COGS 186 Genetic Algorithm

Intro ML lecture.

Discriminant model vs. Generative model.

↓ SVM, ...

↓ GAN, Transformer ...

Genetic Algorithm → An Optimization Algorithm

Naive Bayes.

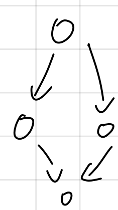
↓

Bayesian Network

Assume some conditional dependence among some parameter

↓

as a network



Bayesian vs. Frequentist, difference view of probability.

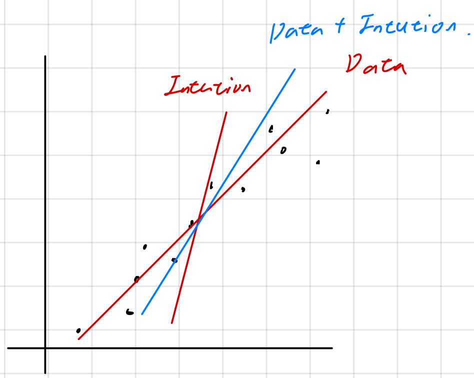
↓

some ground truth.

↓

prior belief.

Data + ground truth.



Linear Regression

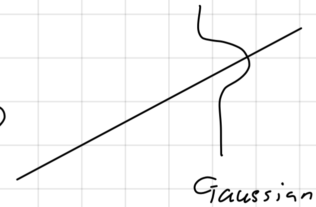
- generative model

independent (explanatory variable).

error term / variations

$$y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

$$\hat{\varepsilon}_i \sim N(0, 6^2)$$



$$y_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + \hat{\varepsilon}_i$$

true value

estimated parameter

uncertainty

model : equation or function.

linear regression

Perceptron

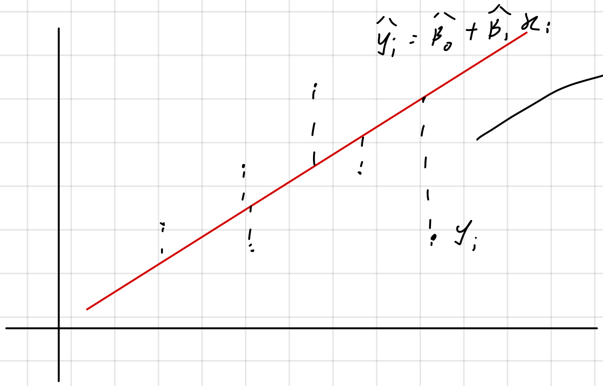
objective function.

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$$

estimated value

no random variation

Quantifying error / cost — the "goodness" of the line.



Distance.

$$\sum_{i=1}^{n} |\hat{y} - y_i| \rightarrow \text{Absolute loss}$$

$$= \sum_{i=1}^{n} |(m x_i + b) - y_i|$$

which parameters result in lowest "cost"/error.

- $\boxed{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}$ → sum of squared error.

least square objective function.

SSE :

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 = \text{risk}$$

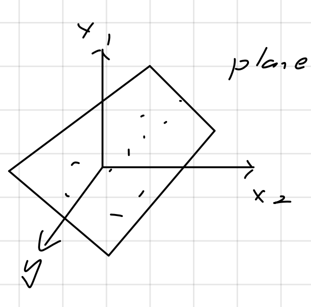
data points.

a function of the parameter

$$x_2 = \beta_1 x_1 + \beta_0$$

$$\boxed{\beta_2 x_2 + \beta_1 x_1 + \beta_0 = 0}$$

multiple features.



plane

extend

$$\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n = 0$$

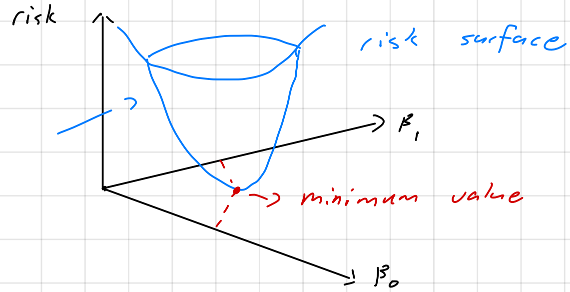
A hyper-plane.

risk surface

(minimize)

$\Downarrow$

bowl shape
b/c quadratic
function.



To find minimum, one way is to find derivative and set to 0.

$$SSE = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_1)^2$$

$$\frac{\partial SSE}{\partial \beta_0} = \frac{\partial}{\partial \beta_0} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_1)^2$$

$$= \sum_{i=1}^{n} 2(y_i - \beta_0 - \beta_1 x_1)(-1)$$

$\Rightarrow$

$$2\sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i) = 0$$

$$n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i = 0$$

$$n\beta_0 + \beta_1 n \bar{x} - n\bar{y} = 0$$

$$\beta_0 + \beta_1 \bar{x} - \bar{y} = 0$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\frac{\partial SSE}{\partial \beta_1} = \frac{\partial}{\partial \beta_1} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_1)^2$$

$$= \sum_{i=1}^{n} 2(y_1 - \beta_0 - \beta_1 x_1)(-x_i)$$

$\Rightarrow$

$$2\sum (y_i - \beta_0 - \beta_1 x_i)(-x_i) = 0$$

$$\sum (\beta_0 + \beta_1 x_i - y_i)(x_i) = 0$$

$$\sum \beta_0 x_i + \sum \beta_1 x_i^2 - \sum x_i y_i = 0$$

$$n\beta_0 \bar{x} + n\beta_1 \bar{x}^2 - n\bar{x}\bar{y} = 0$$

$\downarrow$

$(\bar{y} - \beta_1 \bar{x})\bar{x}$

$\vdots$

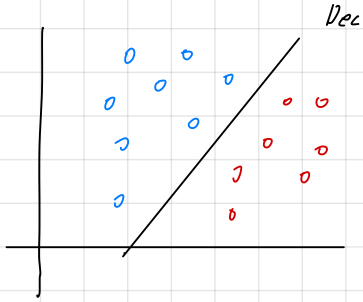
closed-form solution!

$$\Rightarrow \boxed{\begin{aligned} \beta_1 &= \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{Cov(x,y)}{Var(x)} \\ \beta_0 &= \bar{y} - \beta_1 \bar{x} \end{aligned}}$$

Perceptron Algorithm

Classification problem.

