filters → K channels in output feature map.
- Width (W), Height (H) and Depth (D_1 = same as no. of channels of ip.)
  output = $W_2 \times H_2 \times D_2$
- Basically size reduction



$W_2 = W_1 - F + 1$

$H_2 = H_1 - F + 1$

- If we want i/p size = o/p size → padding

Now formula

$W_2 = W_1 - f + 2P + 1$

$H_2 = H_1 - f + 2P + 1$

} Padd. at both ends

Striding → skipping some pixels

With stride $W_2 = \frac{W_1 - F + 2P}{S} + 1$

$H_2 = \frac{H_1 - F + 2P}{S} + 1$

$W_1 = 227$ $F = 11$

$H_1 = 227$ $96 = n$

$D_1 = 3$

$W_2 = 55$

$D_2 = 96$ $H_2 = 55$

Why Convol^n.

Trad^{nl} ML based Vision

- Edge - Detector
- SIFT - HOG
- LBP

} Filter → static feature extrn.

- Classific^n. → SVM, Perceptron classifier

- Instead of handcrafted filters, learn meaningful kernels / filters in
- filters for feature extn.
- Add kernels as parameters, and using back-propgn.

Convolut^{n}al neural nets

- Challenges of applying FNNs to MNIST.
  - 2% degradn.
  - Ignores spatial structure, pixels that are spatially separate are treated the same way as other pixels.
  - IMP → 60 million parameters in 1st layer.

local receptive fields
→ Convolution captures local spatial relns.
→ transl^{n}al invariance of filter → weight sharing.

Pooling layer → condenses info. more

Local receptive fields

→ in CNNs, conn^{n}s are much sparser in this case in comp-to fully connected NN. (FNNs).

Is sparse connectivity a really good thing?
Not really, losing any inf.

Weight sharing

→

Pooling layer

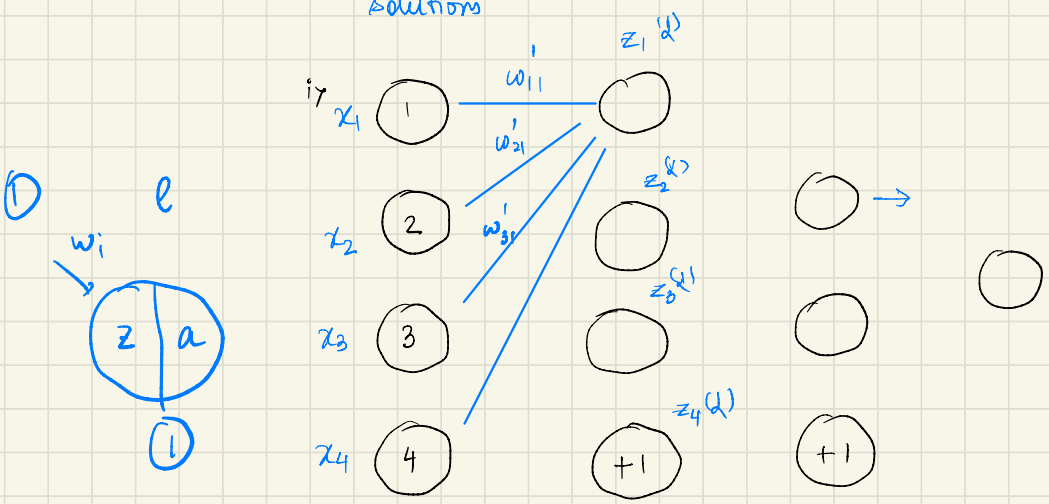
→ down sampling
→ or average pooling
→ diff b/w max pool and convolut^n. ? ? ?

} generally all filter conv. is done in freq. domain.

Transpose Convolution

- Allows for deconvolution or learnable upscaling
- 3D convol^n, 1×1 Pointwise Convol^n, Grouped Convol^n.

Solutions



① Solve XOR problem for 2 hidden layers
② Solve for the weights for all the weights in the above problem.

