Universitat de Girona

Departament d'Electrònica, Informàtica i Automàtica

Assignatura:
Disseny de sistemes de supervisió

Pràctica 5:
"ALGORISMES GENÈTICS"

Pràctica 5:

"ALGORISMES GENÈTICS"

Objectius de la pràctica:

- Comprensió dels conceptes bàsics dels algorismes genètics.
- Utilització de la Genetics Toolbox de MATLAB
- Disseny, prova i ajust d'un sistema senzill basat en algorismes genètics.
- Entendre quina és l'aplicabilitat del sistemes basats en algorismes genètics

Contingut:

Introducció. Els algorismes genètics: la modelització de l'evolució

El principi bàsic dels algorismes genètics consisteix en l'emulació de l'evolució biològica de les espècies. D'acord amb les teories de Darwin es produeix una adaptació al medi per mor d'una evolució de les espècies. Les espècies millor adaptades són les que prosperen en termes d'una millor reproducció (els millors individus tenen major probabilitat de reproduir-se). Posteriorment, amb els treballs d'en Mendel i ja cap el segle XX, es va lligar el tema d'evolució amb una millora del codi genètic. Aleshores, una comunitat d'individus es considera una comunitat de codis genètics que van evolucionant intentant optimitzar la seva *adaptació* al medi.

La modelització en informàtica de la evolució genètica és la que es descriu a continuació. Els gens són símbols que tenen codificades les solucions possibles a un problema d'optimització, definint-se *l'adaptació* com una funció que es vol optimitzar, i els individus són *possibles solucions* que de forma evolutiva i agrupats en poblacions intenten adaptar-se al medi definit per l'esmentada funció. Els operadors de *selecció*, *reproducció*, *mutació*, i altres es defineixen a similitud biològica seleccionant individus de forma iterativa per a cada població, creuant codis genètics entre individus i mutant el codi dels individus seguint variables aleatòries.

Finalment, els algorismes genètics es poden interpretar com una forma elegant d'optimitzar mitjançant aproximacions estocàstiques (optimitzar de forma aleatòria).

En aquesta pràctica s'aplicaran els algorismes genètics en dues parts diferenciades. Primer com a eines d'optimització de funcions, i després com a mecanisme per capturar el coneixement necessari per a què un robot futbolista decideixi on anar.

PART 1: Optimització de funcions

Es tracta d'optimitzar les funcions següents:

1) Potència. $f = x^{10}$, $x \in [0, 1]$
2) Banana. $f(x) = 100 \cdot \sqrt{x(2) - x(1)} + \sqrt{1 - x(1)}$
3) Pics i valls. $z = f(x, y) = \sum_{i=1}^{n} H_i \cdot e^{-(x - X_i^2) - (y - Y_i^2)}$

L'objectiu de la primera funció és familiaritzar-se amb la toolbox de genètics de MATLAB . En la segona veurem les limitacions dels algorismes genètics, que s'estudiaran amb més cura gràcies al model proporcionat amb la tercera funció.

Observar que per optimitzar s'entén maximitzar o minimitzar una funció. Així, per conveniència, en la primer i tercera funció s'intentarà trobar un màxim, mentre que en la segona es cercarà un mínim.

Potència

Anem a minimitzar la següent funció[1]:

$$\mathbf{F = x^{10} , \ x \in [0, 1]}$$

[1] Aquest exemple s'ha pres de: Genetic Algorithms in Search, Optimization, and Machine Learning, David E. Goldberg, Addison-Wiley Publishing Company, Inc., 1989. (Capítol 3).

Per trobar el màxim d'aquesta funció dins del domini d'x, MATLAB proporciona la toolbox genetic.m.

[x, stats, options, bf, fgen, lgen] = **genetic**(fun, x0, options, vlb, vub, bits);

Els paràmetres de la funció són:

Fun ⇒ la funció a optimitzar
X0 ⇒ la població inicial (per defecte es genera aleatòriament)
Options
Vlb ⇒ valor mínim de la x
Vub ⇒ valor màxim de la x
Bits ⇒ nombre de bits amb els que es codifiquen els valors

Hi ha fins a 14 opcions, a les quals se'ls hi pot assignar un valor per defecte mitjançant foptions:

options = foptions([1 1e-3]); *(valors per defecte per a rutines d'optimització)*

Detall d'algunes opcions més interessants:

Options(2) ⇒ valor de convergència (per defecte 0.001)
Options(11) ⇒ grandària de la població (30 per defecte)
Options(12) ⇒ probabilitat de creuament (1 per defecte)
Options(13) ⇒ probabilitat de mutació (0 per defecte)
Options(14) ⇒ màxim nombre de generacions (100 per defecte)

La funció genetic retorna 5 valors:

x ⇒ valor d'x per l'òptim trobat
stats ⇒ estadístiques sobre totes les generacions
options ⇒ actualitza alguns dels paràmetres, com ara
options(10) ⇒ nombre total de generacions executades
bf ⇒ valor màxim trobat per a la funció (*best fitness*)
fgen ⇒ població de la primera generació (*first generation*)
lgen ⇒ població de la darrera generació (*last generation*).