Decision Boundary
(perceptron)



What line is best?

$\Downarrow$

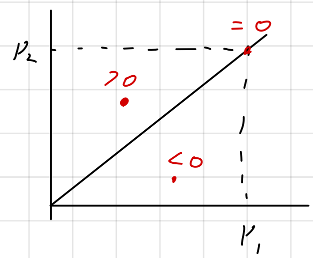
Objective function.

$\Downarrow$

optimization problem

$\beta_0 + \beta_1 X_1 + \beta X_2 = 0$

linear regression



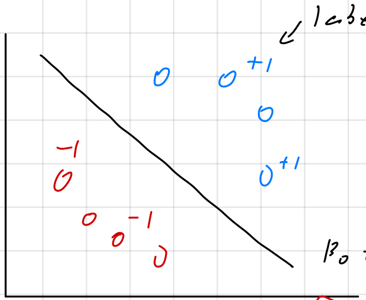
if (p_1, p_2) on the line,
$H(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0.$

Which side of the plane

$\Downarrow$

classification!

label.



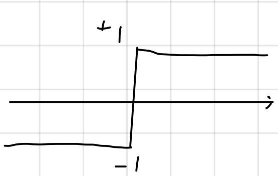
$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$

$$\boxed{\text{sign}\left(\beta_0 + \sum_{i=1}^{d} \beta_i x_i\right)}$$

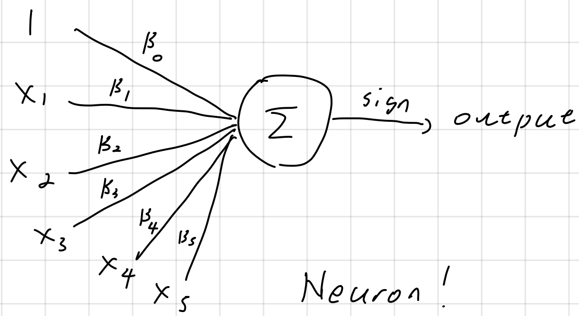
$\Downarrow$

label +1 or -1.

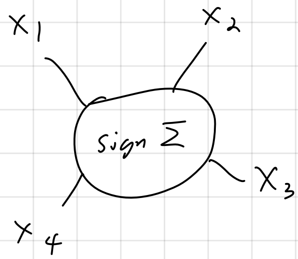
sign function:



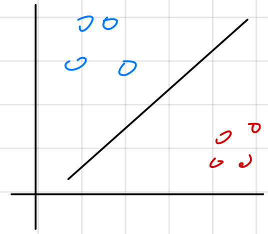
Activation function.

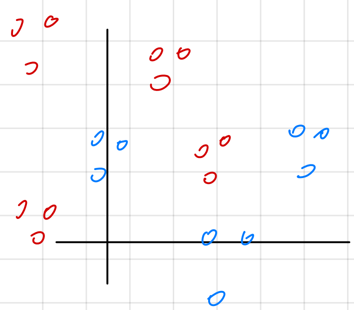


Neuron!



$\Rightarrow$



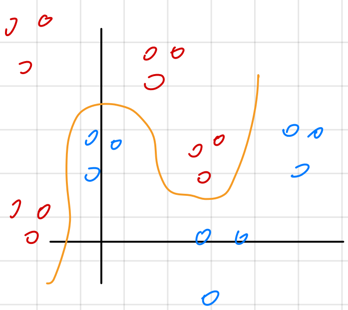


problem
not linearly seperable.

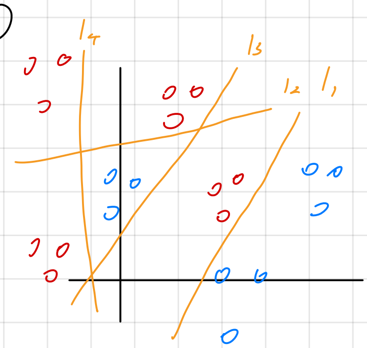
Possible methods:

① bring up the dimension $\longrightarrow$ feature mapping

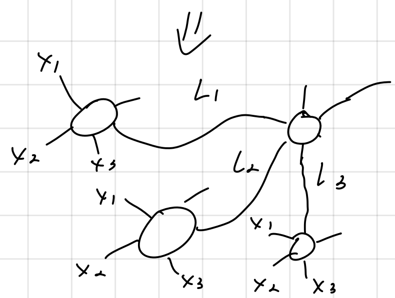
② non-linear classifier.



③



$\Downarrow$



multiple neurons.

Perceptron

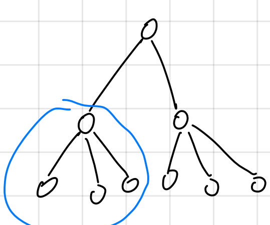
$\downarrow$

hyperplane

Neual Networks! or ANN

$\downarrow$

A collection of planes.
(perceptrons)

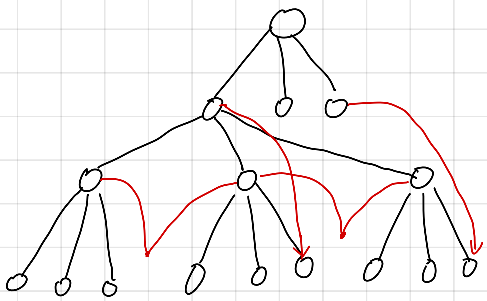


$\longrightarrow$

ANN

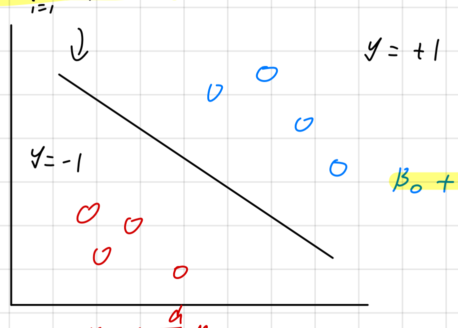
Single layered NN
perceptron
seperating hyperplane.

