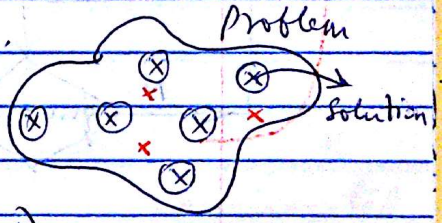
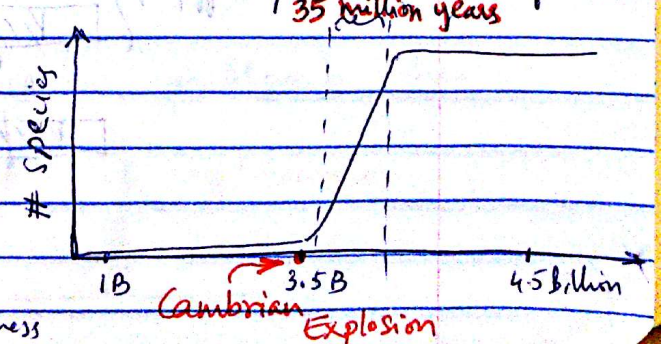
March 16th, 2017 Evolutionary Algorithm

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

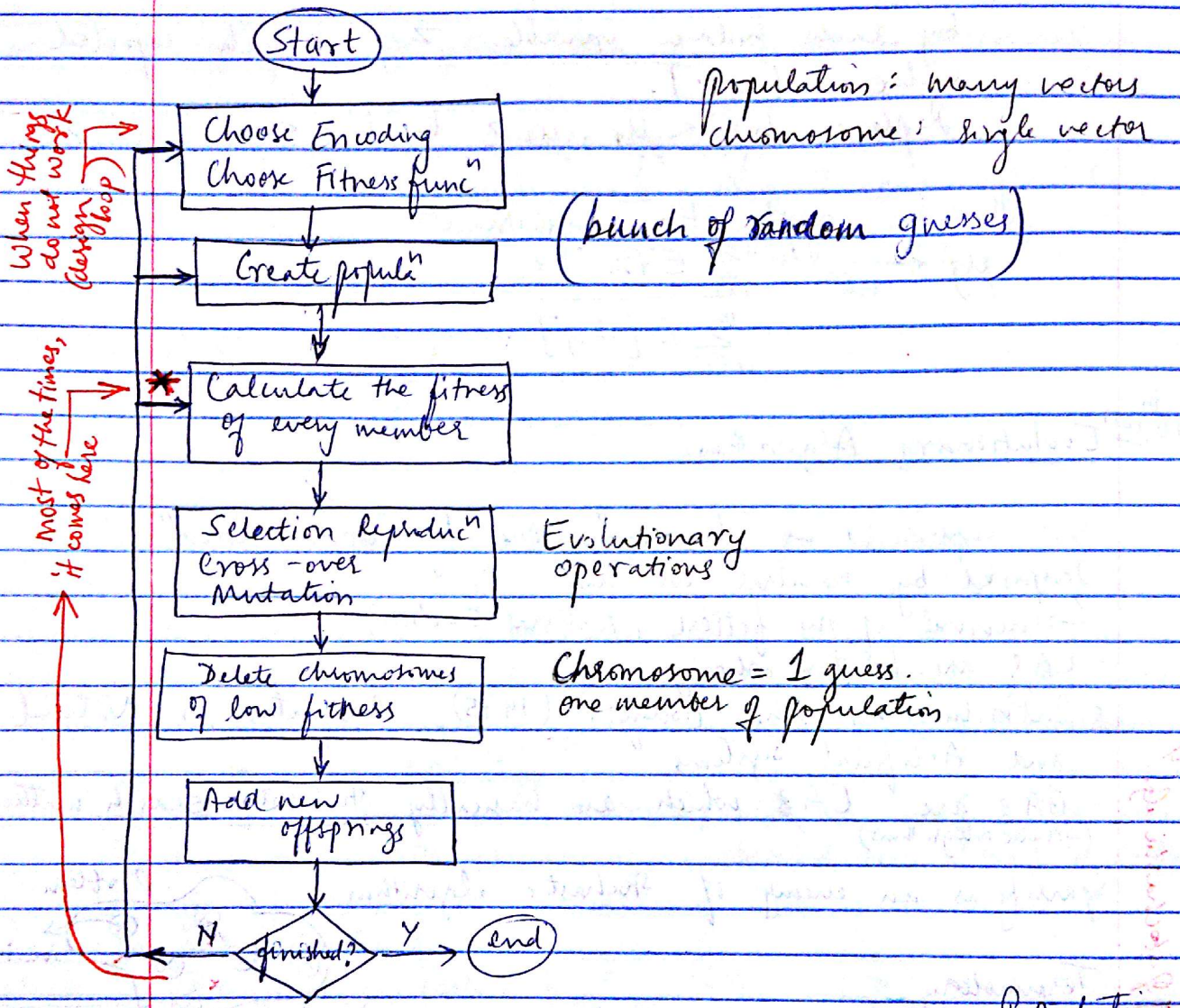
Sparsity is an enemy of stochastic algorithms.



