## Gene Expression Algorithm

```
import math landom base Expansion Algorithm
FUNCS = {"1":2, ". ":2, "x":2 "/":2, "So":1, "coa":1)"
 APPLY . L
      "4": Lambda abiato, "-": lambda abiato,
       " lambda which, " lambda abta if abs(b) ele-12
       else a/b, "sin', lambée a math sin(a),
       "cos"; lambde as math. costa)
        "exp": lamble a math-exp (max(50, min(50, a)));
         log't lambda a imalkilog (obs (a)*1)
 TERMS - CHAP 4 17
def wit chrom (head, tout);
    9 - Forder chose (List (Junes) + TESNS) for in Parge (head)
    9+= (random chave (sexus) for in range (tail)
     C= Estandon-unform(=1,2) for his in ecomorate (g) if size)
 return ("ging, "all of many many many many
def devote (9):
    Sep = ( Ity need = 1; (=0)
  while need to and its len last;
    Sym = 901; seq-opperd((hym, i)); need -1
  H Sym to Funcs, need to Funcs laying a
    return Sug
def eval-enor (sequence) s
   Spos = 54 [3]
   T s in FUNCS.
     args - [] sign
     for a range (funcs as):
```

```
riseral ever (easiers); args amound (-)
     they return carrier to Complete the
     Except: return (Front (" 15") ))
  seture (will see he also a get look o) in)
                    the Distres
det sake eval (chia):
   thy: win etual expr (decide (th () 3), Contich Cicil)
   except return fleat ("in")
   return a Il mathe is minik (a) the shoot ("lat") which was
der mes (ANXIV):
  510-
   for my or mig (x,y).
     desale enal (chie)-4
     If not math itsinhiste (d): return 1812
   Peturn 6/len (x)
der mutale (ch. kes is tall, pm + 0.05, pc = 0+1);
   quetch(a) ch ["c] intental
   tor U in range (n):
       it random random ( ) com :
        9 (1) Francis Chara (Line France) .
       The send the THINS)
         if g[] == "( = (0) + (2) more -uniform [-2,5)
         end the cold col
        of random Partial 1 cpc ; c (1) + Fardom gouss (0,0+)
det crossour (0,4):
  no for (6Eg 3); (ut = random randed (4 m+1)
  that make (gas ph) =
      3-80 C3 "] [Cat] 1 80 CY ] (ut:]; == {}
     for its in enumerate (a) is
         H see"C": c(1)=[po(") if i cout else po(""])
         - get (f. former runiform (-2,2))
```

```
return { "g": g, "c": c}
                return make (a,b), make (b,a)
                                                                        Co ( and I fould recorded to Egones
 def select (pop, fit, k=3):
            i= min (randon: sample (range clen (pop), k), key = lambda j:fit[j]
           return {"g": pop(] ("g"](:), "c": dict (pap (:) ["c"])}
def evolve (f, target = (-3,3), pts = 64, popn=100, head=10, gens=100.
                         Cx: 0.7, Seed = None): ( and day made : dq = 10
           if seed: random: seed (seed) as ( seed ) as the seed of the seed o
           tail = head (max (FUNCS, values ()) - 1)+1
           X = [target [0] + (target [1] - target [0])"i /pts-1) for i in range (pts)
           Y = [f(x) \text{ for } x \text{ in } X]
          Pop = [init - chrom (head, tail) for in range (popn)]
           best, lit = float ("inf"), None" . (1) and interior short for
           for gen in range (gens):
                                                                                                                  (2) roll a rud or
                       fils = [mse (ind, x, y) for ind in pop]
                       b=min (range (popn), key = lambda i=fitx[i])
                       if fits [b] < best: best, fit = fits [b]
                       ["g": pop [b] ["g"] [:], "c": dict (pop [b] ["c"])}
                        new = [fit]
                        while len (new) < popn: my > () mobales mobales
                                    pl = select, (pop, fits); p2 = select (pop, fits)
                                    c) c2 = crossover (p),p2) if random. random () < cx
                                    else (p1, p2)
                                     mutate (ci, head, tail); mutate (cr, head, tail)
                                     new + = [case2] [in 186 : as a file
                       POP = new
           if (gent) % 20 == 0: print ("Gen", gent) "Best" best)
                                                                                                            1 (a) fall 62 3 90
        B_name .. : " .. main ! . !
              f=lambda 2:21 * 3-0.5 * x+ math.sin(sc)
             best, exo = evolve (f.gens=100) min. months of the interest of the
            print (Best exson: ", exx)
            point (Poeds: ", [safe eval (best, x) for x in [2,-1,0,1,2])
                                                                                    in summing in
             C L's" ] dy sels tossi it L's Dag 2 = [3] 3" 3"
```

stell : Fitness function: f(x) - x2 Encoding technique 0 to 31 Use choomosome of fixed length (genotype) ....

step 2. Initial population

5 No	(Gienotype)	Menotype	Value	Fitness	P
	Initial divionosome	(express*)	12	144	0.1247
2	13.3	2x	25	625	0.5411
3	1	a	5	25	0.0216
4	- 3.2	a-2	19	1 361	0.3125
500	y 10 10			- 1155	
ava				288.75	
nax				625	

Actual count Expected Count 800

Step3: Selection of mating pool

SNo.	Selected C		Crossoved point	Offspring	Phenotype
1	.75 +xx +xx		2		- V
2			1/3/2		
4			3		
	. 13	169	O CHOLDEN	mark or	
	24	516			
	27	729		7.	
	17	289			

Step4: Cossover = perform crossover randomly chosen gene posity (not some bits) max fitness after coossover=729

