Classify_New_Instance (x)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \; \hat{P}(v_j) \prod \hat{P}(a_i \mid v_j)$$

|V| should be small.
(# of classification states)

Play Tennis

$P(y)\ P(sun/y)\ P(cool/y)\ P(high/y)\ P(strong/y) = 0.005$

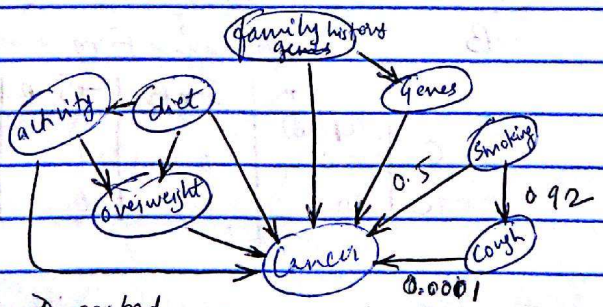
$P(n)\ P(sun/n)\ P(cool/n)\ P(high/n)\ P(strong/n) = 0.021$

↳ $V_{NB} = n.$

even if one probability is 0, then the whole probability = 0.
∴ use lots of data so that no probability is 0.
If it is 0, then we use an m-estimator to find some value to estimate a non zero value when you encounter it in N.B.

m → estimator $P(a_i \mid v_j) = \dfrac{n_c + mp}{n+m} = 0$?

March 14th, 2017



Directed Acyclic Graph (DAG)

Belief network
how to infer?

Bayesian Belief Network

- Naive Bayes is too restrictive
- The problem may prone to be intractable without such assumptions.
- Bayesian Belief Networks describe conditional independance among subsets of variables.
- allows combining the prior knowledge about (in)dependencies

among variables observed in training data.

"Bayesian Belief Networks" = "Bayes Nets"

Conditional Independance.

X is conditionally independent of Y given Z if the probability distribution governing X is independent of the value of Y given the value of Z ; ie, if

$\forall\ x_i, y_j, z_k$

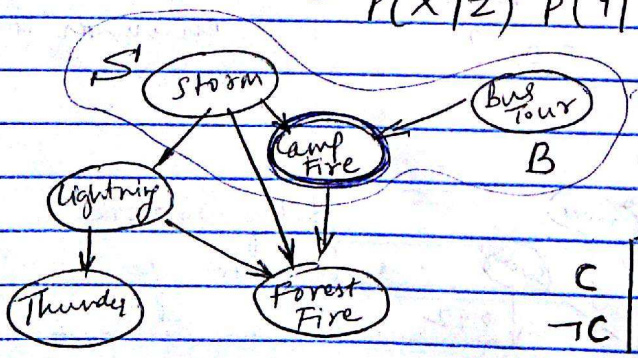
$$P(X = x_i \mid Y = y_j \mid Z = z_k) = P(X = x_i \mid Z = z_k)$$

(or) $P(X \mid Y, Z) = P(X \mid Z)$

"X is not dependant on Y"

Eg $P(\text{Thunder} \mid \text{Rain, Lightning}) = P(\text{Thunder} \mid \text{Lightning})$

$$P(X, Y \mid Z) = P(X \mid Y, Z)\ P(Y \mid Z)$$
$$= P(X \mid Z)\ P(Y \mid Z)$$



Camp Fire = C

No camp Fire = ¬C

	S, B	S, ¬B	¬S, B	¬S, ¬B
C	P(C | S, B) = 0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors.

In General,

$$P(Y_1, Y_2, \ldots Y_n) = \prod_{i=1}^{n} P(Y_i \mid \text{Parents}(Y_i))$$

where Parents (Y_i) denote immediate predecessors of Y_i in the graph. The

(Bayesian net)

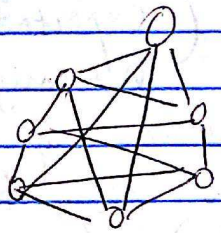
The joint distribution is fully defined by the graph plus the probability of: $(y_i \mid \text{parents}(y_i))$

Inference in Belief Networks

How can we infer the values (probabilities) of one or more network variables given observed values of others?

The Bayes Net contains all information needed for inference. If only one variable with unknown value, that would be easy to infer.
Generally, this is NP-hard. (= intractable)

NP-hard: E.g. Travelling Salesman Problem.



Minimizing the gas milage, how can I travel to all cities?

Learning in Belief Nets

- Structure may be known/unknown.
- Training examples might provide values of all variables or just some.

