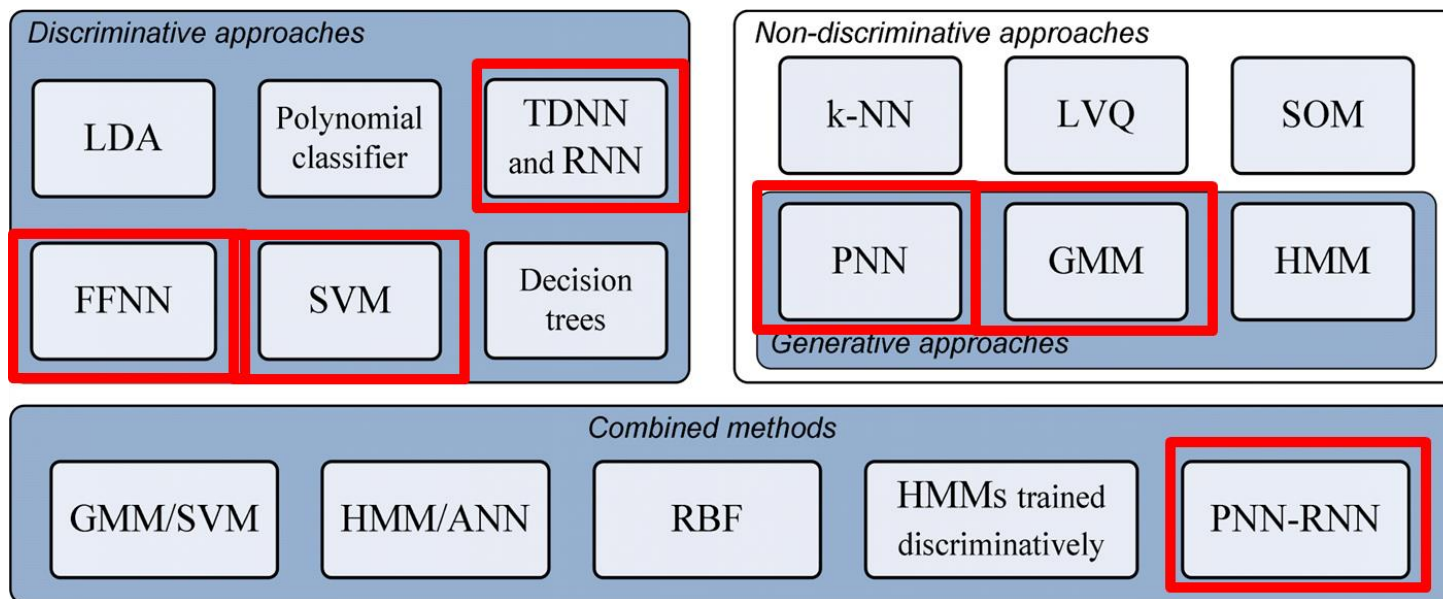


Тема 15: Методи за машинно обучение. Диференциална еволюция (DE). Оптимизация с рояк от частици (PSO)



Получихме основни познания за

- ❑ Дискриминативни и генеративни методи за класификация
- ❑ Класификация с невронни мрежи. FFNN Архитектури. MLPNN. PNN.
- ❑ Рекурентни невронни мрежи (RNN, LRPNN, GLRPNN, pcLRPNN)
- ❑ Класификатор със смес от Гаусови функции (GMM)
- ❑ Класификатор с опорни вектори (SVM)



Как се обучават?