in deep learning, a convoluted neural networks



So, linear seqerating hyper-planes:

$\beta_0 + \sum_{i=1}^{d} \beta_i x_i = 0$



$\beta_0 + \sum_{i=1}^{d} \beta_i x_i > 0$

$\beta_0 + \sum_{i=1}^{d} \beta_i x_i < 0$

The Model:

$$\hat{y} = M(x) = \boxed{\text{sign}\left[\hat{\beta}_0 + \underbrace{(\hat{\beta}_1, \cdots \hat{\beta}_d) X^T}_{\vec{\beta} X^T}\right]}$$

where

$$\text{sign}(A) = \begin{cases} 1 & \text{if } A > 0 \\ -1 & \text{otherwise} \end{cases}$$

Decision boundary

$$\hat{\beta}_0 + (\hat{\beta}_1, \cdots \hat{\beta}_d) X^T = 0$$

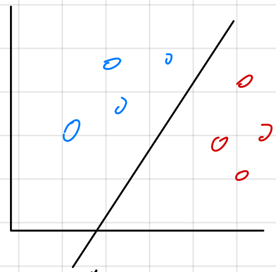
Rosenblatt's Perceptron Learning Algorithm

- Starts with a random hyperplane $(\hat{\beta}_0, \hat{\beta}_1 \cdots \hat{\beta}_d)$
- Incrementally modify the hyperplane such that points that are misclassified move closer to the correct side.

Gradient Descent Optimization Algorithm.

However, we can use genetic algorithm for optimization.

Optimization for perceptron



$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$

how far is each point from DB

$$\text{loss} = \sum_{i=1}^{n} -\underbrace{Y_i}_{y\_true} \cdot \underbrace{(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}_{y\_predicted}$$

minimize using gradient descent.

$$\frac{\partial D}{\partial \beta_0} = -\sum_{i=1}^{n} \frac{\partial}{\partial \beta_0} \left(Y_i (\beta_0 + \beta_1 x_1 + \beta_2 x_2) \right)$$

$$= -\sum_{i=1}^{n} y_i$$

$$\frac{\partial D}{\partial \beta_1} = -\sum_{i=1}^{n} y_i x_{1i}$$

convex function

always have one global minimum.

Gradient Descent:

To maximize F:

$$x = x_0 + \alpha \frac{dF}{dx}(x_0)$$

To minimize F

$$x = x_0 - \alpha \frac{dF}{dx}(x_0)$$

guarantee a global min for convex function.

More generally, we have the following derivative:

$$\boxed{\frac{\partial D(\hat\beta_0, \hat\beta_1, \cdots, \hat\beta_d)}{\partial \hat\beta_0} = -\sum_{i \in n} y_i \qquad \frac{\partial D(\hat\beta_0, \cdots \hat\beta_d)}{\partial \hat\beta_j} = -\sum_{i \in n} y_i x_{ij}}, \quad j = 1, \cdots, d.$$

$\Rightarrow$ in vector form: $\frac{\partial}{\partial \beta} D = -\begin{bmatrix} y_i \\ y_i x_{i1} \\ \vdots \\ y_i x_{ij} \end{bmatrix}$

In gradient descent, we are updating:

$$\beta^{t+1} = \vec\beta^{t} - \alpha \frac{\partial}{\partial \vec\beta} D = \begin{bmatrix} \beta_0^t \\ \vdots \\ \beta_d^t \end{bmatrix} - \alpha \begin{bmatrix} -Y_i \\ -Y_i x_{i1} \\ \vdots \\ -Y_i x_{ij} \end{bmatrix}$$

random guess

no summation b/c we are doing each point individually

$\Downarrow$

Batch Perceptron:

While some (x_i, y_i) misclassified:

for i from 0 to n:

$$w = w + \alpha(y_i, x_i)$$

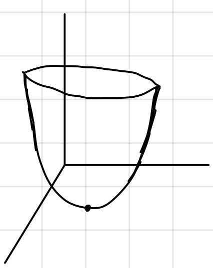
If points are linealy seperable

$\Downarrow$

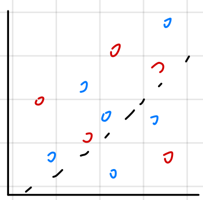
perceptron learning with gradient descent guarantee to find the line that seperates all points.

Optimization

Convex

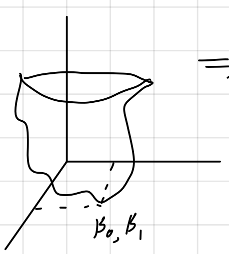


$\Rightarrow$

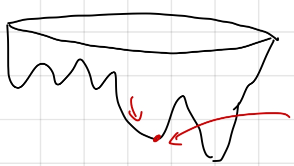


if we add constraints

Constraints



$\Rightarrow$



GD stucks in local min.

highly non-convex with local min

Non-gradient Descent ("Hill-climbing").

idea ①: jumps to neighboring places.

if it is lower than previous, keep going.

if it is higher, continue jumping.

until finding the minimum.

idea ②: Exploration phase

Exploration vs. Exploitation

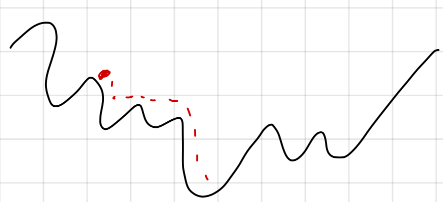
↓ explore ↓ action.