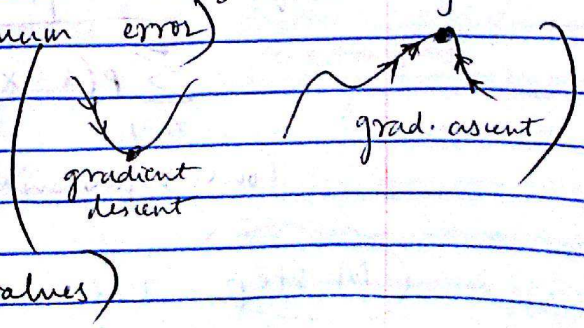
If structure is known and we observe all variables, then it's easy. Just use Naive Bayes classifier!!!

Learning: Gradient Ascent for Bayes Nets

(if we are looking to maximize something, not to get the minimum error)

Let w_{ijk} denote one entry in the cond. probability table for y_i in the network.

$w_{ijk} = P(y_i = y_{ij} \mid \text{parents}(y_i) = \text{the list } u_{ik} \text{ of values})$



Eg. if y_i = CampFire, then u_{ik} might be $\langle$ Storm = True, BusTour = False $\rangle$

Perform gradient ascent repeatedly.

① update all weights w_{ijk} using training data D.

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \left[\frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}} \right]$$

② Renormalize the w_{ijk} to assure

$$\sum_j w_{ijk} = 1 \qquad 0 \leq w_{ijk} \leq 1$$

One may also use EM algorithm for learning (Expectation Maximization)

Repeatedly do:

① Calculate probabilities of unobserved variables assuming h.

② Calculate new w_{ijk} to maximize $E[\ln P(D|h)]$ where D now includes both observed & unobserved (calculated) variables.

(encoding my data given hypoth. | (hypothesis, like "yes there is cancer" or "no cancer")

EM algorithm:

Pick random initial $h = \langle \mu_1, \mu_2 \rangle$ (mixture of Gaussians)

E-step : $E[z_{ij}]$ (estimate the unobserved value based on n distributions)

(unobserved variable)

$$E[z_{ij}] = \frac{P(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} P(x = x_i | \mu = \mu_n)}$$

(only 2 gaussians to build the mixture)

M-step Calculate a new max likelihood hypothesis $h' = \langle \mu_1', \mu_2' \rangle$, assuming the value taken on

by each hidden variable z_{ij} is its expected value $E[z_{ij}]$.

Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu_1', \mu_2' \rangle$

Then we apply the adjustment:

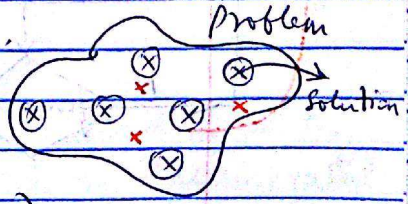
$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

March 16th, 2017 Evolutionary Algorithm

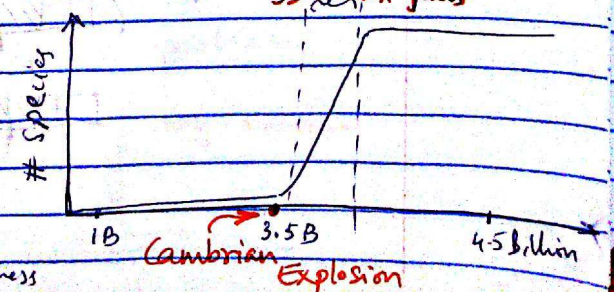
Not considered as "learning". Mainly "optimization"
Inspired by natural evolution
↳ survival of the fittest (Natural selection)
- EA's are function optimizers
- Introduced by John Holland. (1975) "Adaptation in Natural and Artificial Systems"
- GA's are EA's which are basically stochastic search methods. (Genetic algorithms)