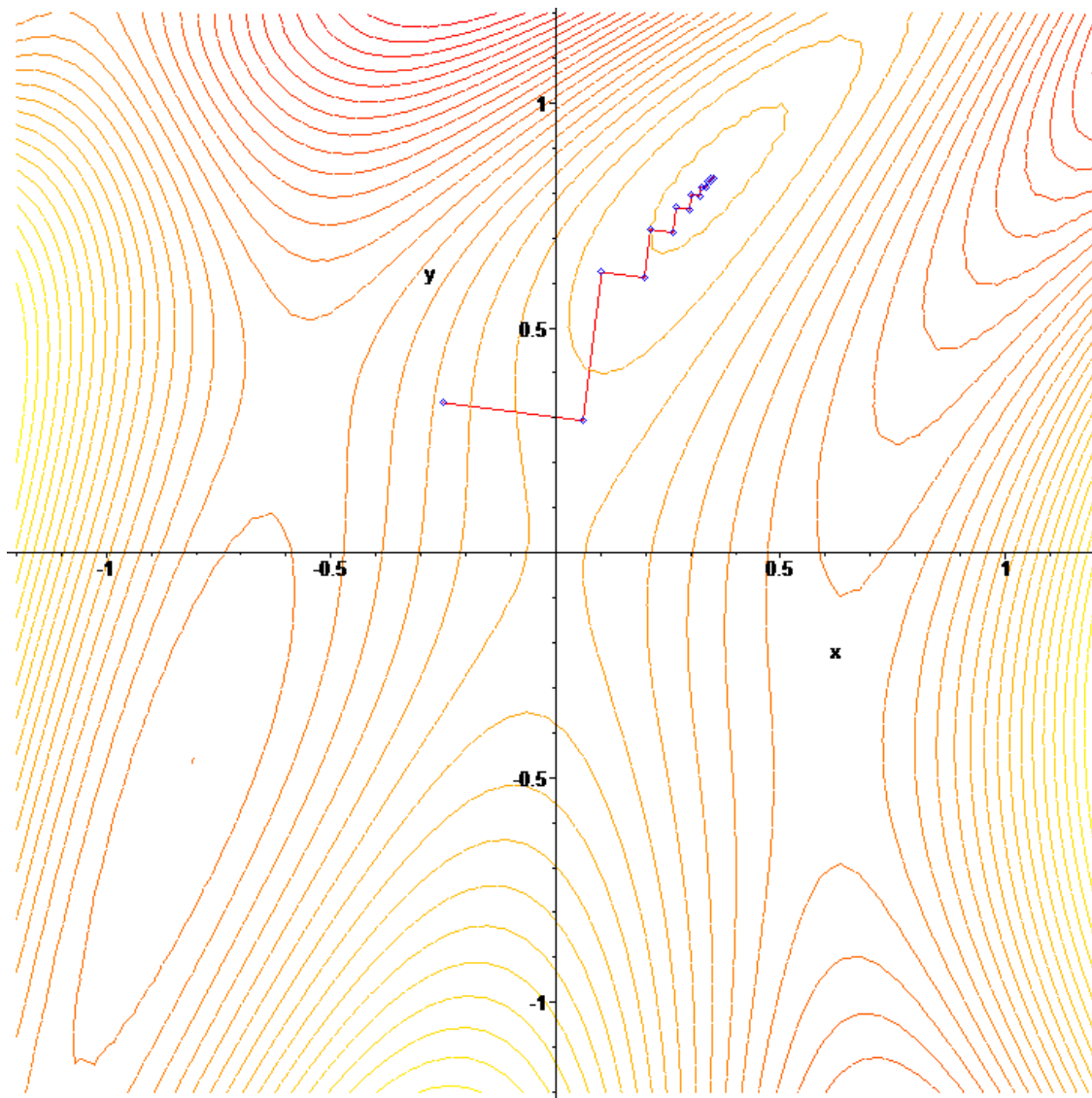
- ☐ Градиентни методи за оптимизация (GDO, GAO)
- ☐ Еволюционни и генетични методи за оптимизация (DE, PSO)

Методи за оптимизация

- Градиентни методи за оптимизация
 - Градиентно спускане (gradient-descent, GDO)
 - Градиентно изкачване (gradient-ascent, GAO)

- Еволюционни и генетични методи
 - диференциална еволюция (differential evolution, DE)
 - оптимизация с рояк от частици (PSO)