Temperature. ↓ high exploring mindset at first, meaning high probability to take bad move. And lower exploration later on.

↓

Simulated (Cooling) Annealing

- Choose a number of "steps" (iterations).
- For each step
  - Randomly introduces a perturbation (a small change to the current combination)
  - Always accept the new alternative if it reduces the cost.
  - Randomly accept some alternatives that increases the cost (uphill change).
- Slowly decrease the uphill acceptance probability
  - later step are less likely to accept bad perturbations.



x_t

$x_{t+1} = x_t + \Delta$

if $f(x_{t+1}) < f(x_t)$,

$x_t = x_{t+1}$

else

"cooling" temperature

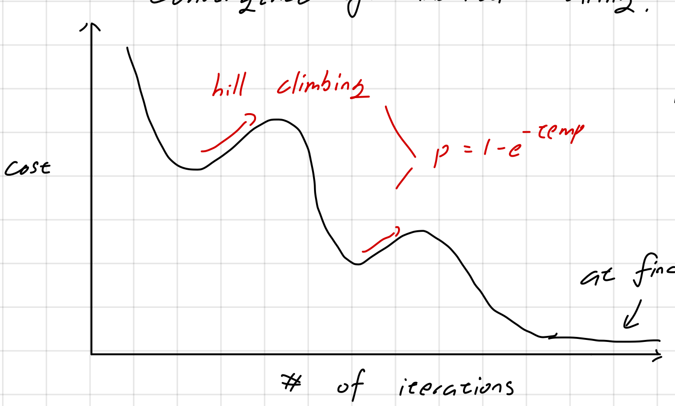
$p = 1 - e^{-temp}$

$x_t = x_{t+1}$ with p probability.

else

$x_t = x_t$

Convergence of simulated annealing.



Problem of GD: Not all curves are continuous

ex) Traveling Salesman Problem
(Discreete Curve).

ex) Four color in n by n checkboard.
No same color is touching each other.
Cost is the number of color touching each other.

(Discrete curve)

learning via optimization / no domain knowledge.

Genetic Algorithm (Biology pov)

can be used to "generate" objects by using them to figure out the rule that makes the object.

↓

Biology

↓

Evolution. → Darwin → Natural Selection.

- Exponential growth in population
- Need → Not all member survive.
  Which member survive is not random.
- Not two member are alike. Most variation are inherited.
- Limited Resources.

⇓

Not every one will survive or reproduce.

Only the best suited member will survive.

- Mutation.
- Population evolve, not individuals.
- Evolution is continuous.
- Gene centric view of the evolution

Hamilton's Rule

$R \cdot B > C$, gene relatedness R, times the benefit to the group, B larger than the cost to the individual

↓

the individual will demosstrate altruistic behavior
(individual satisfy their own benefit to save others)

human's behavior is more complex.

↓

Evolutionary Psychology.

- Natural selection
- sexual
- group / kin
- language instinct.
- ...

How things evolve?

Genotype ⟶ Phenotype

(gene) (appearance).

Genetic Algorithm (Evolutionary Algorithm)

Genetic programming ⟶ G A T C C A T A C C ...

genetic algorithm

↑

as an instruction code

old earth environment $\xrightarrow{\text{molecules}}$ amino acid

$\xrightarrow{\text{rearange}}$ RNA ⟶ (property of copying itself)

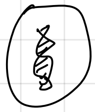
C H N O P

99.9% accuracy

RNA

Virus

Prokaryotic Bateria



One replicator molecules RNA ⟶ DNA ⟶ Virus ⟶ Prokayotic ⟶ Eukayrotic Bacteria ⟶ ... ⟶ Ant / plant

A G C T A G

arrange into something better fit

AGTC _ _ _ _ _ _ . . . optimize code such that => this is high

Genetic Algorithm:

- start with whole bunch of random solution
- keep the best one (natural selection).
- produce their mutated ones
- repeat the process over many generations

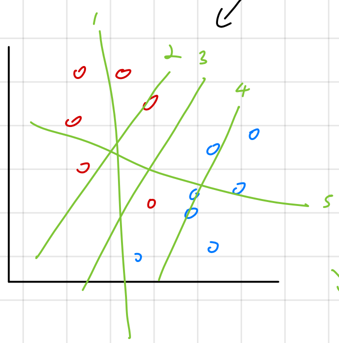
survival reproduction (AGTG ...)

prob fitness / survival

---

ex) in perceptron

bunch of random plane



$[\beta_0, \beta_1 \cdots \beta_d]$

$[0.1, -2.1 \cdots 0.9]$ -> random plane

3 is the best -> give the least mistake.

Fitness function (plane)

↓

classification accuracy on the training set (plane)

↓

mutate & crossover the best member of the population.

mutate with parameters (with constraints)
ex) mutate 1~10% of "genes" by 1~10%

---

idea.

$$f(x, y, z) = x^2 + \sin^2 x - \tan(x) + \frac{e^{-2y}}{10^{-x}} + \cdots$$

find (x, y, z) that minimizes / maximizes

initial population: random values of (x, y, z).

10,000 $\left\{ \begin{matrix} (1, 2.3, -3.5) \\ (0.5, 2.9, 4.9) \\ \vdots \end{matrix} \right)$ -> evaluate at $f(\cdot)$ the "fitness" of individual

⇓

select 1,000 "fittest" members

↓

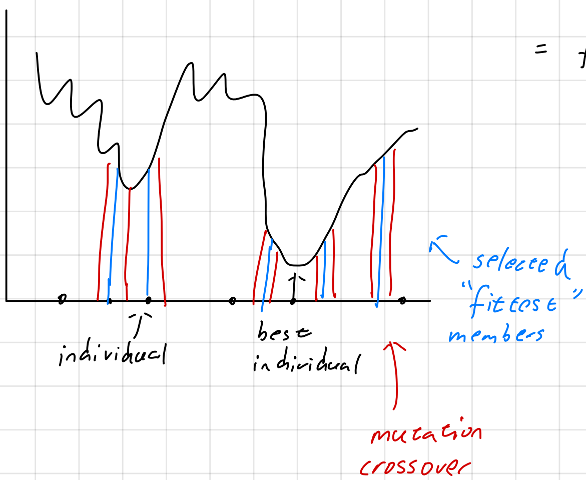
recreate next generation of 10,000

using mutation, cross over, breeding of the 1,000 "fittest" members.

repeat

generation 1 ↪ $X_1 = [\,X_{r0}\ \ X_{r1}\ \ X_{r2} \cdots X_{rp}\,]$

$= f(X_{r0})\ \ f(X_{r1})\ \ f(X_{r2})\ \ f(X_{rp})$



X = genotype

$f(X)$ = phenotype.