Per trobar el màxim de la funció potencial descrita anteriorment, s'assignarà els valors següents als paràmetres:

options(13) = 0.03; *(probabilitat de mutació)*
options(14) = 50; *(número màxim de generacions per defecte)*
vlb = 0; *(valor mím de l'espai de cerca)*
vub = 1; *(valor màxim de l'espai de cerca)*
bits = 30; *(número de bits a codificar els gens)*

A continuació, es crida a la funció d'algorismes genètics **genetic** de la següent manera

[x, stats, options, bf, fgen, lgen] = **genetic**('x^10', [ ], options, vlb, vub, bits);

El resultat de la cerca es mostra en la pantalla. Es proporciona una línia per a cada generació, i en cada línia es mostren els valors corresponents a les dades estadístiques de la població de la generació:

Núm de generació, valor màxim, valor mínim, valor mitjà, desviació estàndard

L'algorisme pot acabar de dues formes diferents:

1. Perquè s'hagin produït totes les generacions indicades en el màxim (options(14)).
2. Perquè s'hagi produït una convergència. La condició de convergència ve determinada per alguna les següents condicions :
   - Si s'han produït més de 5 generacions

$$\frac{max(generacióActual) - max(generacióActual - 5)}{max(generacióActual)} < valor\_convergència$$

On max(x) és el valor màxim de la generació x, generacióActual-5 és la generació 5 llocs enrera de la generació actual, i valor_convergència és el valor explicitat en el paràmetre options(2).

- Si s'han produït menys de 5 generacions
  En aquest

$$\frac{mitjana(generacióActual) - mitjana(generacióActual - 1)}{mitjana(generacióActual)} < valor\_convergència$$

En aquest cas s'agafa la mitjana (enlloc del màxim) i el valor de la generació immediatament anterior a l'actual.

➡ Executa l'algorisme i comprova quina ha estat la condició d'acabament.

➡ Executions múltiples de l'algorisme donen resultats diferents. Per què? Reflexiona la teva resposta en funció del que s'ha explicat a les classes de teoria.
Per a cada execució proporciona els valors obtinguts per:

a) La solució òptima
b) La població incial i final
c) El nombre total de cops que la funció d'adaptació ha estat avaluada. Això és:
   [(Nombre de generacions produïdes)+1]*grandària_població
   Recordar que la població incial també requereix d'avaluació.

Banana.

Aquest exemple probablement mostrarà que (recordem que els AG són probabilístics) encara que GA simples funcionen apropiadament per trobar una bona regió en la que hi pugui existir un mínim (màxim) global (maximum), no tenen les característiques de convergència de les tècniques basades en gradient un cop estem aprop del mínim (màxim).

La funció banana és:

$$f(x) = 100 \cdot \sqrt{x(2) - x(1)} + \sqrt{1 - x(1)}$$

I l'objectiu és trobar un mínim de la funció en el domini x(1)∈ [-2, 2] and x(2)∈ [-1, 3].

Com que l'AG maximitzarà una funció, per fer una minimització es pot fer:

```
f = '- (100*(x(2)-x(1))^2 + (1-x(1))^2)'
```

Però genetic.m està implementada de forma que espera que la funció d'adaptació sigui sempre positiva. Aleshores es pot sumar una constant, quedant finalment la funció com segueix:

```
f = '1000 - (100*(x(2)-x(1))^2 + (1-x(1))^2)'
```

Per visualitzar la funció podeu fer servir el codi proporcionat a plot_banana.m i que es mostra a continuació:

```
% <<  Please be patient while the contour plot is generated. >>
%
figure
xx = [-2:0.125:2];
yy = [-1:0.125:3];
[x,y]=meshgrid(xx,yy) ;
meshd = 100.*(y-x.*x).^2 + (1-x).^2;
conts = exp(3:20);
contour(xx,yy,meshd,conts)
hold on
```

```
xlabel('x1')
ylabel('x2')
title('Minimization of the Banana function')
plot(1,1,'o')
text(1.1,1,'Solution')
```

Es codificaran els valors de x(1) i x(2) en 16 bits, i es posaran les opcions següents:

```
bits = [16 16];
vlb = [-2 -1];
vub = [2 3];
options = foptions([1 -1]);
options(13) = 0.0333;
```

➡🎂 Executa l'algorisme genètic.

```
[x,stats,options,bf,fgen,lgen] = genetic(f,[],options,vlb,vub,bits);
```

Quin són els resultats? Quin és el millor punt trobat? Quantes avaluacions de la funció d'adaptació s'han realitzat?
El que probablement has vist és que l'AG ha convergit ràpidament (en unes quantes generacions) dins d'una regió a prop del mínim de la funció.