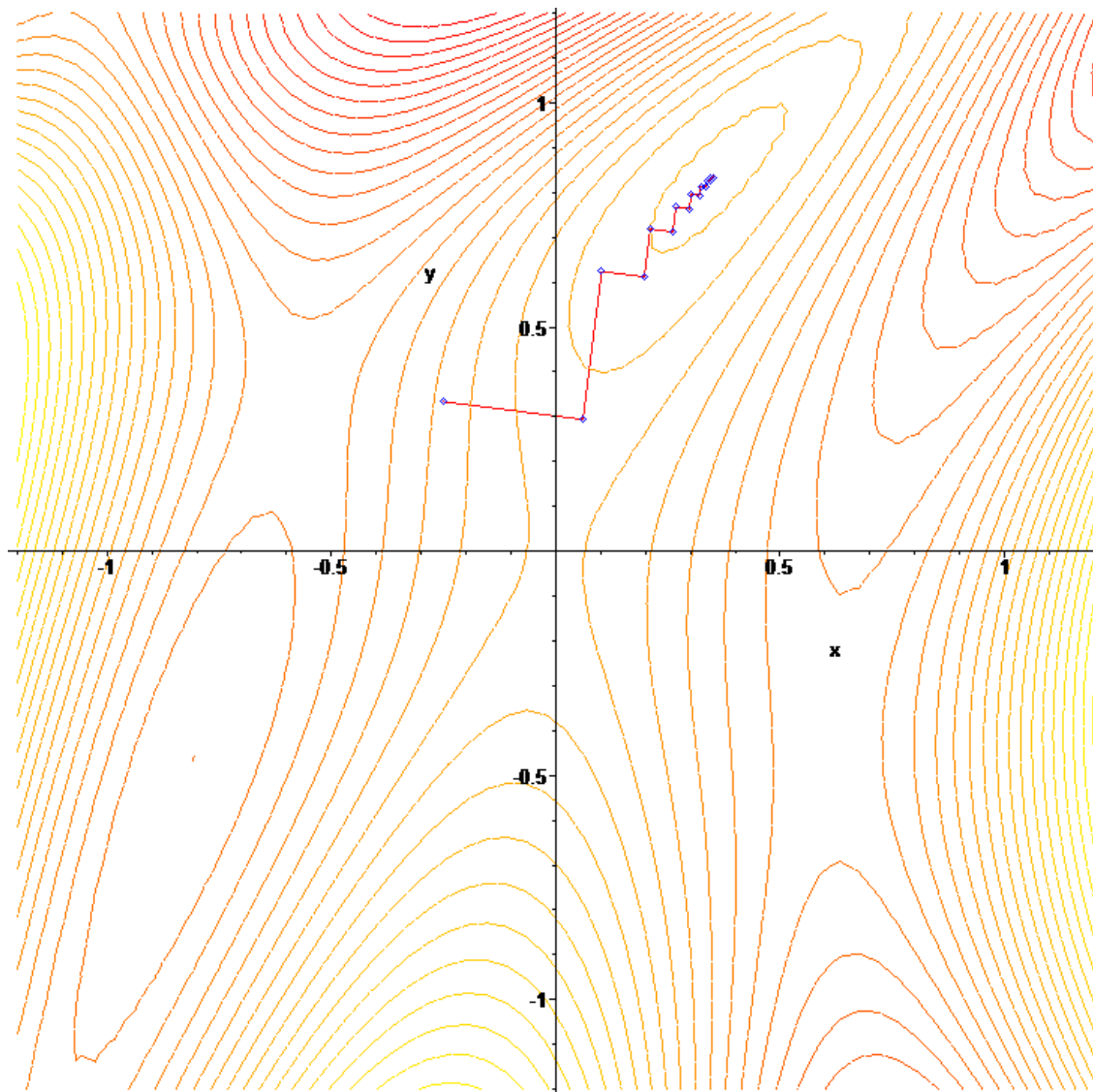
Градиентно спускане (gradient-descent)