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



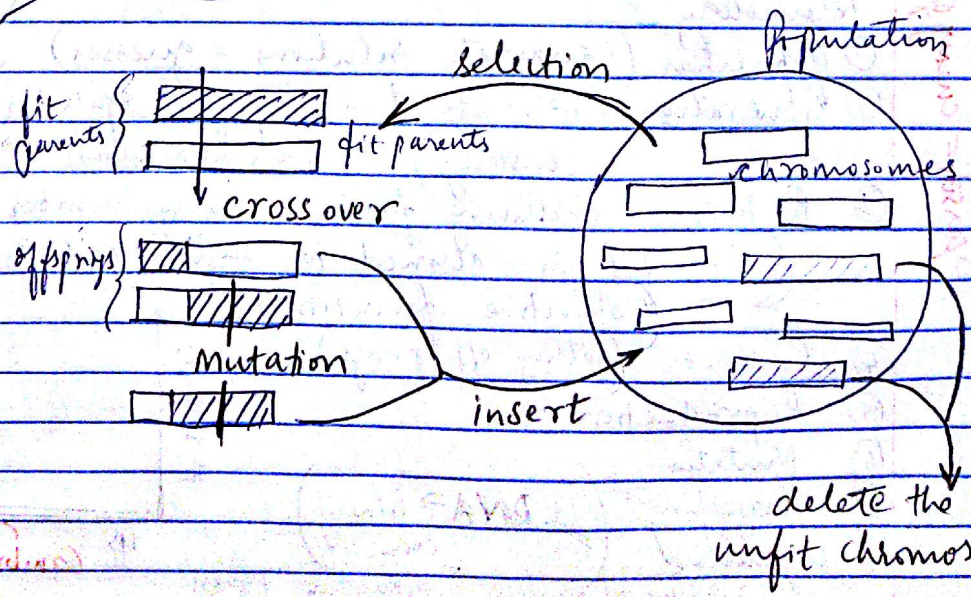
Sparsity = enemy of stochastic algo.



Population : many vectors
chromosome : single vector

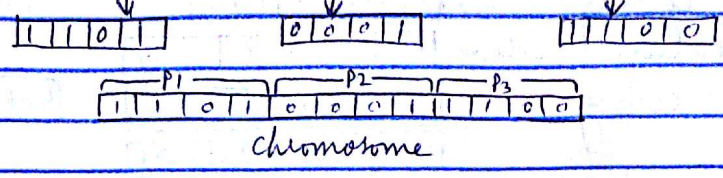
(bunch of random guesses)

Chromosome = 1 guess.
one member of population



Function to be optimized

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



chromosome

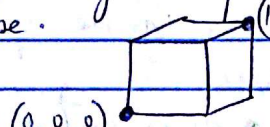
Coding & the GA's secret.

(Stochastic Optimization)

we look at bit strings of size L = |0|1|1|1|1|1|1|1| (L bits)

Size of search space = 2^L

GAs basically sample the corners of this L dimensional hypercube.



this analogy is only for binary encoding.

