Lecture-08

Best-first Search : 1st algo of Informed Search.

→ যেই node/Vertex কে expand করতে চাই

→ কোন Vertex কে visit করবো

→ Must be evaluated

→ Evaluation function ***.

→ next-most promising node.

Diff algos based on evaluation function.

Heuristic func : Thumb Rule.

- No exact knowledge.
- Why 'thumb'?
- function এর input কোই কিভাবে জানি না।

prob related information in informed Search.

- computational Resource Budget is the limitation.
- Real Search tree not feasible
- Heuristic functions are used.

h(n) can be used to measure completeness

to be used as Additional information.

Optimal Path:

- Multiple paths
- choose the minimum cost.

linear distance

- 4 properties to evaluate a Search Algorithm:
  1. Complete: Solution exist করলে। আমার algo Solution টা reach করতে পারে কি না।
  2. Optimal:

Solution যতভাবে পাওয়া যেত তার মধ্যে Best solution পাওয়া যাচ্ছে কি না।

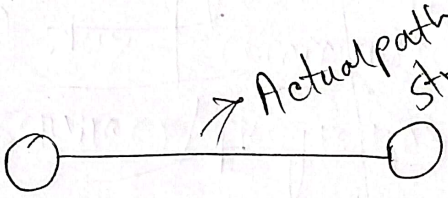
Eval func can be used to measure optimality.

→ Additional space to bookkeep ⇒ আরও complete হবে।

→ heuristic function কখনো Over-estimate করবে না।

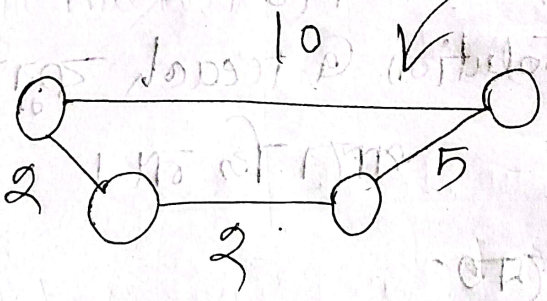
heuristic cost ≤ Actual Cost

Optimality : সবসময় decide করা যাবে না।
situation dependent.



$f(n) = h(n) + g(n)$

n থেকে goal



A* Search :

$f(n) = g(n) + h(n)$

Real যত cost হইসে তার summation

ধারণা করছি last state যাইতে কেমন cost হইতে পারে।

Admissible heuristics:

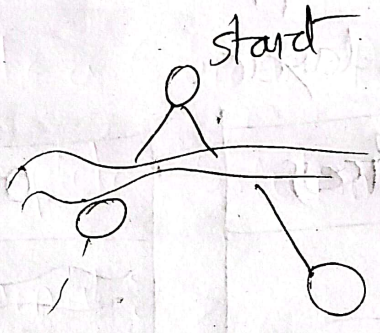
* Why A* search provides optimal solution?

* ≡ either true cost / optimal cost.

$\Rightarrow h(n) \leq h^*(n)$

SLD = straight line distance.

Proof:



optimal

G (goal)

(G_2 goal)