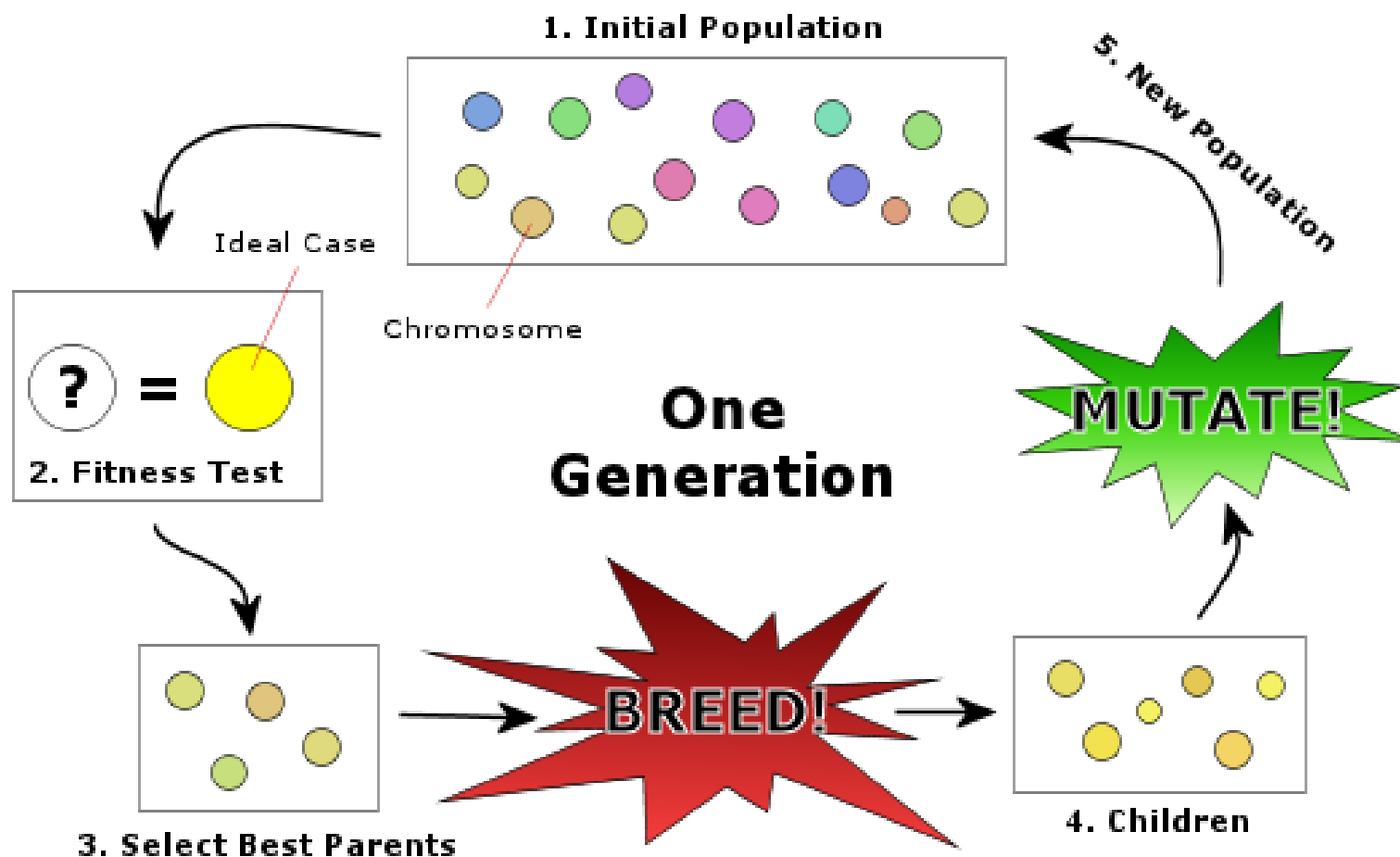
Градиентно спускане (gradient-descent)

```
For  $j = 0, \dots, d$   
     $w_j \leftarrow \text{rand}(-0.01, 0.01)$   
Repeat  
    For  $j = 0, \dots, d$   
         $\Delta w_j \leftarrow 0$   
    For  $t = 1, \dots, N$   
         $o \leftarrow 0$   
        For  $j = 0, \dots, d$   
             $o \leftarrow o + w_j x_j^t$   
         $y \leftarrow \text{sigmoid}(o)$   
         $\Delta w_j \leftarrow \Delta w_j + (r^t - y)x_j^t$   
    For  $j = 0, \dots, d$   
         $w_j \leftarrow w_j + \eta \Delta w_j$   
Until convergence
```

Градиентно спускане (gradient-descent)



Еволюционни методи




```
{  
    initialize population;  
    evaluate population;  
    while TerminationCriteriaNotSatisfied  
    {  
        select parents for reproduction;  
        perform recombination and mutation;  
        evaluate population;  
    }  
}
```

Еволюционни методи

Общи признаци

- ❑ Подбират се по-добрите индивиди
- ❑ Не винаги се избира най-добрия
- ❑ Не е задължително да се отхвърлят най-неудачните
- ❑ Няма гаранция
- ❑ Смес от алчна експлоатация и приключенско проучване

Differential Evolution метод

$$E(\mathbf{w}) = E_{lt}(\mathbf{w}) + G_{imp} E_{dif}(\mathbf{w}),$$

(1) Мутация,

(2) Рекомбинация,

(3) Селекция

$$\mathbf{v}_{g+1}^i = \mathbf{w}_g^{best} + \mu \left(\mathbf{w}_g^{r1} - \mathbf{w}_g^{r2} \right),$$

$$\mathbf{v}_{g+1}^i = \mathbf{w}_g^{best} + \mu \left(\mathbf{w}_g^{r1} - \mathbf{w}_g^{r2} \right) + \mu \left(\mathbf{w}_g^{r3} - \mathbf{w}_g^{r4} \right).$$

Particle Swarm Optimization

- Апроксимира колективното поведение в децентрализирани само-организиращи се системи
- Мотивирани от изучаването на колонии от организми и рояци от социални организми
- Колективната интелигентност е резултат от взаимодействието на организмите