- Evolution is not random:
- Reproduction is key to genetic algorithm for optimization.

<u>perceptron classification problems</u>:

Start with p

Parameters :

P = population size

E = elite size

M = mutation rates, how many elements

C = crossover type

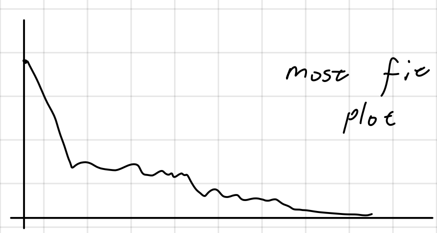
Fitness function

T = selection type

G = generations

Rosenblatt's $= \Sigma\,[\,Y \cdot (\vec{\beta}\,\vec{x})\,]$

1. Initialize 1000 random planes
2. Evaluate the value of the fitness function for each one of these planes
3. Rank these planes by the "most to least fit"
4. Perform the crossover and mutation on the top 100.



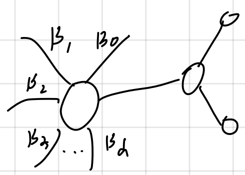
ex) Most fit

	2.7	-1.3	2.0	3.1	2.3	7.1
	1.3	-0.5	7.0	3.1	2.0	8.0
Mutate by 1 :	2.8	-1.2	2.1	3.2	2.6	7.2
Crossover :	2.7	-1.3	2.0	3.1	2.0	7.1
	1.3	-0.5	7.0	3.1	2.3	8.0

Genetic algorithm :

is good for solving problems when they are way too many potential solutions

Neural Network + GA (Neuroevolution):

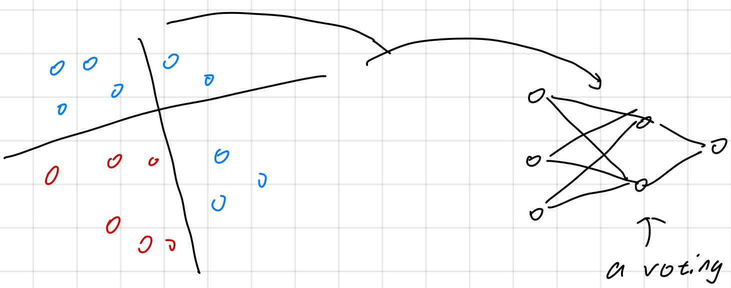


neurons learning

neurons firing.

Instinct = recognition

idea of hidden layers in perceptron classification problem:



a voting mechanism on different planes.

Traveling Salesman Problem :

shortest path to go through all the cities one time.

- start with random tours
- reproduce from the fittest members
- crossing over

· Selection Criteria : ⟶ GA will likely preserve section of chromosome.

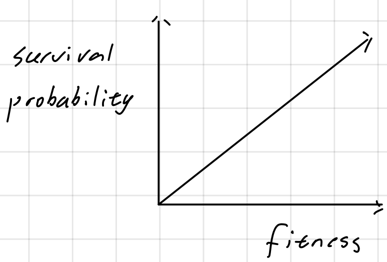
- proportionate (roulette wheel selection)
- tournament selection
- Elitism.

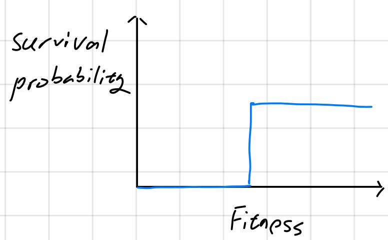
Schema Theorem : ⟶ concept

Genes ⟿ learned biological concept function
↓
GA ⟿ learning concepts.

Fitness of all individuals

ranking ⟶ in nature ⟶ modeling survival probability.
1 . . .
2 . . .
⋮





Selection Methods of GA:

① Generate initial population
- random
- heuristics (rough approx. way of getting a solution).

② Evaluate every individual fitness

③ Rank individuals based on the fitness

④ Select some individuals for reproduction & survive.

setting:

$A = 15$
$B = 10$
$C = 8$
$D = 5$
$E = 3$
$F = 2$
$G = 1$
$H = 1$

total = 45.

- Ranking based → select each individual with probability.

  $A = \frac{15}{15}$ prob.

  $B = \frac{10}{13}$ prob.

  ....

  $p_i \propto \frac{1}{\text{rank of ith}}$

- Proportion based → $A = \frac{15}{45}$ prob.

  $B = \frac{10}{45}$ prob.

  ...

  $\text{prob} = \frac{f_i}{\sum_{i=1}^{n} f_i}$

  with replacement

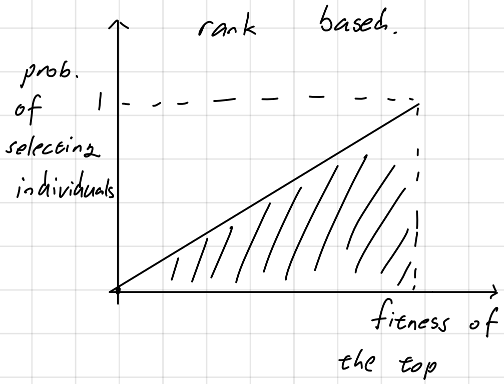
- Tournament based → select subset of individual and complete.