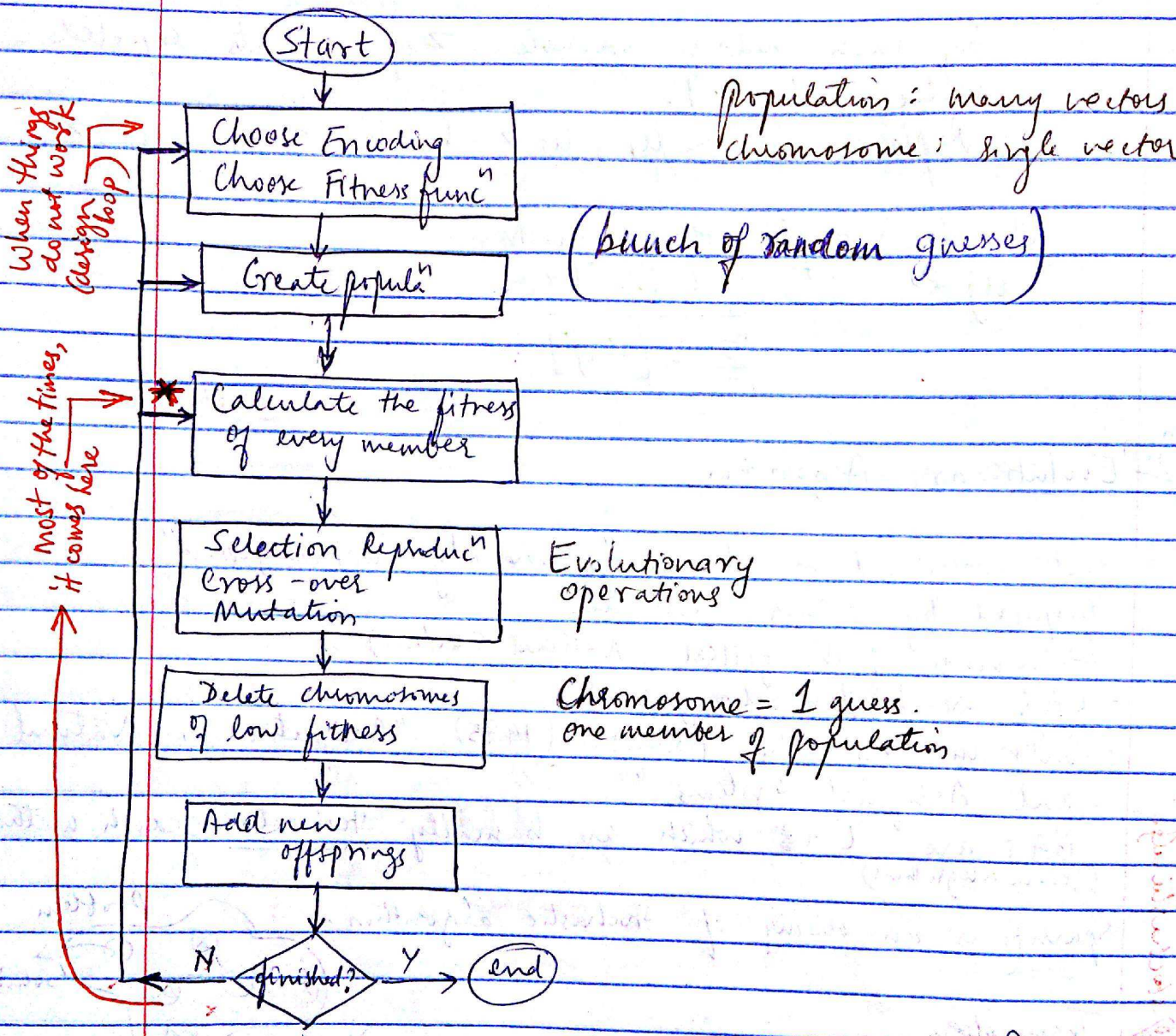
Sparsity is an enemy of stochastic algorithm.

"Sparsity = enemy of stochastic algo."



Terminology
① Population (candidate solutions = guesses)
② Generation (i.e., "iteration", "epochs", "episodes")
(in clustering) (Neural networks) → (in Reinforcement Learning)
③ ~~Re~~ Fitness → driving force every member of population that is fit is allowed to survive and reproduce itself.
↳ = (objective function)
④ Crossover (getting off/springs)
⑤ Reproduction
⑥ Mutation
⑦ Encoding (Eg: DNA → Binary code)
⑧ Chromosome
↳ 1 individual member in population. 1 guess