Particle Swarm Optimization

- Предложен от Kennedy & Eberhart 1995
- Вдъхновен от социалното поведение на ята птици и пасажи риба
- Оптимизация основаваща се на интелигентността на рояк от агенти
- Недетерминистичен метод
- Оптимизация основана на популация от агенти
- Резултати подобни на генетичните алгоритми

Particle Swarm Optimization

- Рояк: набор от частици/агенти (S)
- Частица: кандидат решение на проблема
 - позиция, $X_i = (x_{i1}, x_{i2}, \dots, x_{in}) \in \mathbb{R}^n$
 - скорост, $V_i = (v_{i1}, v_{i2}, \dots, v_{in}) \in \mathbb{R}^n$
- Всяка частица запазва/използва
 - Най-добрата собствена позиция

$$P_i = (p_{i1}, p_{i2}, \dots, p_{in}) \in \mathbb{R}^n$$
$$pbest_i = f(P_i)$$

- Рояка запазва/помни глобално най-добрата позиция:

$$P_g \in \mathbb{R}^n$$
$$gbest = f(P_g)$$

PSO алгоритъм

Основен алгоритъм на PSO:

1. Инициализация на рояка в пространството на решенията
2. Оценяване на пригодността на всяка частица
3. Отчитане на индивидуалната и глобалната най-добра позиция
4. Обновяване на скоростта и позицията на всяка частица
5. Върни се към стъпка 2, и повтаряй до достигане на условието за прекратяване

PSO алгоритъм

1. Преизчисляване на скоростта

$$V_i^{t+1} = \underbrace{V_i^t}_{\text{Инерция}} + \underbrace{\varphi_1 \cdot r_1 (P_i - X_i^t)}_{\text{Познавателен компонент}} + \underbrace{\varphi_2 \cdot r_2 (P_g - X_i^t)}_{\text{Социален компонент}}$$

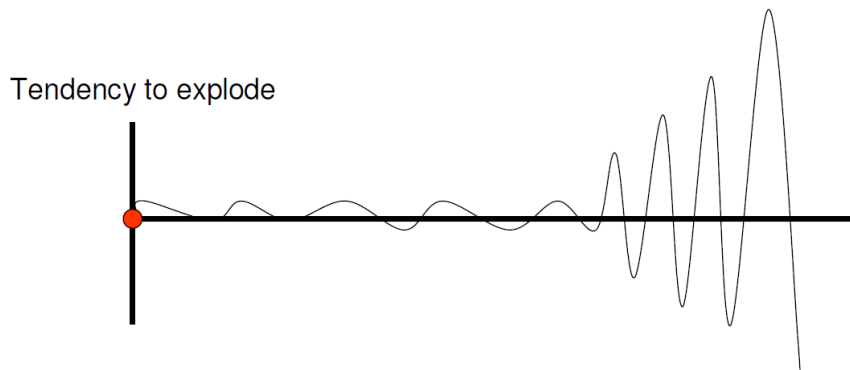
- където $r_1, r_2 \sim U(0,1)$
- φ_1, φ_2 : константи регулиращи ускорението

2. Преизчисляване на позицията

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

PSO алгоритъм

- Константи на ускорение
 - Малки стойности - ограничават предвижването на частиците
 - Големи стойности: тенденция да експлодират към безкрайност