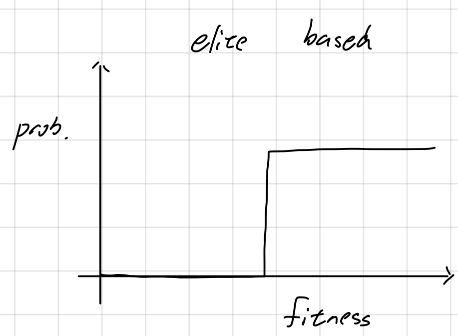
  $D = \frac{5}{5+3}$

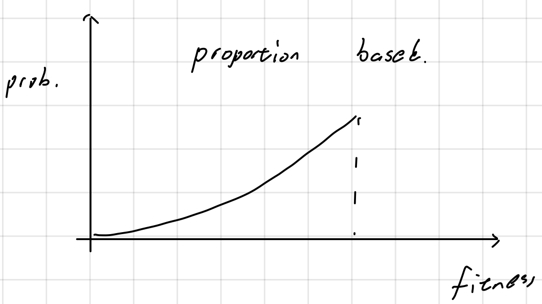
  $E = \frac{3}{5+3}$

  $\xrightarrow{\text{complete}}$

- Elite based → select the top Nth. individual







⑤ Mutate some individuals → ex)

| 0 | 1 | 1 | 0 | 1 | 1 | 0 | → | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

each bit has 1% of mutation.

⑥ Crossover some individuals.

ex) swap bits

| A | B | C | D | E | → | A | D | C | B | E |

ex) single point ordered crossover

A B C D | E F
G H I J | K L
⟶ A B C D K L
G H I J E F

disadvantage:
distant elements got broken up.

ex) double point crossover

A B | C D | E F
G H | I J | K L
⇒ A B I J E F
G H C D K L

ex) average crossover

12 13 1.1 ...
↕ ↕ ↕ average
20 10 5 ...

ex) unordered crossover

A B C D E F G
H I J K L M N

A B J K E F N
H I C D L M G

ex) Constrainted crossover

such as crossover in travelling salesmen problem.

Schema Theorem:

pattern

"learned concept"

ex)

pattern

1 * 1 * 0

3 fixed
2 varies

1 0 1 0 1
1 1 1 0 0
1 0 1 1 0

all follows the pattern/scheme.

- Order of Schema

\# of fixed bits

$O(H_1) = 3.$

- Defining length of a schema.

$\ell(H_1) = 5 - 1 = 4.$

location of last fixed bit − location of first fixed bit

$f(H)$ = average fitness of population member that contain H.

$M(H, t)$ = # of members in the population that contains H at generation t.

H is good if $f(H) > \frac{\sum f_i}{n}$ ← average fitness of the entire population.

Using proportional selection, where the probability of ith member getting selected is $\frac{f_i}{\sum_{i=1}^{n} f_i}$

# of member with H schema that will get selected is

$$M(h, t+1) = M(h, t) \cdot \frac{f(H)}{\bar{f}}$$

average of all

if $f(H) > \bar{f}$, then $\frac{f(H)}{\bar{f}} > 1$,

$$M(H, t+1) > M(H, t)$$

Game Theory

- Social interaction games
- Prisoner's Dilemma. → Both people got caught.

get interrogated seperately

Other person

		Stay quiet	Confess
You	stay quiet	-1, -1	-12, 0
	confess	0, -12	$\boxed{-8, -8}$

Nash Equilibrium. (If everyone is making a choice that is optimal given everyone else's choice stays the same).

⇓

best strategy is confessing regardless what other choose.

⇓

logical induce to the result that both will confess

ex) Coke, Pepsi

	Don't advertise	Advertise
Don't Advertise	500, 500	1000, 5
Advertise	5, 1000	✓✓

ex)

Opponent

prisoner dilemma

		A	B	C
You	left	2, 3	1, 4	~~7, 3~~
	midde	4, 1	3, 3	~~6, 2~~
	right	~~-1, 9~~	~~2, 8~~	~~8, -1~~

- opponent will never take C as B is better over all than C.
- you will never take right as middle is better
- Nash Equilibrium

ex)

	Deer	Rabbit
Deer	$\boxed{5, 5}$	0, 2
Rabbit	2, 0	$\boxed{1, 1}$

"rabbit hunting day"

Deer = 10 lbs, Rabbit = 1 lb, 2 rabbits, 1 deer

· Contract Theory → constraint

· Nash equilibrium shifted due to constraint

ex).

	Stay Home	Go Out
Stay Home	$\boxed{5, 5}$	2, 8
Go Out	8, 2	0, 0

Nash equilibrium tells you how others will play

or

how others will keep a behavior without any enforcement

ex)

	Dive Right	Dive Left
kick right	0, 0	-1, 1
kick left	1, -1	0, 0

⇒ No Nash Equilibrium.

Random, 1/2 right, 1/2 left

		② C	② D
①	A	3, -3	-2, 2
①	B	-1, 1	0, 0

⇒ mixed strategy nash equilibrium

↓

assume player B can choose either C or D with equal chance

reward for player B is same if they pick C or if they pick D.

Assume player ① pick A with prob p, pick B with prob. $1-p$.

⇒ expected reward for picking C is $P_A(-3) + (1-P_A)(1)$

$= -4P_A + 1$

reward for D is $P_A(2) + (1-P_A)(0)$

$= 2P_A$

· Pick P_A such that Reward C & Reward D are same.

So, $P_A(-3) + (1-P_A) = P_A(2) + 0$

$$1 - 4P_A = 2P_A$$

$$P_A = 1/6$$

$\Downarrow$

player 1 picks A with prob. 1/6, there is no strategy that player 2 can pick that increases the reward.