Step?	5. Mutation	mutation applied		offspring after	Phenotype
5-No-	offspring before mutation				
-	-	+->	-	-X.*	x+(x
2	+373	none >+		-a+	22 2+2+2
3	4X-				
4	+12	none		+22	212
-		avalue	Fitness		
		29	841		
		24	576		
		27	729		-4

400

Step 6 here Expression and evaluation decode each genotype -> phenotype calculate fitness EFQJ=841+576+729+400=2546

avg = 636.5

Max=841

Step 7: Itexate until convergence Repeat step 3 to 6 until fitness improvement is negligible so generation limit has reached

Output;

1000 generations

Genes [29.53, 29.82, 29.84, 28.57, 16.09, 21.83, 23.83, 30.81, 28.51, 26.22]

2:26.37

FW) = 695.45

```
Code:
print("Shreya Raj 1BM23CS317")
import math, random
# --- Function set ---
FUNCS = {"+":2,"-":2,"*":2,"/":2,"sin":1,"cos":1,"exp":1,"log":1}
APPLY = {
 "+":lambda a,b:a+b, "-":lambda a,b:a-b, "*":lambda a,b:a*b,
 "/":lambda a,b:a if abs(b)<1e-12 else a/b,
  "sin":lambda a:math.sin(a), "cos":lambda a:math.cos(a),
  "exp":lambda a:math.exp(max(-50,min(50,a))),
 "log":lambda a:math.log(abs(a)+1)
}
TERMS=["x","C"]
def init_chrom(head,tail):
  g=[random.choice(list(FUNCS)+TERMS) for _ in range(head)]
  g+=[random.choice(TERMS) for _ in range(tail)]
  c={i:random.uniform(-2,2) for i,s in enumerate(g) if s=="C"}
 return {"g":g,"c":c}
def decode(g):
 seq=[];need=1;i=0
 while need>0 and i<len(g):
   sym=g[i];seq.append((sym,i));need-=1
   if sym in FUNCS: need+=FUNCS[sym]
```

```
i+=1
  return seq
def eval_expr(seq,i,x,c):
 s,pos=seq[i]
 if s in FUNCS:
    args=[];j=i+1
   for _ in range(FUNCS[s]):
     v,j=eval_expr(seq,j,x,c);args.append(v)
   try:return (APPLY[s](*args),j)
   except: return (float("inf"),j)
  return (x if s=="x" else c.get(pos,0),i+1)
def safe_eval(ch,x):
  try:v,_=eval_expr(decode(ch["g"]),0,x,ch["c"])
 except:return float("inf")
  return v if math.isfinite(v) else float("inf")
def mse(ch,X,Y):
  s=0
 for x,y in zip(X,Y):
   d=safe_eval(ch,x)-y
   if not math.isfinite(d): return 1e12
    s+=d*d
 return s/len(X)
```

```
def mutate(ch,head,tail,pm=0.05,pc=0.1):
 g,c=ch["g"],ch["c"];n=len(g)
 for i in range(n):
   if random.random()<pm:</pre>
     g[i]=random.choice((list(FUNCS)+TERMS) if i<head else TERMS)
     if g[i]=="C":c[i]=random.uniform(-2,2)
     elif i in c:del c[i]
 for i in list(c):
   if random.random()<pc:c[i]+=random.gauss(0,0.1)
def crossover(a,b):
 n=len(a["g"]);cut=random.randint(1,n-1)
 def make(pa,pb):
   g=pa["g"][:cut]+pb["g"][cut:];c={}
   for i,s in enumerate(g):
     if s=="C":c[i]=(pa["c"] if i<cut else pb["c"]).get(i,random.uniform(-2,2))
   return {"g":g,"c":c}
 return make(a,b),make(b,a)
def select(pop,fit,k=3):
 i=min(random.sample(range(len(pop)),k),key=lambda j:fit[j])
 return {"g":pop[i]["g"][:],"c":dict(pop[i]["c"])}
def evolve(f,target=(-3,3),pts=64,popn=100,head=10,gens=100,cx=0.7,seed=None):
 if seed:random.seed(seed)
 tail=head*(max(FUNCS.values())-1)+1
```

```
X=[target[0]+(target[1]-target[0])*i/(pts-1) for i in range(pts)]
 Y=[f(x) for x in X]
 pop=[init_chrom(head,tail) for _ in range(popn)]
 best,fit=float("inf"),None
 for gen in range(gens):
   fits=[mse(ind,X,Y) for ind in pop]
   b=min(range(popn),key=lambda i:fits[i])
   if fits[b]<best:best,fit=fits[b],{"g":pop[b]["g"][:],"c":dict(pop[b]["c"])}
   new=[fit]
   while len(new)<popn:
     p1=select(pop,fits);p2=select(pop,fits)
     c1,c2=crossover(p1,p2) if random.random()<cx else (p1,p2)
     mutate(c1,head,tail);mutate(c2,head,tail)
     new+=[c1,c2]
    pop=new
   if (gen+1)%20==0:print("Gen",gen+1,"Best",best)
  return fit, best
if __name__=="__main__":
 f=lambda x:x**3-0.5*x+math.sin(x)
 best,err=evolve(f,gens=100)
 print("Best error:",err)
 print("Preds:",[safe_eval(best,x) for x in [-2,-1,0,1,2]])
```

## Output:

```
Shreya Raj 1BM23CS317
Gen 20 Best 0.2711388144567344
Gen 40 Best 0.08276309290483551
Gen 60 Best 0.02638736211355352
Gen 80 Best 0.0038234107968136122
Gen 100 Best 0.0035796711491496228
Best error: 0.0035796711491496228
Preds: [-7.947649551427884, -1.2578413596603832, 0.0, 1.2578413596603832, 7.947649551427884]
```