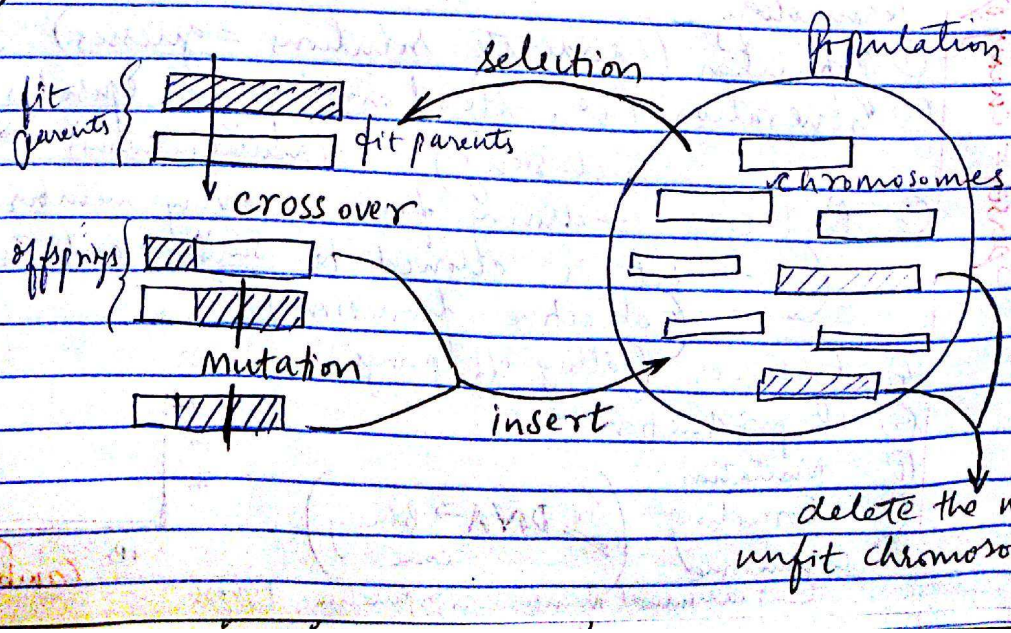


Population : many vectors
chromosome : single vector

(bunch of random guesses)

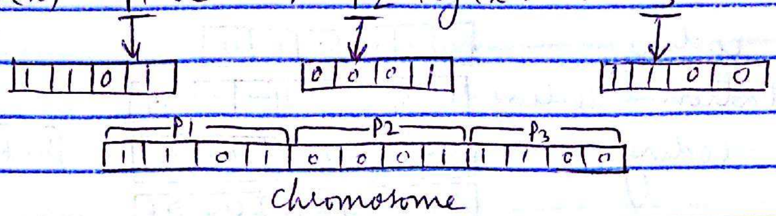
Evolutionary operations

Chromosome = 1 guess.
one member of population



Function to be optimized

$f(x) = P_1 x^2 + P_2 \log(x) + P_3 \sin(x)$.

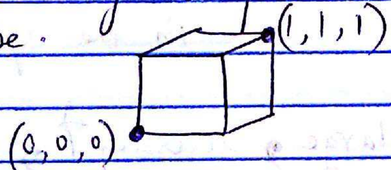


Coding & the GA's secret. (Stochastic optimization)

we look at bit strings of size L = [0 1 1 . . . . . 1 1] (L)

size of search space = 2^L

GAs basically sample the corners of this L dimensional hypercube.



(for neural network, you go inside, the continuum of all answers) ∴ more accurate answers using NN.

this analogy is only for binary encoding.

But the problems you can solve using NN is something that cannot be solved using GA & vice versa.

Most (test functions =) problems can be encoded with max 30 bits. Any-thing less can be **enumerated**

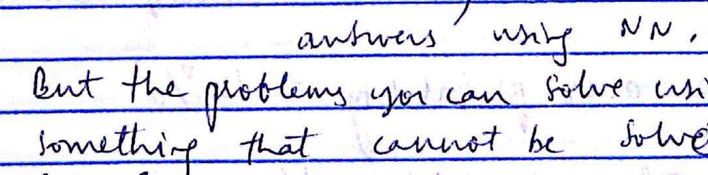
Eg. chess game → branching factor = 16

→ a game is made of 100 moves (on average)

∴ 16^{100} possibilities. $= (2^4)^{100} = 2^{400}$

If # of solutions is small (sparse problem) then stochastic search will not work.

Eg.



Encoding.

① Binary coding

② permutation coding

③ value coding.

1	1	0	1	1	0

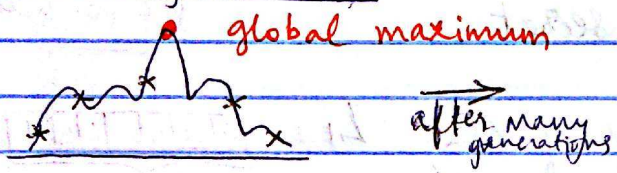
1	3	7	5	9	2	8	6	4

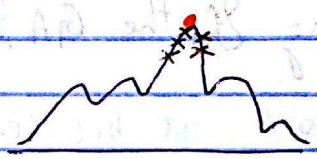
1.234	9.657	0.1501

back	back	left

A	C	W	Y	B	X	N	I	O	J	D	I

Diversity & GAs





Converge towards actual maxima.
(but we may never get to exact maxima)

|population| = 5 (5 chromosomes. ∴ 5 guesses randomly in the space)

→ Size of population ?
- large, accuracy ↑, becomes sluggish
- small, less accurate, but fast.

→ Cross over probability = 100%

→ Mutation probability = 1/1000 : one in 1000 offsprings, you perform mutation. i.e., just flip a bit.