suboptimal

যদি G_2 goal হয়,

$f(G_2) = g(G_2)$ // $h(G_2) = 0$

$g(G_2) > g(G)$ G_2 suboptimal

$f(G) = g(G)$ // $h(G) = 0$

$\therefore f(G) = g(G) + 0$

$f(G_2) = g(G_2) + 0$

As, $g(G) < g(G_2)$

$\therefore f(G) < f(G_2)$

↑

$\therefore$ We'll take this path

API

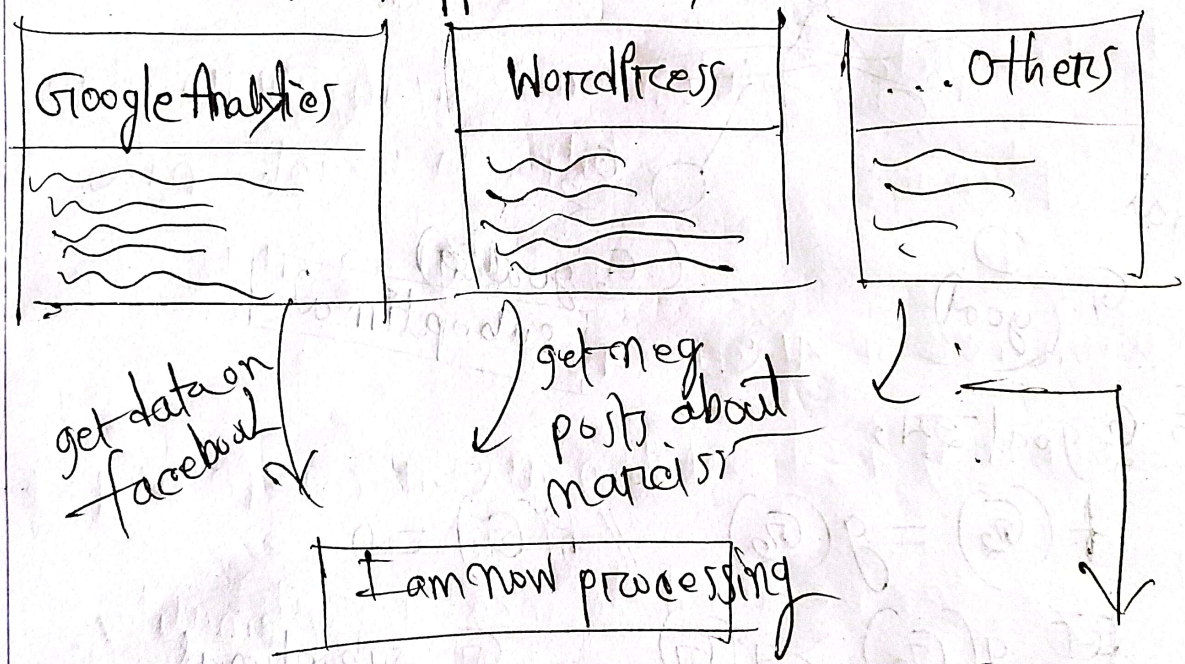
- What is API?

→ Application Programming Interface.

: A way of communication bet^n two different services/programs.

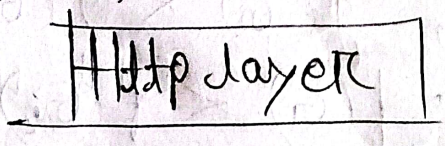
- Http API : Communication happens over the internet

→ Works on application layer.

→ 

এখন common
হয় API এর
মাধ্যমে।

API framework



Lecture-01

Local Search: (& Optimization)

→ 1টা single solution নিয়ে কাজ করি।
→ only one ,, is tracked.
→ current state থেকে neighboring state এ transition
↳ long jump হয় না.
→ final state এর দিকে যত যাবে → jump ধরেই ছোট.
Assuming: পাশাপাশি state এর অবস্থা almost same.
How to identify neighbor?
Co-relation between Solutions: ✓
Neighboring state কে measure করতে diff metrics.
→ Local Search অনেক memory ব্যবহার করে।
↑ Advantage
↓
অল্প সময়ে reasonable solution.

Optimization prob: prob statement এ superlative degree Word থাকবেই ***.
→ প্রতি state কে measure করতে objective func থাকতে হবে।

→ কোনো prob এ diff objective func থাকতে পারে।

↓
multi objective

$f1 > f2 > f3$

in terms of priority.