① st.

Ans. $z_1^{(2)} = \sum_{i=1}^{4} \sum_{j=1}^{3} w_{ij}^{(1)} x_i + b_i$

So, $z^{(2)} = W^{(1)T} X + B_1$ → Matrix eqn. of 1st layer

$$\frac{\partial z^{(2)}}{\partial w_{ij}} = \frac{\partial w^{(1)}}{\partial w_{ij}}^T X + \cancel{\frac{\partial X}{\partial w_{ij}} w^{(1)T}} \to 0$$

$$= \frac{\partial}{\partial w_{ij}} \left(w^1 X \right) + \cancel{\frac{\partial B_1}{\partial w_{ij}}} \to 0$$

$$\frac{\partial z^{(2)}}{\partial w_{ij}} = \begin{bmatrix} 0 \\ 0 \\ x_i \\ 0 \end{bmatrix} \leftarrow j^{th} \text{ posn.} \qquad = \qquad x_i \text{ in } j^{th} \text{ posn.}$$

lets say, $X_{ij} =$

$$\begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 & w_{14}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 & w_{24}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 & w_{34}^1 \\ w_{41}^1 & w_{42}^1 & w_{43}^1 & w_{44}^1 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \to z^1 \text{ or } a^1$$

4×4 $\quad$ 4×1

$w^{(1)}$ $\qquad$ $z^{(1)}$

$$a^2 = f^2(z^2)$$

$$\frac{\partial a^2}{\partial w_{ij}^{(1)}} = \frac{\partial f^2(z^2)}{\partial w_{ij}}$$

$$= \frac{\partial f^{(2)}(z^{(2)})}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_{ij}}$$

$$= \frac{\partial f^{(2)}(z^{(2)})}{\partial z^{(2)}} X_{ij} = f^{2\prime}(z) X_{ij}$$

$$z^{(3)} = w^{(2)} a^{(2)} + b^{(2)}$$

$z^y \quad w_{ij}^x$

$x \times y$

$$\frac{\partial z^{(3)}}{\partial w_{ij}^{(1)}} = \frac{\partial w^{(2)}}{\partial w_{ij}^{(1)}} \cdot a^{(2)} + 0 + w^{(2)} \frac{\partial a^{(2)}}{\partial w_{ij}^{(1)}}$$

any one of them $= 0$

$$= \frac{\partial z^{y+1}}{\partial w_{ij}^x} = \frac{\partial w^y}{\partial w_{ij}^x} a^y + w^y \frac{\partial a^y}{\partial w_{ij}^x}$$

$$\frac{\partial a^{y}}{\partial w_{ij}^{x}} = \frac{\partial f(z)^{y}}{\partial z} \times \left(\frac{\partial w^{y-1}}{\partial w_{ij}^{x}} a^{y-1} + w^{y-1} \frac{\partial a^{y-1}}{\partial w_{ij}^{x}} \right)$$

$$= h^{y}(\theta; x)$$

$$a^{y} = f^{y}\left(w^{y-1} a^{y-1} + b^{y-1} \right)$$

$$L(\theta) = \frac{1}{M} \sum_{i=1}^{M} L(\theta; x^{i}, y^{i})$$

$$= \frac{1}{2M} \sum_{i=1}^{M} \| h_{\theta}(x^{i}) - y^{i} \|^{2}$$

$$L(\theta) = \frac{1}{2M} \sum_{i=1}^{M} \| a^{f_e}(x^{i}) - y_i \|^{2}$$

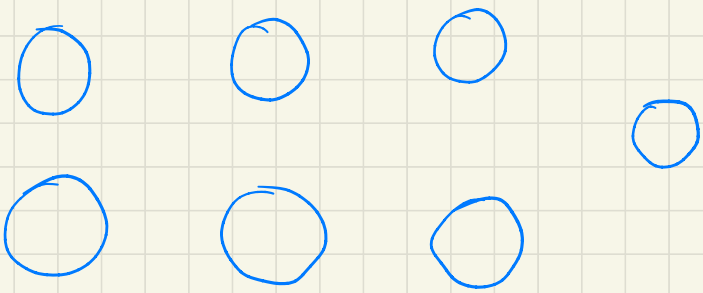
$$L(\theta) = \frac{1}{M} \sum_{i=1}^{M} \| a^{f_e}(x^{i}) - y_i \|$$