· Do the same for player 2.

reward for A = $P_c(3) + (1-P_c)(-2)$

reward for B = $P_c(-1) + (1-P_c)(0)$

So, $3P_c - 2 + 2P_c = -P_c$

$$P_c = 1/3$$

Over time, player 1 will pick A with 1/6
player 2 will pick C with 1/3

$\Downarrow$

Equilibrium.

maximizing the reward given that we don't know the opponent.

B

A

	keep quiet	confess
keep quiet	20,20	0,21
confess	21,0	5,5

equilibrium

maximize points

1. Given all the poential actions of my opponents, I pick my best move.

2. If my opponents know what I am going to do, they still can't hurt me.

Iterative Prisoner's Dilemma $\xrightarrow{\text{strategy}}$ tip for tat

$\rightarrow$ holding grude (keep quiet, than confess

Review of concepts :

N. E. if everyone is picking their best strategy

		Ⓑ	
	L	M	R
U	13, 3	1, 4	7, 3
Ⓐ M	4, 1	$\boxed{3, 3}$	6, 2
D	-1, 9	2, 8	8, -1

best strategy for A.

best strategy for A

ex) Two Nash Equilibrium.

	L	R
U	9, 9	4, 6
D	6, 5	7, 8

ex) No Nash Equilibrium.

	L	R
U	6, 8	4, 7
D	7, 6	3, 7

Battle of the Sexes

		Ⓕ	
		Boxing	Musical
Ⓜ	Boxing	2, 1	0, 0
	Musical	0, 0	1, 2

- people stick with the plan the other made.

		Ⓢ	
		L	R
Ⓖ	L	0, 0	-1, +1
	R	-1, +1	0, 0

$\Rightarrow$ mixed strategy nash equilibrium.

- Assume prob p and $1-p$.
- expected reward for Ⓢ picking L if Ⓖ pick L with p and R with $1-p$.

$= (0)(p) + (1)(1-p)$

$= 1-p$

ER for $K = (1)(p) + (0)(1-p)$ $\Rightarrow$ ER for L = ER for K

$= p$

$1 - p = p$

$p = \frac{1}{2}$.

ex)

	L	R
U	-3, 3	-2, 2
D	-1, 1	0, 0

mixed up picking strategy with probability.

$\Downarrow$

genetic evolution!

$\downarrow$

run genetic algorithm that calculates evolving probability.

$\downarrow$

approach global min/max

nash equilibrium $\Longleftrightarrow$ evolutionary stable strategy.

$\downarrow$

self-enforcable

$\Downarrow$

- change the scale in the grid to approach some nash equilibrium for manipulation.

ex)

	U	D
U	-3, -3	1, -1
D	-1, 1	0, 0

- everyone is play D.
- Net gain in the population if one choose U.

{Dove - Hawk Game}

	H	D
H	-1/4, -1/4	2, 0
D	0, 2	1, 1

- If only Doves, Hawk gets 2 points,
  Doves gets 1 or 0 points
- If only Hawk, Hawk gets -1/4 points
  Doves get 0 pt.

Evolutionary Stable Strategy (ESS).

- In a population of a species, what will be the proportion of these strategy.
- With what probability will a species member will ulitize one strategy over the other.

Different strategy (behavior)

Bully : Dove pretend to be hawk

Assessor : assess each other and play roles as needed.

- Indicators of fitness - Female choose and male compete.

- Conspicuous consumption - wasteful behavior to show indicator of fitness to be competitive.
  (Biological Behavior).

ex)

	H	D
H	0, 0	$\frac{3}{2}$, $\frac{1}{2}$
D	$\frac{1}{2}$, $\frac{3}{2}$	1, 1

What is EES?

- Expected payoffs for the Dove and Hawk have to be same.
- Suppose H with σ
  D with $1-$

$$\text{expected}(H) = (0)(\sigma) + (1-\sigma)\left(\frac{3}{2}\right)$$

$$\text{expected}(D) = \left(\frac{1}{2}\right)(\sigma) + (1-\sigma)(1)$$

similar to Mixed strategy nash equilibrium.

$$\text{expected } H = \text{expected } D$$

$$\frac{3}{2} - \frac{3}{2}\sigma = \frac{1}{2}\sigma + 1 - \sigma$$

$$\sigma = \frac{1}{2}.$$

ex)

		② q L	$1-q$ R
① p	U	3, -3	-2, 2
$1-p$	D	-1, 1	0, 0

What should be the strategy for ① and ② for it to be stable mixed strategy nash equilibrium?

- ① should pick p such that payoff for player ② will be same if it plays L or R.
- expected (L) if ① choose U with p.
- . . .

$$E_2(L) = (-3)(p) + (1-p)(1)$$

$$E_2(R) = 2(p) + 0(1-p)$$

① plays U with 1/6

$$-3p + 1 - p = 2p$$

$$p = 1/6$$

$$E_1(U) = 3(q) + (-2)(1-q)$$

$$E_2(D) = (-1)(q) + 0(1-q)$$

② plays L with 1/3.

$$3q - 2 + 2q = -q$$

$$q = 1/3$$

Particle Swarm Optimization (PSO)

find the heighest peak.

idea:
- Directional velocities
- Move in the direction of
  - global best individual
  - personal best
  - random or previously chosen direction

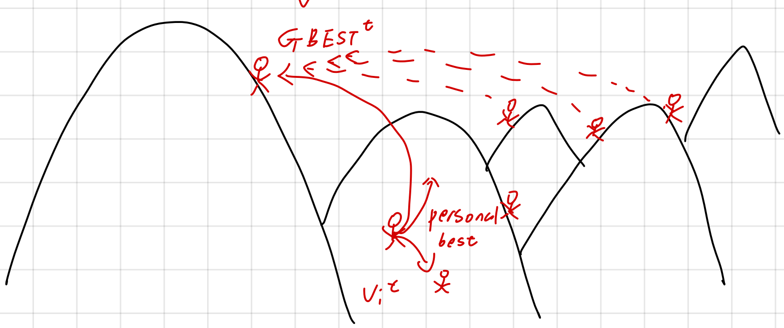
gradient descent
simulated annealing. } -> start with one point

genetic algorithm -> population spread across the func.

mutate, crossover, reproduce.

population fixed.