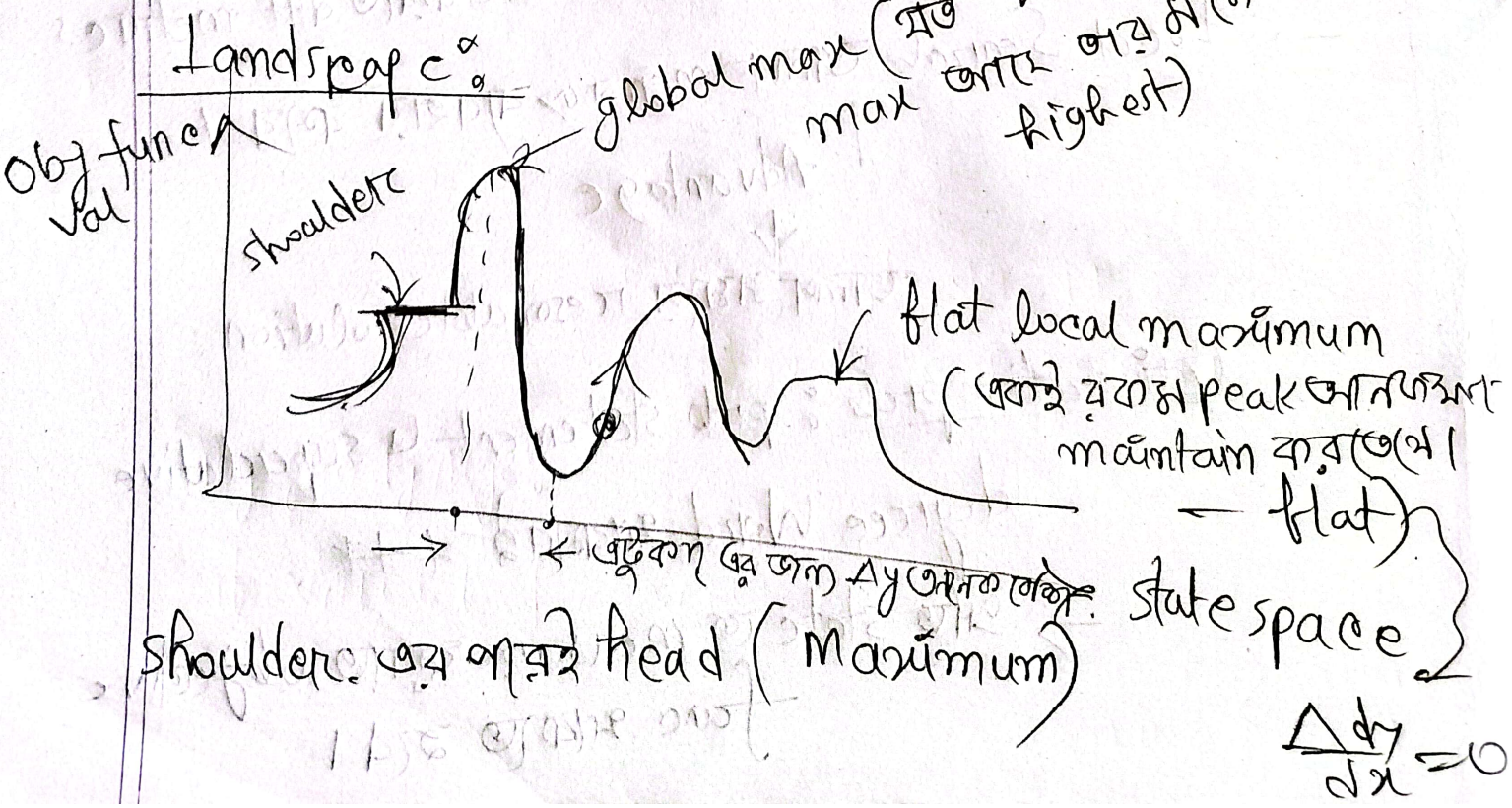
linear combination:

$= w_1 \times f_1 + w_2 \times f_2 + w_3 \times f_3$

where $w_1 >> w_2 >> w_3$.

→ Optimization prob° path is not the target.

Highest/lowest val is preferred over anything else.



$= x_{old} + d_i \times \eta_i$

$x_{new} = x_{old} + \text{direction} \times \text{step size}$

new state — current state — direction detect করার পর যেদিক কতটুকু যাবে

- Iteratively progress.
- Neighbor এ move ≡ step size ছোট হবে।
- Real probs always have limit.

• Search process এর শুরুর দিকে step size বড়
শেষের দিকে " " ছোট

(শেষের দিক) time ↑ budget ↓ step size ↓

(শুরুর দিক) time ↓ budget ↑ step size ↑

or # of iterations.

