Метод роя частиц для задачи глобальной оптимизации

snegovick — Чт, 30/09/2010 - 17:49

Метод оптимизации с помощью роя частиц (Particle Swarm Optimization, далее PSO), базирующийся на моделировании поведения популяции частиц в пространстве параметров задачи оптимизации, был предложен в работе [1].

Предлагаемый метод привлекателен простотой реализации и тем, что в процессе вычисления не используется градиент. Метод может использоваться для решения многих задач, включая обучение нейросетей, задачи поиска минимума функции, а также задач, типичных для генетических алгоритмов.

PSO, как и все алгоритмы, принадлежащие к семейству эволюционных алгоритмов, является стохастическим, не требующим вычисления градиента, что позволяет использовать PSO в случаях, где вычисление градиента невозможно, либо имеет высокую вычислительную сложность.

Описание алгоритма PSO

Алгоритм PSO использует для решения рой частиц, где каждая частица представляет собой возможное решение задачи оптимизации. Пусть s - количество агентов в рое. Каждая i-ая частица может быть представлена как объект с рядом параметров.

x_i - положение частицы

v_i - скорость частицы

y_i - личное наилучшее положение частицы

Личное наилучшее положение частицы - положение частицы с наилучшим значением оценочной функции, которое когда-либо посещала частица. Пусть f - функция, которую необходимо минимизировать, тогда выражение для наилучшего личного положения в зависимости от времени:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases}$$

Существует две версии базового алгоритма PSO, называемые gbest и lbest. Разница заключается в том, с какими набором частиц взаимодействует каждая частица. Обозначим символом $\hat{y}$ это

взаимодействие. Определение $\hat{y}$ в случае gbest представлено в следующем выражении:

$$\hat{y} \in \{y_0(t), y_1(t) \ldots y_s(t)\} | f(\hat{y}(t)) = \min\{f(y_0(t)), f(y_1(t)), \ldots, f(y_s(t))\}$$

Данное выражение говорит о том, что $\hat{y}$ - лучшая позиция когда-либо посещенная кем-либо из роя.

Стохастическая составляющая алгоритма представлена двумя случайными параметрами $r_1 \sim U(0,1)$ и $r_2 \sim U(0,1)$, которые масштабируются с помощью постоянных коэффициентов ускорения c_1 и c_2, отвечающих за величину шага, который может сделать частица за одну итерацию времени. Как правило $c_1, c_2 \in (0,2]$. Скорость частицы на i-м шаге расчитывается отдельно для каждого измерения $j \in 1..n$, таким образом $v_{i,j}$ обозначает j-е измерение вектора скорости i-й частицы. Расчет j-го компонента вектора скорости i-й частицы на $t+1$ шаге производится по формуле:

$$v_{i,j}(t+1) = w_c v_{i,j}(t) + c_1 r_1(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_2(t)\left[\hat{y}(t) - x_{i,j}(t)\right]$$

Таким образом, c_2 управляет воздействием глобального лучшего положения, а c_1 управляет воздействием личного лучшего положения на скорость каждой частицы. Для улучшения сходимости алгоритма вводится коэффициент инерции w_c [2]. Положение каждой частицы в i-м измерении вычисляется по формуле

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Подготовительная часть алгоритма заключается в следующем :

1. Присвоить координатам $x_{i,j}$ случайные значения в пределах $[-x_{max}, x_{max}]$ для всех $i \in 1..s$ и $j \in 1..n$,
2. Инициализировать векторы скоростей нулями,
3. Присвоить y_i значения x_i.

Пример решения задачи оптимизации с помощью алгоритма PSO на maxima

Пусть задана функция.

$$f(x_1, x_2) = x_1^2 + x_2^2$$

Необходимо найти минимум этой функции на промежутке $x_1, x_2 \in [-10, 10]$. Известно, чтоминимум находится в точке (0).
Решение:

```
f(x1,x2):= x1^2+x2^2;

ll: -10;
ul: 10;
numpart: 20;
numdims: 2;
wc: 0.9;
c1: 1.8;
c2: 1.9;

xv: ematrix(numpart,numdims,0,1,1);
vv: ematrix(numpart,numdims,0,1,1);
fitgbest: 2000;
fitlbest: ematrix(numpart,1,0,1,1);
fitlbest: fitlbest+fitgbest;
xlbest: ematrix(numpart,numdims,0,1,1);
xgbest: ematrix(1,numdims,0,1,1);

```

```
fitx: ematrix(numpart,1,0,1,1);

for i:1 step 1 thru numpart do (
    for j:1 step 1 thru numdims do (
        xv[i][j]: ll + random((ul-ll))
    )
)$

min: 0$
minind: 0$
display (xv);

globalbest : [];

for i:1 step 1 thru 20 do (
    for j:1 step 1 thru numpart do (
        fitx[j][1]: f(xv[j][1], xv[j][2]),
    if fitx[j][1] < fitlbest[j][1] then (
       fitlbest[j][1] = fitx[j][1],
       for k:1 step 1 thru numdims do (
           xlbest[j][k]: xv[j][k]
       )
    )
    ),

   min: fitx[1][1],
   minind: 1,
   for j:1 step 1 thru numpart do (
       if fitx[j][1]<;min then (
          minind: j,
      min: fitx[j][1]
       )
   ),

   dispm: addcol(xv,fitx),
   display (dispm),

   if min<;fitgbest then (
      fitgbest: min,
      for j:1 step 1 thru numdims do (
            xgbest[1][j]: xv[minind][j]
      )
   ),

   display (fitgbest),
   globalbest : append(globalbest, [[i,fitgbest]]),

   for j:1 step 1 thru numpart do (
       for k:1 step 1 thru numdims do (
           r1: (random(1000))/1000,
           r2: (random(1000))/1000,
           vv[j][k]: wc*vv[j][k]+c1*r1*(xlbest[j][k]-xv[j][k])+c2*r2*(xgbest[1]
           xv[j][k]: xv[j][k]+vv[j][k]
       )
   )
)$

display(globalbest);

plot2d([discrete, globalbest], [y,0,50], [psfile, "globalbest.eps"]);
```