GBEST can change at every iteration.



velocity of i^{th} individual at the t^{th} step

advantage of idea:

individual can communicate with each other to share their information

X_i^t is the location of the ith person at the tth iteration.

$$\Rightarrow \boxed{X_i^{t+1} = X_i^t + V_i^t.}$$

- How to update V_i^{t+1}?

memory of better position.

$$V_i^{t+1} = \alpha V_i^t + \beta (\text{personal best direction}) + \gamma (\text{global best direction}).$$

- α is the most exploring variable.
  - Decrease the value of α every iteration.

- Guarantee to converge

global min?

local min?

decrease exploration mindset
(pretty much like simulated annealing).

Knapsack problem - dynamic programming.
minimize risk & maximize reward.

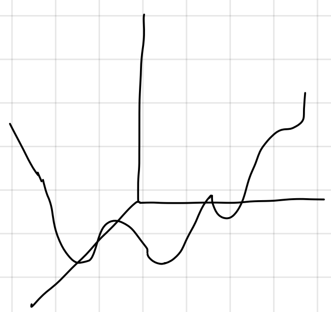
- β = randomness $= r_1 c_1$ where r_1 is a random # between 0 & 1 and c_1 is maximum weight to the personal best.
- γ = randomness $= r_2 c_2$ ...

So, $V_i^{t+1} = \alpha V_i^t + r_1 c_1 (P_i^t - X_i^t) + r_2 c_2 (G_i^t - X_i^t)$

decrease with t $(0, c_1)$ direction of personal best   random # $(0, c_2)$.   direction of global best.

$X_i^{t+1} = X_i^t + V_i^{t+1}$

Objective function: $7x^4 - 8y^4 + 3x^3 y - \cdots + 17$



optimize the function.

- Calculus - derivative / gradient - solve $\frac{df}{dx} = 0$, $\frac{df}{dy} = 0$
- gradient descent - $x_{t+1} = x_t + \alpha \frac{\partial f}{\partial x}(x_t)$

$Y_{t+1} = Y_t + \alpha \frac{\partial f}{\partial Y}(Y_t)$

- $(AX + BY + C)^4$ for some A, B, C

$\downarrow$

- $(AX_1 + BX_2 + CX_3 + \cdots + ZX_{26})^{38}$ $\swarrow$ 37 peaks on a 27 dimensional surface.

gradient descent not efficient
highly non-convex.

Seach space is massive!

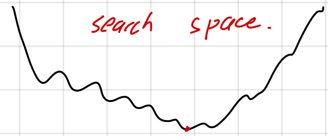
- Genetic Algorithm
- PSO

)-> population based!
individual communicate!

↓

implementation.

converge in search space.



Class particle:

position $(x_1, x_2, \cdots, x_d)$

velocity $(V_1, V_2, \cdots, V_d)$

current fitness $f(x_1, x_2, \cdots, x_d)$

best_position $(xb_1, xb_2, \cdots, xb_d)$

best_fitness $(f(\text{best position}))$.

Class particle swarm:

start random particles

give them random initial velocity.

→ compute everyone's fitness

compute everyone's new position. ie. $X_{t+1} = X_t + V_t$

update the velocity, $w = 0.99w$

"No Free Lunch Theorem" - No single optimization algorithm will be better than all the rest.

$$V_{t+1} = wV_t + r_1 c_1 (Pbest_i^t - x_i^t) + r_2 c_2 (Global^t - x_i^t)$$

eventually go to 0.

only these two direction exists and will converge to one position.

In PSO, No Crossover compared to genetic algorithm.
No Mutation
No Selection.

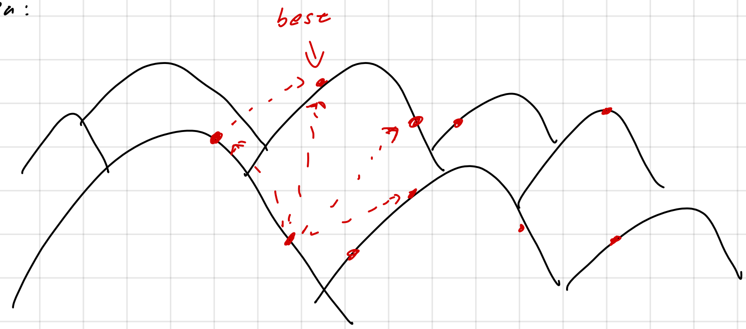
PSO does not guarantee global minimum/maximum.

Variation of PSO algorithm:

Firefly Algorithm:

idea: more than one particle will influence the velocity of the other. compared to PSO.

idea:



- Brightness of each firefly is the "Fitness".
- Fireflies are attracted to brightness of other fireflies.

For each firefly i
  For each firefly j
    if ith brightness < jth brightness:
      move $\frac{(x_j - x_i)}{r^2}$ towards j

$\beta_0 e^{-\gamma r_{ij}^2}$ ← brightness that ith firefly sees from jth firefly.

In convergence, we will get cluster of fireflies.

some exponential decay

Update: $x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} (x_j^t - x_i^t) + \alpha \varepsilon_i^t$

- β_0 is the brightness level at distance 0. generally set to 1.

moving in the direction of jth scaled by some factor

- γ is the light scattering factor. Larger it is. The less light (visibility) there is.

- r_{ij} is the euclidean distance between x_i & x_j.

$$\sqrt{(x_{1i} - x_{1j})^2 + \cdots (x_{di} - x_{dj})^2}$$

- α is a random exploring tendency.
- ε_i^t is a random velocity.

Firefly is good at finding multiple peaks.

- nash equilibrian in game theory.