// let's say আমি বললাম 40 iterations maybe.
∴ যখন 35, 36 এ যাচ্ছি, আমি last এর দিকে।

Hessian func?

double derivative.

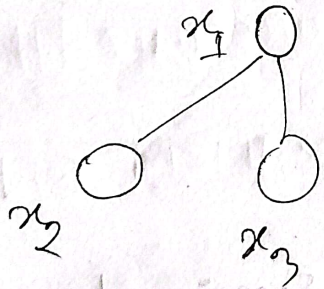
direction বের করতে।

Hill climbing Search:

- prob state → node.

আমি n_0 set করে দিব।

∴ current state থেকে যতগুলো state $\max n_0$ দূরত্ব, তারা " " এর children.



x_4 (if $\Delta(x_4 - x_4) > n_0$ max

current state এর সব successors. কে evaluate করতে হবে।

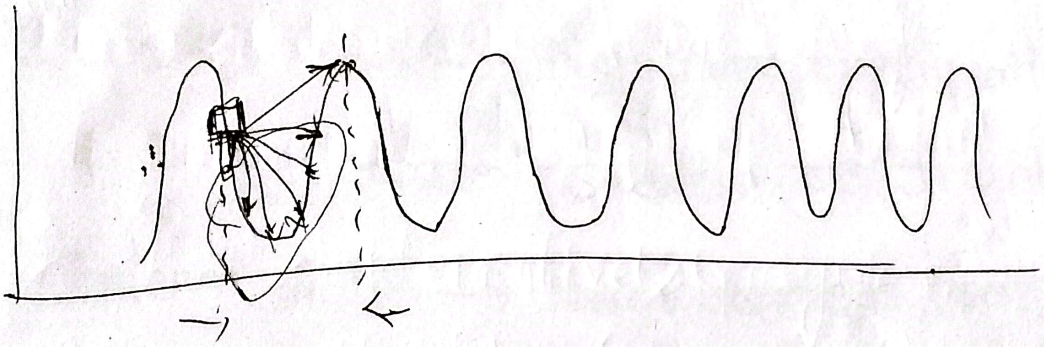
Why greedy Algo:

if VALUE[ ] ≤ VALUE( ). ← greedy Algorithm হলে use করবো।

Hill-climbing algo : // greedy

.. gradient Ascend algo. → উপরে উঠতে চাই

" descend " → নিচে নামতে চাই.



Lecture-2

Greedy: Why hill-climbing algo is a greedy...?

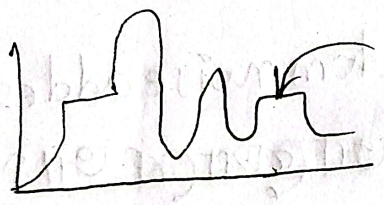
— greedy algo.

— We always move towards improvement.

local Search → Neighbor exploration.

শুধু current-solution কেই track করে।

Intermediate → Neighbor এর আশেপাশে. State.



trapped at local optima.

↓

বের করে আনার উপায় কী?

— Search process বার বার নিয়ে random initial point থেকে শুরু করবে।

যদি → i) Luck favor করে

ii) Computational budget থাকে.

আমরা confirm না → আগের মতোন trapped হবে না।

but আগেরটাও store রাখা লাগবে না।

→ Solution এর মধ্যে Random noise ✓

→ new Component add করলাম।

1st approach:
Random restart
আগের sol^n stored হবে না।

Constraint/motivation release
সবসময় improvement এর দিক না।

2nd Approach:
আগের sol^n stored + Random noise added.
এটাকে base করেই আমরা আরোপাকে আগাব।

3rd Approach: Simulated Annealing + greedy.
A good property about greedy algo
→ Convergence faster by searching/going for the best.

In a given computational budget →
greedy local optima তে trap করলে বের হয়ে আসতে পারে না।

A new algo:

convergence slow.
local optima থেকে
বের হতে পারবে।

greedy

convergence fast.
local optima থেকে
বের হতে পারবো না।

8-queen prob:

* Hill climbing Variation.