$$L(\theta) = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \left(a^{f_e}(x^{i}) - y_i \right)^{2}}$$

$$\frac{\partial L(\theta)}{\partial w_{ij}^{x}} = \frac{1}{2} \times 2 \left(\frac{1}{M} \sum_{i=1}^{M} \left(a^{f_e}(x^{i}) - y_i \right)^{2} \right)^{-\frac{1}{2}} \times \left(\frac{1}{M} \sum_{i=1}^{M} \frac{\partial a^{f_e}(x_i)}{\partial w_{ij}^{x}} \right)$$

and $$\frac{\partial a^{f_e}(x_i)}{\partial w_{ij}^{x}} = \frac{\partial f^{f_e}(z)}{\partial z} \left(\frac{\partial}{\partial w_{ij}^{x}} w^{(f_e-1)} a^{(f_e-1)} + w^{(f_e-1)} \frac{\partial}{\partial w_{ij}^{x}} \left(a^{(f_e-1)} \right) \right)$$

XOR Problem



Q i) Can we take any threshold values?

→ Yes

$\theta = 1$

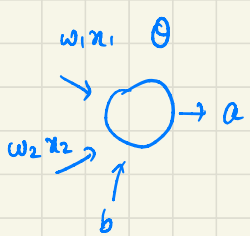
Q ii) How do we calculate weights and biases?

Two ways → diff. and find their local minima.

Else use existing solutions.

Way → go from behind → XOR = $AB' + A'B$

OR b/w two process

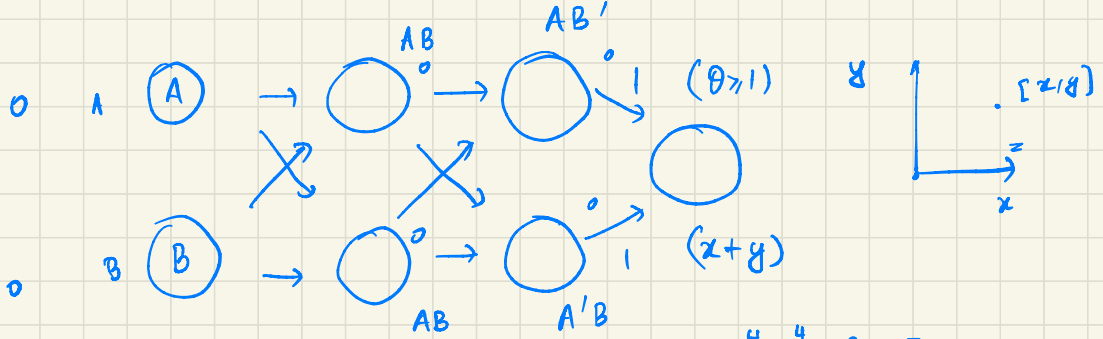


$a = (w_1x_1 + w_2x_2)$

$w_1 = w_2 = 1$

$= x_1 + x_2 = (0, 1, 1, 2)$

if $\theta > 1$ → logical or.



$\omega_1^4 = \omega_2^4 = [1\ 1\ 1]$

$\theta > 1$ $\quad$ $\theta > 1$ $\quad$ $\theta > 1$

$\omega_1^2 = \omega_2^2 = [0.5, 0.5]$ $\quad$ $\omega_1^3 = [1\ 1\ -1]$

$\omega_2^3 = [-1\ 1\ 1]$

Neighbour list calculation

$r_l = r_{skin} = r_c + \Delta R$

ΔR $\quad$ list of all particles within (r_s)

next 100 it,

$$A \oplus B = AB' + A'B$$