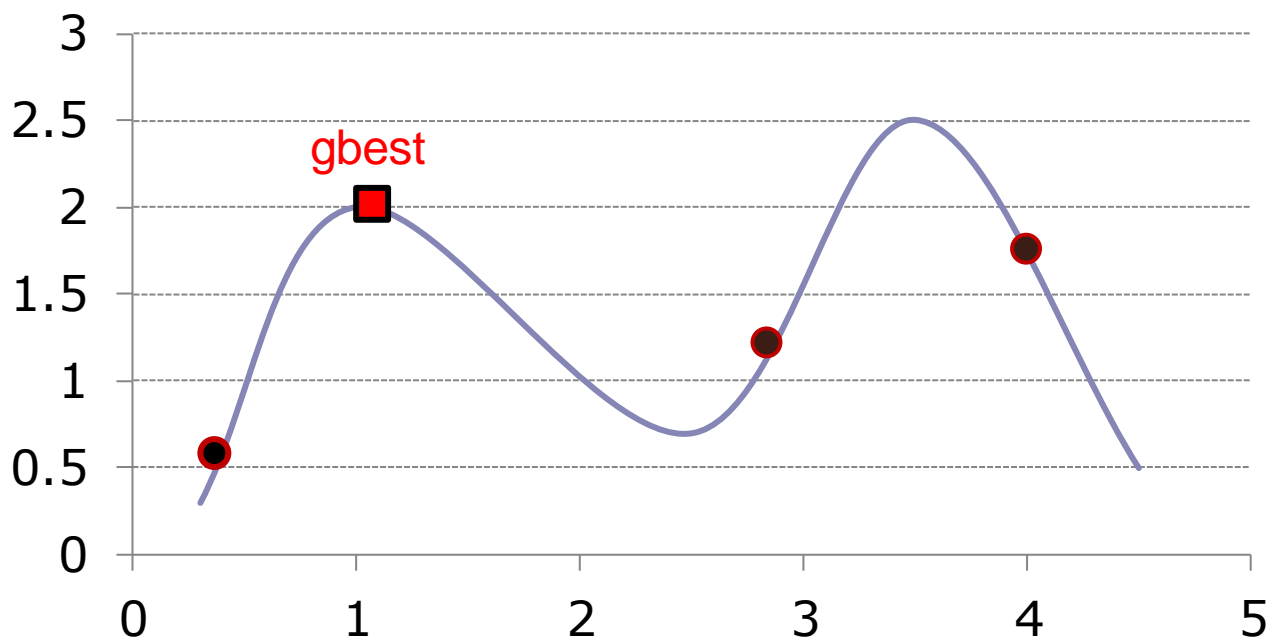


- Максимална скорост
 - Скоростта е стохастична променлива => траекторията е непредсказуема

Лесен 1D пример



Инициализация на рояка и оценяване на пригодността на частиците ($t=0$)

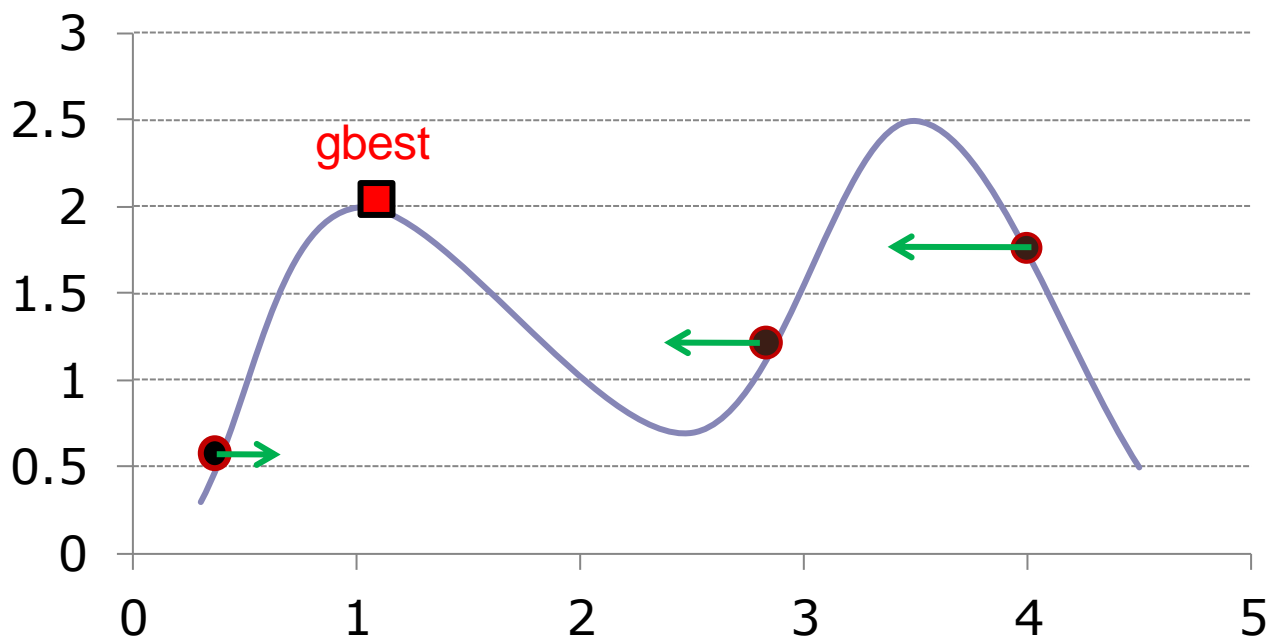


Лесен 1D пример



Обновяване на скоростта и позицията (t=1)

