Enxames de Partículas

Computação Natural
Gisele L. Pappa

Inspiração

- Proposto por Eberhart and Kennedy em 1995, inspirado no comportamento social de pássaros e peixes

Enxames de partículas (PSO)

- Inspiração vem de um conjunto de conceitos bem diferentes dos de insetos sociais:
  - Psicologia social
  - Interações entre indivíduos
  - Inteligência é resultado de interações entre os indivíduos (e adaptação)
- Enxames de partículas consistem em um conjunto de algoritmos inspirados no comportamento natural de grupos

Idéias Básicas - PSO

- Em PSO, partículas não tem autonomia ou especialização
  - Assim, é uma metáfora em alto nível (pessoas são todas diferentes)
- Com pessoas, interações levam a mudanças
- Comportamento social ajuda os indivíduos de uma espécie a se adaptarem ao ambiente

Ideias Básicas - PSO

- Criados para simular sociedades de indivíduos
  - Esses indivíduos estão trabalhando em um problemas...
  - ... e sendo influenciados por seus vizinhos
- Altamente influenciados por interações entre humanos e entre animais
- Ideias são transmitidas, modificadas, afetam as pessoas e podem ser modificadas

Princípios básicos de adaptação cultural (de acordo com a psicologia)

- Avaliar – todo organismo precisa avaliar estímulos
  - Pré-requisito para a aprendizagem – detectar estímulos positivos ou negativos
- Comparar – as pessoas se comparam umas com as outras
  - Essas comparações nos levam a melhorar
- Imitar – as pessoas tendem a imitar o comportamento de outras (especialmente das bem sucedidas)
  - Poucos animais são capazes de imitar uns aos outros

Idéia básicas - PSO

- Partícula representa um indivíduo com 2 tipos de informação:
  - Experiência própria (ele sabe que escolhas foram as melhores no passado)
  - Conhecimento de outros indivíduos, suas escolhas e sucesso associado
- Utiliza o conceito de uma partícula com uma velocidade se movendo num hiperespaço de busca (velocidade e posição)

Componentes Principais

- Um conjunto de partículas
- Cada partícula representa uma solução candidata
  - Elementos da partícula representam parâmetros a serem otimizados
- Uma partícula possui
  - Uma coordenada no espaço de busca ($\mathbf{x}_i$)
  - Uma velocidade (v_i)

O algoritmo PSO

- Velocidade é responsável pelo processo de otimização
- Esquecer ou aprender são vistos como aumentar ou diminuir o valor da posição da partícula
- Conceito de melhor posição (fitness)
  - Altera a velocidade para nos levar a essa posição
- Indivíduos interagem com um número *k* de vizinhos

O Algoritmo PSO

Cria uma população inicial aleatoriamente
(partículas são colocadas em posições aleatórias com velocidades aleatórias)
Enquanto (critério de parada não satisfeito)
para cada partícula
Avalia o quão boa ela é (Se ela for melhor que a melhor partícula até o momento, ele se torna a melhor)
Avalia a vizinhança
Atualiza a velocidade V_i
Atualiza a posição da partícula
fim para
fim enquanto

O algoritmo PSO

- Atualizando a posição da partícula:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t)$$

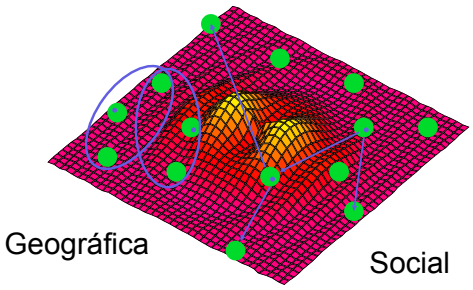
$$\mathbf{v}_i(t) = \omega \mathbf{v}_i(t-1) + c_1\varphi_1(\mathbf{p}_i - \mathbf{x}_i(t-1)) + c_2\varphi_2(\mathbf{p}_g - \mathbf{x}_i(t-1)$$

Onde:

$\mathbf{P}_i$ = local onde a melhor fitness do indivíduo foi encontrada

$\mathbf{P}_g$= local onde a melhor fitness do melhor vizinho foi encontrada

Vizinhança



A Velocidade

$$\mathbf{v}_i(t) = \omega\mathbf{v}_i(t\text{-}1) + c_1\varphi_1(\mathbf{p}_i\text{-}\mathbf{x}_i(t\text{-}1)) + c_2\varphi_2(\mathbf{p}_g\text{-}\mathbf{x}_i(t\text{-}1))$$