Drawback:

i) trapping prob. — ভয়ানক prob. (তাও used হয়
↓
convergence
property থাকায়)

ii) X

Stochastic: parent node $\xrightarrow[\text{func}}^{\text{successor}}$ child nodes.

যে best- তাকে নিবে

তবে best-এর চেয়ে → যে-কোনো 1 টা-best.
কম কয়টা ↳ Randomness.

Generation of Random Noise:

- distribution function.
  property — Overall space থেকে 10 টা pick করলাম → Uniformity দেখাতে হবে।
- Randomness কে to some extent control করতে পারবো by use of distributive function.

Evolutionary Algorithms

- Genetic Algorithm (GA)
- Genetic Programming (GP)
- Evolutionary Programming (EP)
- Evolutionary Strategy (ES)

Basic Diff :

1) population of Solution নিয়ে deal করছে।
∴ Multidirectional Search

2) Solution S_i যখন req fulfill করলো না
→ Random Restart না
→ নতুন A_j apply না.
→ Learned experience + evolutionary operator দিয়ে solution quality improvement করার try করি।

classical algo

1) 1 টা solution search

Unidirectional Search.

A_1
A_2
A_N

A_i → applied to P
=> if solution S_i satisfies my requirement, then

→ A_i এর কিছু param change (Random Restart)
OR → take A_j, apply on P
Requirement Satisfy? ok.

Best কি না করলাম নাই।

GA, GP

solution representation level

EP, ES

solution representation level

The diff is at operator level

GA	GP
solution = non-tree	solution = Tree

□ Genetic Algorithm: Goldberg এর বই ✱

natural selection
natural genetic
Survival of fittest

→ solution req fulfill না করলে কোন না দিয়ে improve করার try করা হয়

২টা operators → cross-over, mutation.

① Random Sol generation

② Evaluation : ১টা ও req fulfill করে না.

↓

selection operator.

// আমাদের objective 6bit-এর highest num (যার decimal val. বেশি)

// 6bit highest number — আমি ১টা expect করতেছি

Either right or not.

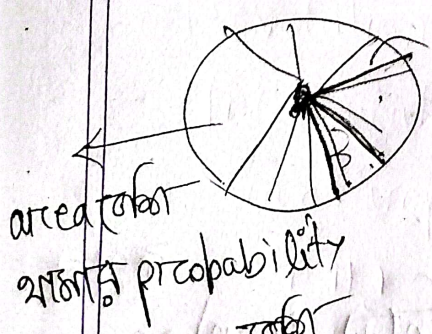
Parent Selection: কাদেরকে নিয়ে সামনে অগ্রসর হবে.

// threshold val এর উপর কার্যকে নিব না

// " " নিচে " " .

// আবার Randomness ও চাই.

Roulette wheel Selection.



area বেশি
থাকার probability
বেশি.

area কম
থাকার probability
কম

∴ fitness score অনুযায়ী —
we can assign area.
কিছু solution একবার কম
selected হতে পারে — তাদেরকে improve
করার try করবে — Mapping part.

applied upon parents.

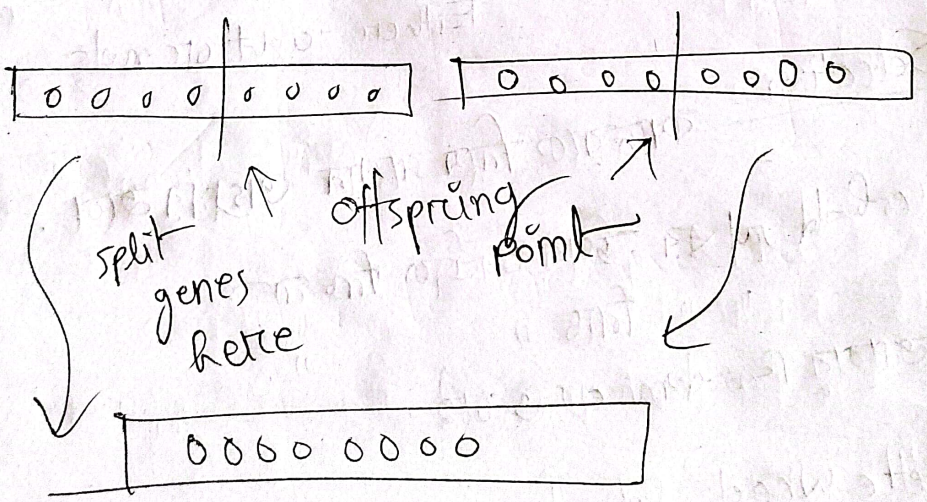
Operators: Crossover, improvement operators.