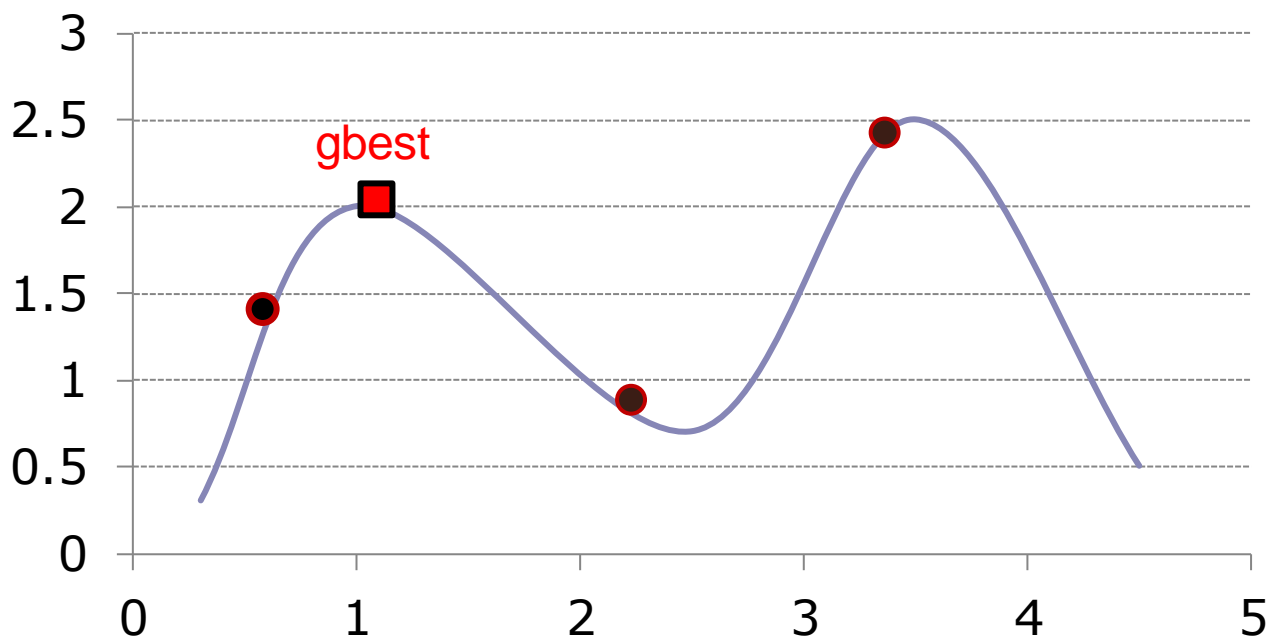
$$V_i^{t+1} = V_i^t + \varphi_1 \cdot r_1 (P_i - X_i^t) + \varphi_2 \cdot r_2 (P_g - X_i^t)$$



Лесен 1D пример



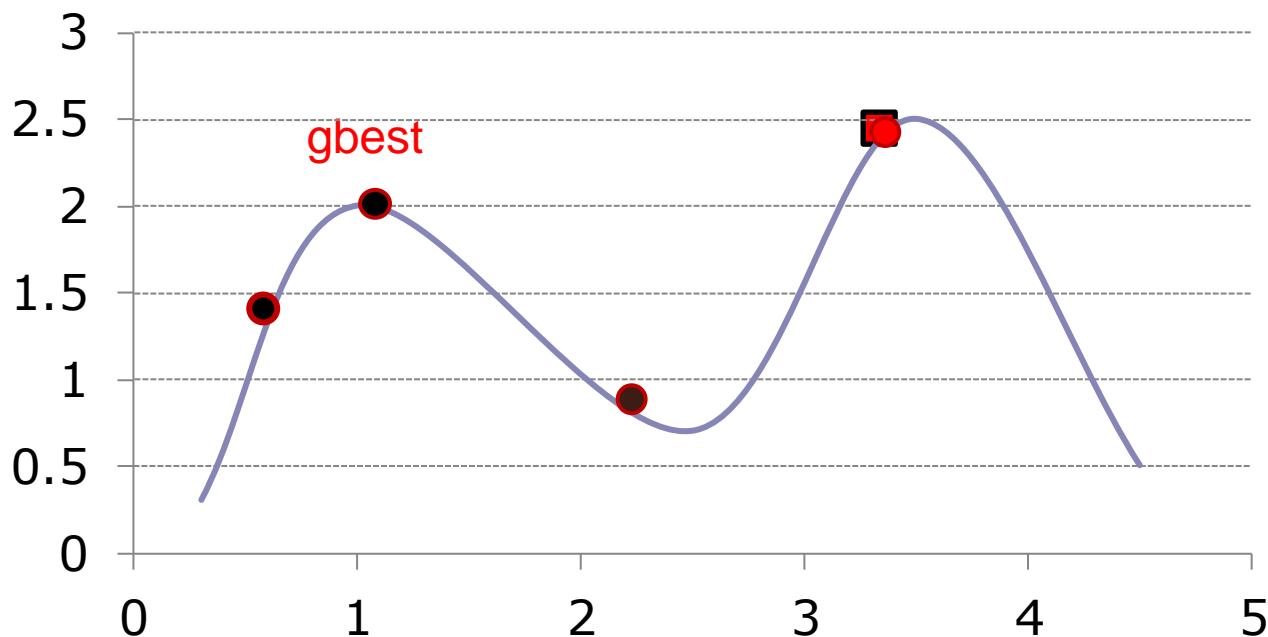
Оценяване на пригодността.
Обновяване на индивидуалната и глобално
най-добрата стойност ($t=2$)