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

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. Anything less can be **enumerated**

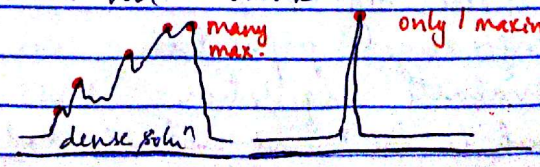
Eg: chess game → branching factor = 16

→ 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:



fitness : driving force for evolutionary algo.
error : driving force for neural network.
reward/punishment : for learning algo.

Encoding

① Binary coding
② permutation coding
③ value coding.

1	1	0	1	1	0

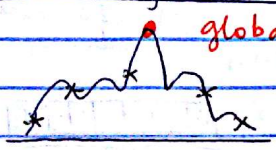
1	3	7	5	9	2	8	6	4

1.234	9.657	0.1501

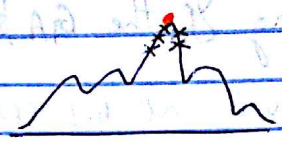
back	back	left

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

Diversity & GAs



after many generations



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

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

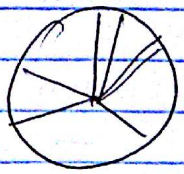
→ Size of population ?
- large, accuracy ↑, becomes sluggish
- small, less accurate, but fast.

→ Cross over probability = 100%

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

March 21st, 2017

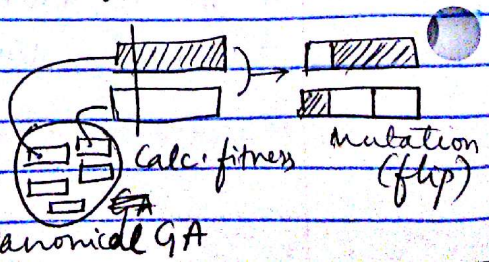
Selection of Chromosomes



① Roulette wheel selection
② Rank selection
Sort chromosomes based on fitness, then rank them

pseudo code

```
Genetic Alg () {
initialize population
evaluate population
while the end not reached {
```

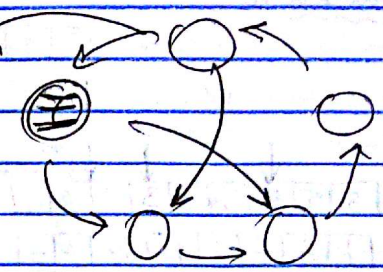


Canonical GA

- evolu" tech. are very slow ∴ done offline, get solution or application is not time critical
- diversity of solution ↓ses over time

- low diversity ⇒ you are going to solution. ∴ Stop.
- evolutionary alg. does not require data. No thing to save / store. You only obtain a set of solutions.

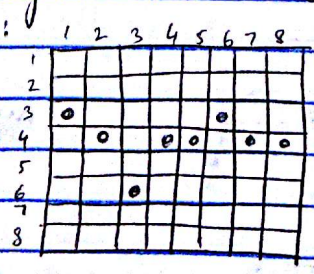
select solutions
perform cross over / mutation
evaluate population
add offspring
delete unfit ones



Many GA Models
↳ Island mode

Generally:
1. Initialization – usually done randomly
   – can integrate existing solutions
2. Termination – check it in every generation
   – by reaching a fitness
   – by arriving at max. # of generations
   – by achieving a minimum level of diversity.

Problem: 8 – Queens problem
The problem is to place 8 queens on chessboard such that they do not check each other.
Question: What is the fitness function?
fitness: $\frac{1}{\#\text{check}}$
fitness function: (This has to be maximized)
(# of checks have to be min.)
Manually: 92 solutions.
Phenotype:



Chromosome: | 3 | 4 | 6 | 4 | 4 | 3 | 4 | 4 |

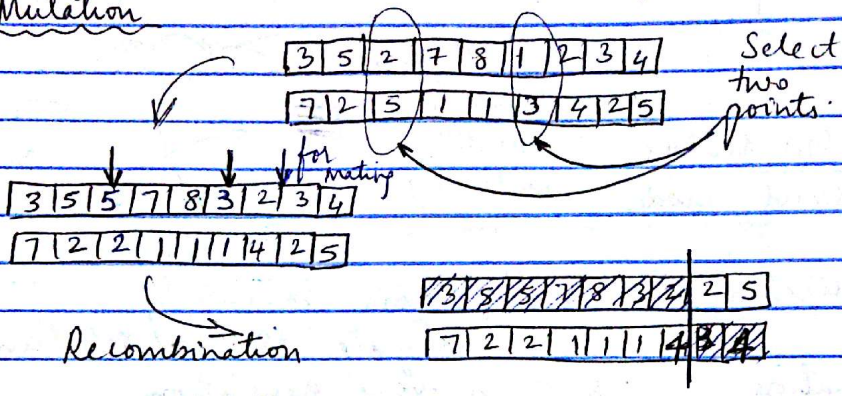
chromosome is based on permutations

Fitness: ① Penalty of one queen is the # of queens she can check.
② Penalty for configuration is the sum of all penalties.
③ Penalty has to be minimised

④ Fitness of the configuration : max $\frac{1}{\Sigma \text{penalties}}$.

(has to be 8 i.e. not 9 elements in all chromosomes (eg 8-queen) not 9.)

Mutation

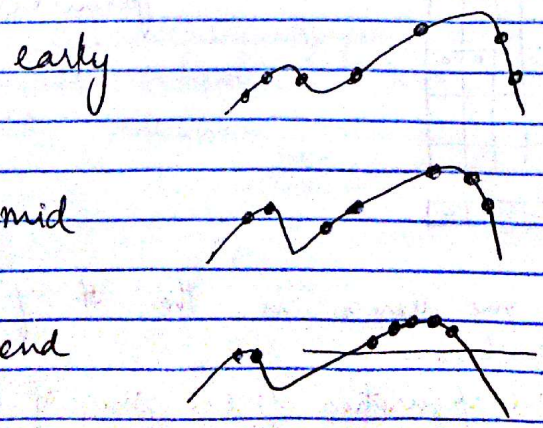


Selection

Pick 5 parents and take the best 2 to undergo cross-over

Recombination probability = 100%
Mutation → swap (Generates more diversity
Mutation probability = 80%
Parent selection → best 2 out of 5.
Survival selection → Replace the worst.
Termination → after 100,000 iterations

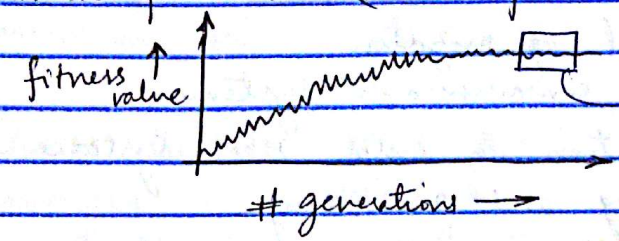
Typical EA's behaviour



high diversity (required to be high. otherwise it is a killer, if you have low diversity)

low diversity (means converged to a solution)

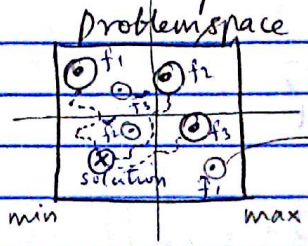
Fitness function (the way it changes over time)



Is it worth it to spend time to improve initialization?

↳ It depends

Given a problem, we make random guesses r_1 and r_2 to solve the problem. Compared to when we get r_1 and its opposite $\breve{r}_1$

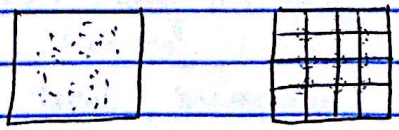


$\breve{X} = min + max - X$

Random trajectory towards solution

$$P\left(g(r_1, r_2) < g(r_1, \breve{r}_1)\right) \approx 12.5\%$$

r_1, r_2: 0.1, 0.3 ; $r_1, \breve{r}_1$: 0.1, 0.9



March 23rd 2017

Swarm Intelligence

Inspired by social behaviours of animals

Ant colonies, Bird flocking, fish schooling

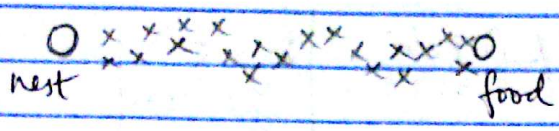
Ant colonies (AC)