Parents

↙

children (from characteristics inherit করে + Environmental influence.

simulation করে যে crossover apply করব।



☐ mutation operators: applied on children

Replacement: from parent যেমন নেব

" offspring " " .

parent-এর কোনো কোনো offspring দিয়ে replace

Result

1. crossover : আগের changed
   mutation : ছবি আগের "

2. crossover applied on children
   mutation " " offspring.

Lecture –

Operators.

1. Fitness based selection.
2. Crossover / Recombination.
3. Mutation.

Search algorithms main components:

* Balancing between exploration and exploitation.

Which point to consider for cross-over?

let, কোন prob এর Solution যদি number হয়

যদি → recombination operator can easily be used

অযদি → binary → decimal
decimal → binary
২ বার conversion

Ex. 7.3 9.2 5.4

→ binary representation ✓

→ crossover ✓

* cross-over যে-কোনো অবস্থায় use করা যাবে (decimal point এর left / right এ)

আমরা real coded আছে যাচ্ছি binary representation নাই।

x_1 (২.3) x_2 (9.2)

$$x' = \frac{x_1 + x_2}{2} * \mu$$ ← constant factor (varies between 0 & 1)

$$x'' = (1-\mu) * \frac{x_1 + x_2}{2}$$

২টা parent কে into carry করবে।

- Mutual operator applied on offspring.

$$-0.01 \sim 0.01$$

$$x' = x' \pm \eta$$

- crossover দিয়ে offspring

~~১ টা parent দিয়ে~~

↓

disrupting operator

বড় বড় পরিবর্তন

fine-tuning operator

ছোট পরিবর্তন

Search space: প্রথমে from exploration
থেকে » exploitation.

big changes are allowed

big changes are not allowed.

Probability:

starting cross ↑ mutation ↓
ending cross ↓ mutation ↑

can be controlled by probability.

—— end of GA.

Simulated Annealing

↓

heat bath এ solid টিকে রেখে temperature high, then gradually cool করা।

↓ দ্রুত ভাবে cool করলে irregular crystal.

Solid — Iron/Metal ~(1000°C)

Normal property — বাঁকানো যায় না.

উত্তপ্ত — গলে যায় (atom mobility high)

then নিজের মতো জিনিস বানায় নেয়া যায়।

* Metal এর property

→ শক্ত —reverse→ নরম —reverse form→ শক্ত.

→ **Algo:**

$x_1 \xrightarrow{a} x_2$, score of $x_2 = f_2$; score of $x_1 = f_1$

if x_2 is better than x_1.

$\therefore f_2$ " " " f_1.

যদি better হয় → 100% তবে μ-state হিসাবে Accept করার chance 100%.

Let,

Solid এর state	energy
i	E_i
j	E_j

SA	greedy
better? Accept: reject with probability.	better? Accept; Reject

Local minima is a bottleneck of greedy search algo.

⟶ This can be solved with SA.

∴ Local optima ও trap এড়াতে probability ব্যবহার।

* Search Algorithms ⟶ GA এবং।

Lecture-

Simulated Annealing :

if $E_j \leq E_i$

$P(\text{accept } S) = 1$ ← ২টা different energy point.

$\exp\left(\frac{E_i - E_j}{k_B T}\right)$ ← Temperature.

Boltzman Constant.

$x = x_1 . x_2 . x_3 . \ldots\ldots . x_n$

$f(x)$

• $f(x+\Delta x)$ // objective function এর value

২টা value কে ২ ভাবে compare করতে পারি।

→ ± difference

→ ratio

$P = \frac{\text{(যেটা চাই)}}{\text{,, + Actual টা}}$ Page = 8 ; eq (1.3)

$x^{(0)}$ → iteration number.

b) random number with probability dist^n function.