Pics i valls.

De l'exercici anterior podem concloure que els AG estan millor dotats per determinar les regions on pot existir un màxim global. Un cop trobades les regions doncs fora millor aplicar-hi tècniques d'optimització més clàssiques (eines de càlcul de gradient com el *hill climbing*).

Per exemple, considera el problema de determinar el pic més alt a una regió amb múltiples pics. Es pot descriure tal regió de forma matemàtica mitjançant una adaptació de l'equació que genera els pics. Específicament, considera una equació de la forma:

$$z = f(x,y) = \sum_{i=1}^{n} H_i \cdot e^{-(x-X_i^2)-(y-Y_i^2)}$$

Si **n** fos 1, aquesta podria ser l'equació d'una colina simètrica amb un pic d'alçada H centrat al punt (X, Y). Es pot fer la colina asimètrica aplicant-hi constants multiplicatives als termes de l'exponent. Perfils de colines més complicats es poden crear fent H = H(x,y).

En aquesta pràctica, però és seguirà un model senzill: **n** colines de diferents alçades i emplaçades a prop entre elles, en resulta un topologia suficientment interessant. Es generaran a l'atzar 10 colines dins del rang 0<x<6 i el rang 0<y<6. Les alçades seran generades també de forma aleatòria entre 10 i 20.

```
X = 6*rand(10,1);
Y = 6*rand(10,1);
H = 10*rand(1,10) + 10;
```

Aleshores, si **exp** és un vector columna, llavors l'operació **H*exp** es podrà fer transposant H.

Per qüestions de notació del programa genetic.m, X, Y i H es reanomenen P1, P2 i P3 respectivament:

```
P1 = X;
P2 = Y;
P3 = H;
```

Quedant la funció definida com segueix:

```
f = 'P3*exp(-(x(1)-P1).^2 -(x(2)-P2).^2)'
```

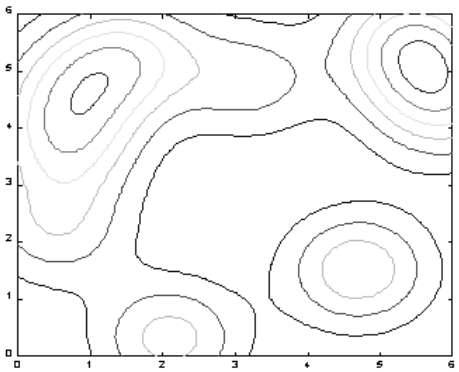
Primer de tot, es farà una visualització del contorn de la funció:

```
pts = 0:0.05:6;
x_y = ones(length(pts),1)*pts;

z = 5*ones(size(x_y));
for i=1:10,
   z = z + H(i)*exp(-(x_y-X(i)).^2 -(x_y'-Y(i)).^2);
end
hold off

contour(x_y,x_y',z)
```

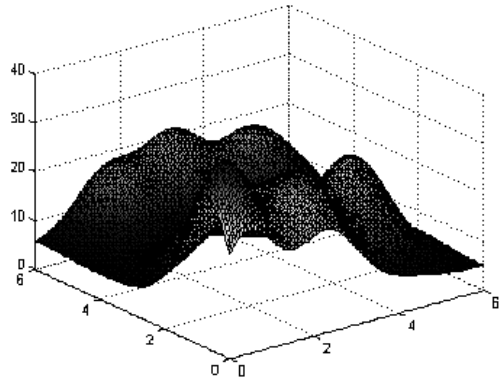
Un contorn possible és:



I després visualitzeu la superfície (podeu usar també plot_pics_i_valls.m):

```
surf(x_y,x_y',z);
hold on
```

Una superfície possible és:



➡🎂 Executa l'algorisme genètic amb les opcions següents i inicialitzant les variables vlb, vub, bits adequadament.

```
options = foptions(1);
options(13) = 0.0333;
 [x,stats,options,bf,fg,lg] = genetic(f,[],options,vlb,vub,bits,X,Y,H);
```

Vegeu com les matrius X, Y i H són passades com a arguments addicionals
Quin és el valor màxim trobat? Per a quin valor?

➡ Compara aquest amb el maxim punt de la malla dibuixada:

```
[i,j] = find(max(max(z))==z);
z(i,j)
```

Que es troba a:

```
0.05*[i j]
```

PART 2: Aplicació als robots futbolistes.

Dissenyar una població genètica que intenti capturar el coneixement per entrenar a un robot futbolista a decidir on anar.

Feu servir les directrius de codificació esmentades a la classe de teoria.

Finalment recordeu que disposeu en MATLab de la funció **decode** que converteix nombres binaris en variables.