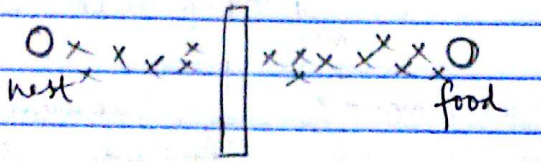
- Current population = 10^{16} ants on this planet.
- been there since 100's of millions of years.
- one of the most successful species on the planet.

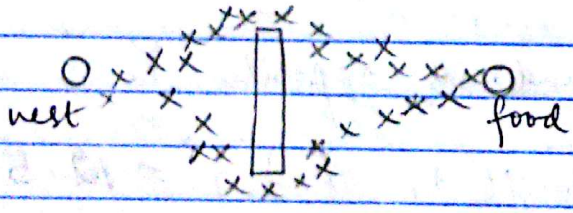
Concept of stigmergy: indirect communication used by social insects to co-ordinate activities.

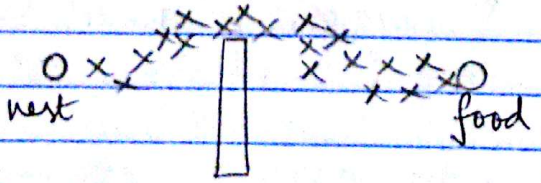
Meta heuristic

Applications of ACs
1. Combinational optimization
2. Routing for communication systems.
3. Task allocation in multi-robot systems.
4. Graph drawing and partitioning
5. Exploratory data analysis.

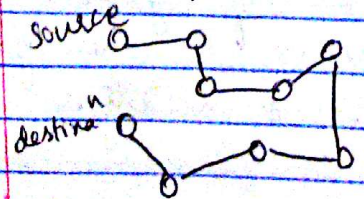






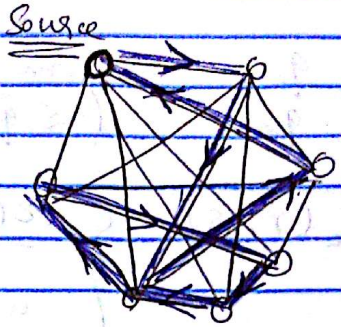


- Ants use pheromone trail
- Pheromone is a diffuse chemical substance
- Auto catalytic process.
- shortest path = higher pheromone concentration.
- Trail laying / trail following behaviours.
- Artificial ants = piece of software.
  (an agent moving from point to point on a TSP (Travelling Salesman Problem) graph.



TSP.

Find a path with minimum cost. Simple one:



Choose the cities that are connected by edges with a lot of pheromone trail.

Initially 'm' ants are placed on randomly selected cities.

At each step, the ants move to new cities and modify the pheromone trail. → "local trail updates".

When all ants have completed a tour (meaning cycle of actions → (source → destination → source)) the ant that made the shortest tour modifies the edges of its tour. "global trail update"

Ideas from nature

① Preference for paths with high pheromone concentration
② Higher rate of growth of the amount of pheromone on shorter paths
③ The trail mediated communication among ants.

The ant k in city r chooses the city s not in the working memory M_k by applying a probabilistic decision.

$$S = \begin{cases} \text{argmax}\{\tau(r,u) * [\eta(r,u)^\beta]\} & \text{if } q \le q_0 \\ S \text{ else} \end{cases}$$

$\tau(r,u)$ = pheromone on the edge (r,u).
$\eta(r,u)$ = heuristic function (= inverse of distance)
β = weighs the relative importance of pheromone trail & closeness.

q = random number in $[0,1]$ (changes for a tour)

q_0 = $(0 \le q_0 \le 1)$ (doesn't change for a tour)

$$p_k(r,s) = \begin{cases} \dfrac{\tau(r,s)\,\eta(r,s)^\beta}{\sum_{u \in M_k} \tau(r,u)\,\eta(r,u)^\beta} & \text{if } s \in M_k \\ 0 & \text{else} \end{cases}$$