How to generate Δx in multidimensional space?

Random number generation এ pdf (probability distribution function) use করা হয়।

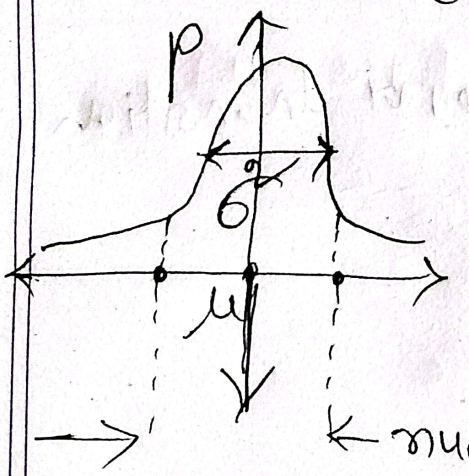
Distribution function has 3 characteristics.

1) Gaussian dist^n : (mean, variance).

Non-decreasing
right-continuous
normalized.

pdf → probability dist^n func.

μ = mean
σ = variance
of " ".



← number generate হবার probability সবচেয়ে বেশি (mean এর আশেপাশের)

যে jump-gaussian applicable না তখন cauchy dist^n func use করা হয়।

Δx বেশি ↑ variance ↑ এমন ধরণের function.

variance ↓ হলে value একসাথে — ছোট ছোট jump - কাছের দিকে গুলো।

" ↑ value গুলো scattered.

↓

বড় বড় jump

↑ দূরের দিকে ও গুলো use করতে হবে।

II **SA:** gradual cooling → cooling schedule?

i) initial temp

ii) final temp

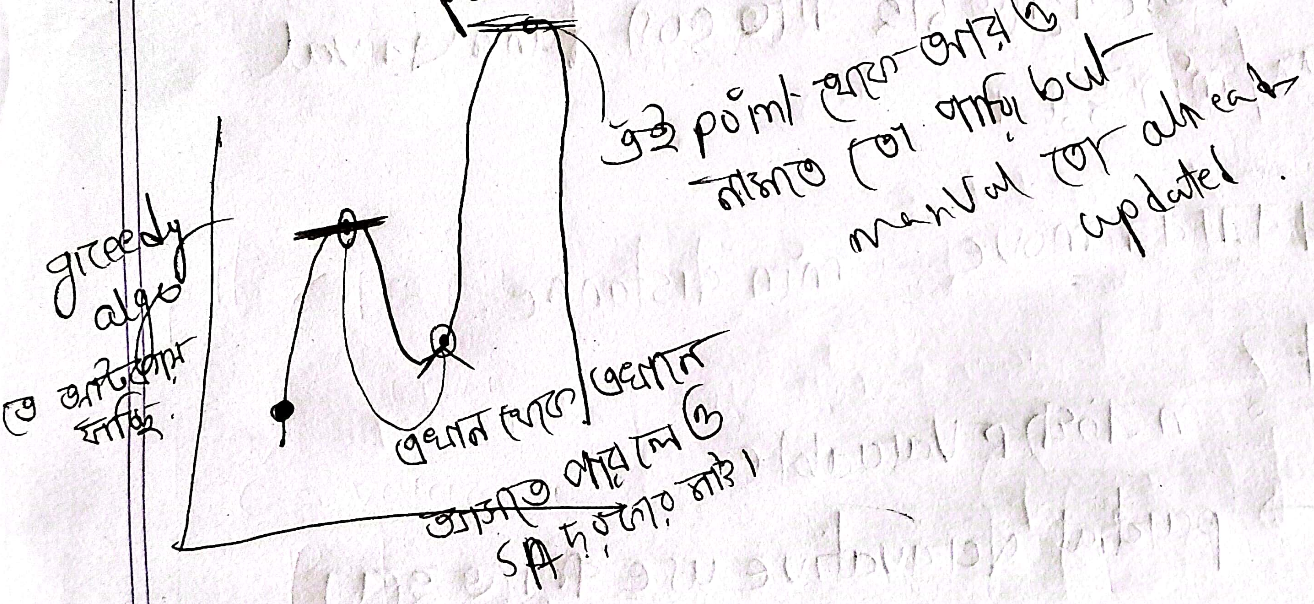
iii) rate of decrement

iv) 1 টা temperature এ, থেকে max কয়টা transition allowed.