- i representa uma dimensão da partícula
- ω : constante de inércia (mantém partículas em movimento)
- $C_{1,2}$: constantes que afetam o quanto cada uma das melhores partículas afetam a partícula atual
- φ_1 escalar aleatório (taxa de aprendizagem)

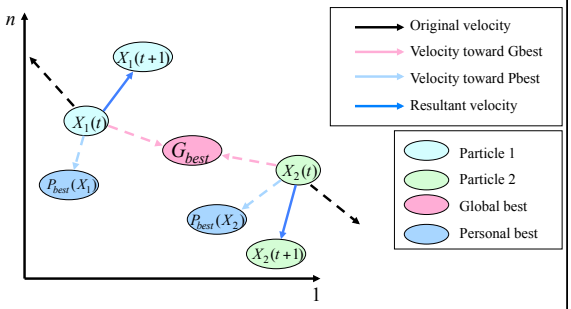
A Velocidade

- ω : constante de inércia (mantém partículas em movimento)
- Se >= 1, velocidade aumenta ao longo do tempo
  - Swarm diverge
  - Partículas não conseguem mudar de posição para encontrar regiões mais promissoras
- Se 0 < ω < 1, partículas desaceleram
  - Convergência depende dos valores de c_1 e c_2

A Velocidade

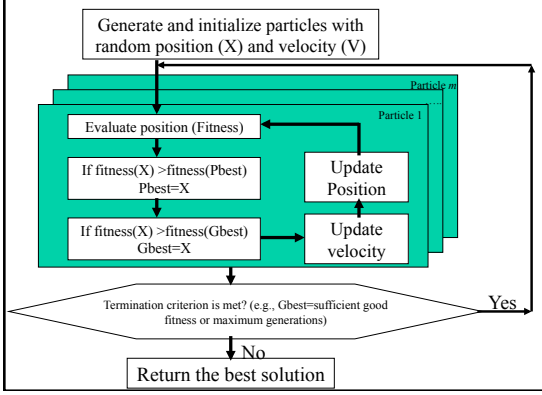
- c_1 e c_2 : Controla *exploration* e *exploitation*
- *Exploration* – habilidade de explorar regiões do espaço de busca
- *Exploitation* – habilidade de se concentrar em uma região ao redor de uma área promissora e refinar a busca

Movimentos das Partículas

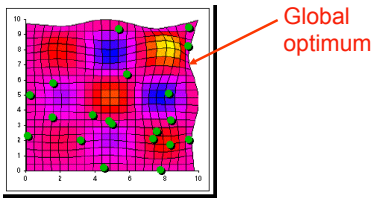


Partículas 1 e 2 vão em direção ao Gbest e a seu Pbest no espaço n-dimensional

PSO



Demo



Tipos de PSO

- Existem dois tipos de algoritmos básicos
  - Um para dados contínuos
    - Mais utilizado e, consequentemente, mais estudado
  - Outro para dados binários

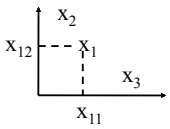
PSO para dados contínuos

- Uma partícula consiste em um vetor de números reais (valores das variáveis)
- Notação para a partícula *i*:

  $x_i = < x_{i1}, \ldots, x_{ij}, \ldots, x_{iD}>$

  onde x_{ij} = o valor da coordenada j da partícula i

  e $j = 1,\ldots,$ número de variáveis (número de dimensões: D)



Um exemplo simples de 3 partículas em um espaço 2D

PSO para dados contínuos

- A velocidade de uma partícula é um vetor de números reais adicionado a posição da partícula em um espaço de tempo (uma iteração)
- Atualizando a posição da partícula:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t\text{-}1) + \mathbf{v}_i(t)$$

$$\mathbf{v}_i(t) = \omega\mathbf{v}_i(t\text{-}1) + c_1\varphi_1(\mathbf{p}_i\text{-}\mathbf{x}_i(t\text{-}1)) + c_2\varphi_2(\mathbf{p}_g\text{-}\mathbf{x}_i(t\text{-}1)$$

PSO para dados contínuos

- Ajuste de posição:

  $x_i(t) = x_i(t-1) + v_i(t)$, onde

  $x_i(t)$, $x_i(t-1)$ são as posições da partícula *i* nos tempos t e t – 1

  $v_i(t)$ é a velocidade da partícula *i* no tempo *t*
- x_i e v_i são vetores de números reais, então a equação acima pode ser reescrita como:

  para cada dimensão (variável) $j, j = 1,\ldots,D$:

  $$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t)$$

PSO para dados contínuos

- Para evitar que a partícula saia do espaço de busca, devemos limitar a velocidade
- Para cada dimensão (variável) $j, j = 1,\ldots, D$:

  se $v_{ij}(t) > V_{max}$ então

  $v_{ij}(t) = V_{max}$

  senão se $v_{ij}(t) < -V_{max}$ então

  $v_{ij}(t) = -V_{max}$

PSO Parameters

- Number of particles : (10—50) are reported as usually sufficient.
- C1 (importance of personal best)
- C2 (importance of neighbourhood best)
- Usually C1+C2 = 4. No good reason other than empiricism
- Vmax – too low, too slow; too high, too unstable.

Parâmetros do PSO

- Número de partículas na população e número de iterações
  - Em geral, quanto mais partículas e quanto mais iterações, maior a chance de encontrar uma solução (quase)-ótima, mas mais lento o PSO vai ser
- Fator de inércia *w:* menor que 1, mas não muito pequeno
- O que acontece se:
  - w é maior que 1?
    - Velocidade aumenta rapidamente com o tempo
  - w é muito pequeno?
    - Partícula para rápido, levando a convergência prematura
- w pode ser fixo ou variado durante a busca. Podemos começar com w = 0.7 e diminuir esse valor linearmente ao longo das interações

PSO para Dados binários

- Uma solução candidata é representada por um string binário
- prob(x_{ij} = 1) representa a probabilidade da partícula *i,* variável (dimensão) *j* ter valor 1.
  - Isso também significa que prob(x_{ij} = 0) = 1 – prob(x_{ij} = 1), uma vez que a soma das probabilidades de um 1 e um 0 ocorrerem devem somar 1 por definição
- As noções de "posição" e "velocidade" de uma partícula para dados contínuos são re-definidas
- "posição" representa o estado da string de bits
  - x_{ij}= valor do bit da partícula *i* na dimensão *j*
- "velocidade" é a tendência de se escolher o estado 1
  - v_{ij} = tendência da partícula *i*, dimensão *j* receber o valor 1

PSO para Dados binários

- Em dados binários, a velocidade v_{ij} é utilizada para atualizar o valor de x_{ij} (estado do bit *j* na partícula *i*) de maneira probabilística:

  prob(x_{ij} = 1)(t) é calculada através de um mapeamento do valor da velocidade no intervalo [0…1], chamada função de "squashing":
- prob(x_{ij} = 1)(t) = $\frac{1}{1 + e^{(-V_{ij}(t))}}$, onde e = 2.718…
- Para converter a probabilidade em um estado de bit, para cada bit *j* da partícula *i*, gere um número aleatório $r_{ij}(t)$, e:

  IF $r_{ij}(t) < prob(x_{ij} = 1)(t)$ THEN $x_{ij}(t) = 1$; ELSE $x_{ij}(t) = 0$

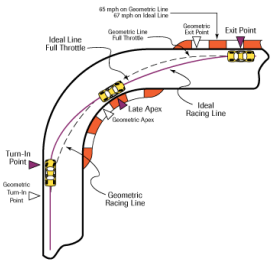
Aplicações

- Análise de dados
  - Agrupamento e classificação
- Swarm bots
- Swarm music

Otimizando os parâmetros de um Jogo de Corrida

Objetivo

- Encontrar o melhor caminho a ser seguido pelos carros controlados pelo computador em um jogo de corrida



Parâmetros

- Cada partícula representa um conjunto de 10 parâmetros que representam o caminho a ser percorrido
- 20 partículas são distribuídas no espaço
- O algoritmo roda 100 iterações
- A fitness é dada pelo tempo que o carro leva para completar uma volta

Resultados

- Jogador humano leva em média 70s para completar uma volta
- Na primeira iteração do algoritmo, a melhor partícula tinha um tempo de 80s, e a média de todas as partículas era de 115 s
- Ao final da execução, a melhor partícula completava a volta em 63s, tornando o computador imbatível por um usuário humano

Agradecimentos

- Alguns desses slides foram retirados das aulas de computação natural de Alex A. Freitas e Jon Timmis

Bibliografia

1. http://www.swarmintelligence.org/
2. http://www.swarmintelligence.org/tutorials.php
3. Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995
4. http://www.gamasutra.com/features/20051213/villiers_01.shtml