Global trail update

$$\tau(r,s) \leftarrow (1-\alpha)\,\tau(r,s) + \alpha\,\Delta\tau(r,s)$$

$$\Delta\tau(r,s) = \frac{1}{\text{length of shortest tour.}}$$

Similar to Reinforcement Learning ⇒ better solutions get a higher reward (= pheromones in our case)

Local Trail update

$$\tau(r,s) \leftarrow (1-\alpha)\,\tau(r,s) + \alpha\,\tau_0$$

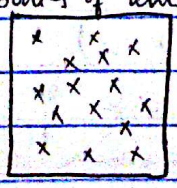
Motivated by trail evaporation.

Goal: avoid a very strong edge being chosen by all ants.

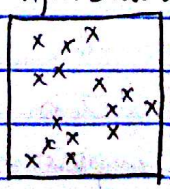
(especially in the beginning, we can get into local maxima & get stuck there).

Cemetry Organization

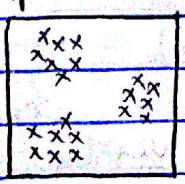
dead bodies of ants:



After 3 hours:



After 5 hours:



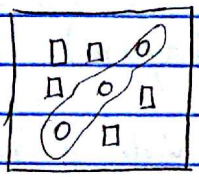
Exploratory Data Analysis

March 28th, 2017 Boosting

Is it worth it to design a complex solution for a problem at hand?

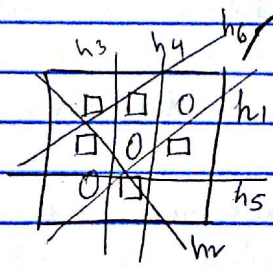
* Yes: we need complex structures to solve difficult probs.
* No: we can reach the same level of generalization with simple solutions.

Eg



non linear, perhaps SVM / MLP / deep net. (if higher dimension)

Yes can be solved but requires lot of effort.



No

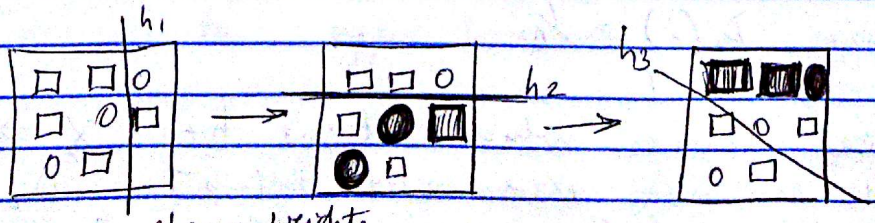
Instead of a powerful classifier, use several weak classifiers.

What is a weak classifier?

- It can classify data
- It has to be better than random (>50% accurate)
- ideally, the # of hypothesis should be $\rightarrow \infty$

$|H| \rightarrow \infty$

Boosting parts come from manipulation of weights.



change weights of those that have been misclassified.

$Y = \text{sign}\left(\sum \alpha_i h_i\right)$

For $h_i \in H$, we can approach high accuracy if
① weights of misclassified instances are manipulated.
② each classifier is better than random (accuracy >50%)
☆ → ③ no outlier / noise
(Boosting will collapse if we have outliers)
Most common out of the box classifier is boosting multiple small decision trees.

1990 – Boost by majority algorithm
1995 – AdaBoost (Freund and Shapire)
1997 – Generalized AdaBoost
2001 – AdaBoost for face detection (Viola & Jones approach)
2006 – RAMO Boost (Boosting for imbalanced data)

- AdaBoost is a linear classifier.
- AdaBoost can be used as feature selector.
- Sequential decision making.
- AdaBoost delivers a "strong" classifier as