A very

1) T_0 = High temperature
2) Decrement Rule / Procedure.
3) Stopping Criteria
4) No of state transition at each temperature.

User defined parameter.



Lecture-

Gradient Descent:

let, a cost function = C

যে function কে objectify করতে চাই = X

↓

variables = $x_1\ x_2\ x_3 \ldots\ldots x_n$.

Optimization: n সংখ্যক component এর এমন সব value choose করতে চাই যাতে cost function এর val max/min হয়।

মধ্য বরাবর move → min distance.

gradient? n সংখ্যক Variable

→ partial derivative use করতে হবে।

$$X_{new} = X_{old} - d_i \times \eta_i$$

minimize করতে চাই → '−'

gradient descent :. − হবে.

d_i → direction → $\frac{\partial C}{\partial x_i}$

η_i → step size

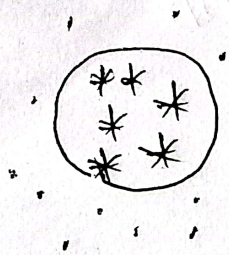
* attribute 1টা থাকলে complete derivati

Learning as Optimization

* Error decision boundary
* Learning algorithm – data driven.

Example কোন discriminative feature.

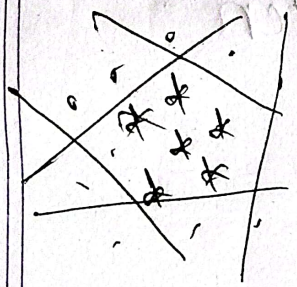
আলাদা করা হয়।



→ straight line দিয়ে আলাদা করা যাবে না।

→ Multiple lines can be used also

→ NN – একাধিক decision boundary (linear) combine করতে পারি



→ Non-linear decision boundary

attribute
class label

Dataset: $\{(x_1, c_1), (x_2, c_2), (x_3, c_3)\}$

Learning machine determines the relation between x_i and c_i.

* SVM → Marking concept (Decision boundary)

NN

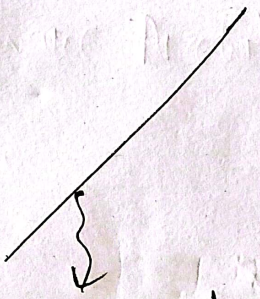
A func can be

→ sign $(W \cdot x)$

↓

learning model parameter

let, a decision boundary is



how we found this?

$y = mx + c$

↓ ↓

learning machine parameter

$f(x, W)$

↑

$\{m, c\}$

let, $y = f(x) = x^2 - 3x + 5$

↑

functional expression.

data fixed, you can't modify.

learning machine এর আউটপুট এবং relation বের করতে
W change করতে থাকে।

Error: learning machine Outcome — Actual Outcome.

W কে এমন ভাবে change করে যেন error minimized হয়।

kinda a curve fitting prb.

Intelligent Agent

Agent এমন এক entity যার ২ components থাকে যেমন—

HW architecture
- i) Sensor: Environment থেকে data perceive করে।
- ii) Actuator: Environment এ action perform করে।

Software Architecture: Agent program.

"intelligence part"

Rational agent:

যেই agent তার performance কে maximize করতে চেষ্টা করে।

performance metric থাকে।

Autonomous Agent:

Agent action depends on it's experience and perception (from sensor).

∴ learning component থাকতে হবে।

এটা হলে exp gain or use করতে পারবে না।

Non-deterministic / uncertain environment / preferentable.

Expert System's in 90's.

Knowledge থেকে knowledge generate করতে পারলেই intelligent.

Task Environment:

PEAS description.

- Performance
  - time to complete the course
- environment
- Actuators
- Sensors

Agent কে task basis এই create করতে হবে।

basic idea থাকতে লাগে।

- SA বাদ.
- Evolution Algo total বাদ
- A* Search বাদ x