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

(α_t: weight we try to induce; $h_t(x)$: weak classifier; $y \in \{-1, +1\}$)

$$H(x) = \text{sign}(f(x))$$

Say,
Given $(x_1, y_1), \ldots (x_m, y_m)$; $x_i \in X$, $y_i \in \{-1, +1\}$

Initialize $D_1(i) = 1/m$
For $t = 1, 2, \ldots T$
① Weak learner which returns $h_t = X \to \{-1, +1\}$ with minimum error wit distribution D_t
② Choose $\alpha_t \in R$
③ Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

($D_t(i)$: random; weights; y_i: desired output; $h_t(x_i)$: current output; -1 or $+1$ for each instance; Z_t: decision on hyperplane)

Z_t is a normalizing factor so that D_t is a distribution to normalize

(from training data)

Strong classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Complexity $\propto f(t)$

All information is stored in D_t. → { hypothesis $h_t(\,)$, input x_i, decision y_i, weights α_t }

When we call "weak learner" given the distribution D_t, it will return a weak classifier $h_t : X \to \{-1, +1\}$ from $H = \{h(x)\}$ (all decision trees we can generate)

Select a weak classifier with the smallest weighted error.

$$h_t = \underset{h_j \in H}{\text{argmin}} \; \underbrace{\varepsilon_j}_{\text{error}} = D_t(i)\left[y_i \neq h_j(x_i)\right].$$

find Distribution value when

Condition: $\varepsilon_t < 0.5$

Most common decision tree builder, perceptron learning rule.

Main goal: minimize $\varepsilon_t = \frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right|$

(count when x_i (how many times) has been misclassified)

This can be upper bounded by $\varepsilon_t(H) \leq \prod_{t=1}^{T} Z_t$

How to select α_t:

Select such that we minimize $Z_t(\alpha)$ in each step

If $Z_t(\alpha)$ is a convex differentiable function with one extremum, then $h_t(x) \in \{-1, +1\}$ then optimum

then optimal $\alpha_t = \frac{1}{2} \log\left(\frac{1+r_t}{1-r_t}\right)$

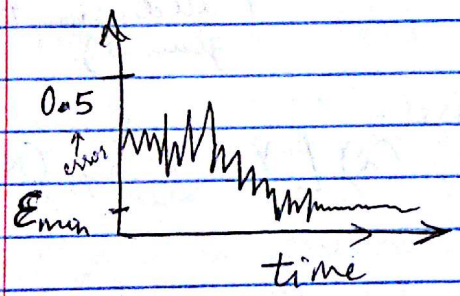
where $r_t = \sum_{i=1}^{m} D_t(i)\, h_t(x_i)\, y_i$

If we have that, we can define:

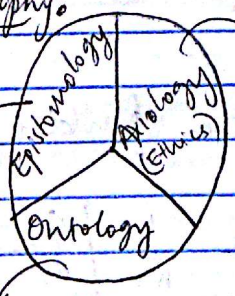
$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \leq 1$ for optimal α_t

$\Rightarrow D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

$D_{t+1}(i)$: new value



March 30th 2017 Philosophy:



What is right / wrong?

"What is knowledge"
"perception / conception"

"Does god exist"? Religion... some driving force

1. Sociates: Inquiry, (find it yourself)
2. Plato: Ideas are the source of everything
   : There is such a thing as "perfect state" when enlightened people rule.
3. Thomas Aquinas: Faith & reason will converge (must)

④ Thomas Hobbes : Social Atomism : Selfish interest drives everything.
To end the war of all against all, he proposes a "Social Contract".

⑤ John Locke (one of the initiators of Enlightenment).
- He is strongly a protective of private property.
- There are god given rights.

⑥ Immanuel Kant
- Happiness cannot be the motivation of our deeds.
- Ethics is simply there → There is god since we know ethics. We know naturally what is right/wrong

⑦ Jeremey Bentham : ◦ All actions are motivated by pleasure.
- Utilization : whatever is useful is right.
Thinking is not the measure for protection
- If you suffers, you should have rights

⑧
- J. S. Mill "on Liberty".
- Harm principle
You are free to do whatever you like unless you are harming others or yourself.

Thinking → Right Choice?
↓
Knowledge
↕
Memory → Conception

Is there such a thing as the right choice?

But there is another venue.

... other venue is perception