Лесен 1D пример



Оценяване на пригодността.
Обновяване на индивидуалната и глобално
най-добрата стойност ($t=2$)

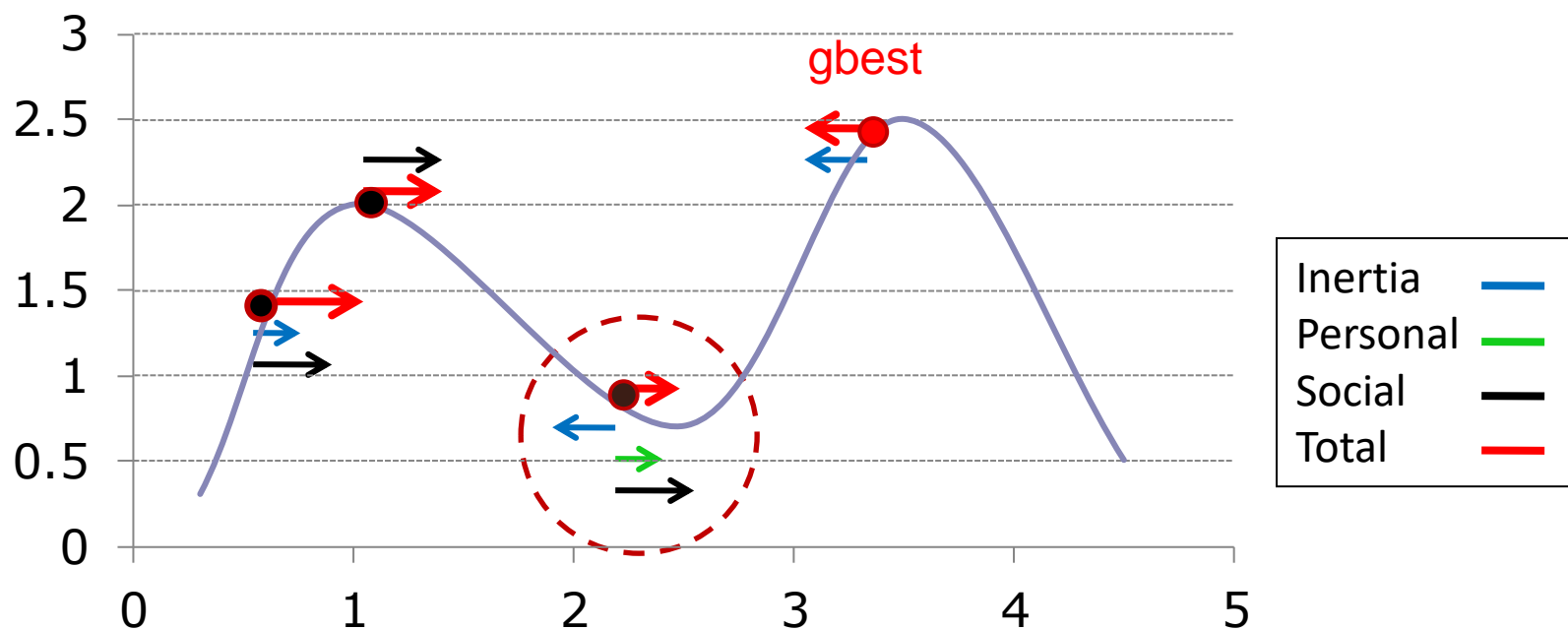


Лесен 1D пример



Обновяване на скоростта и позицията (t=2)

$$V_i^{t+1} = V_i^t + \varphi_1 \cdot r_1 (P_i - X_i^t) + \varphi_2 \cdot r_2 (P_g - X_i^t)$$



Подобряване на сходимостта

- Коефициент на инерция:

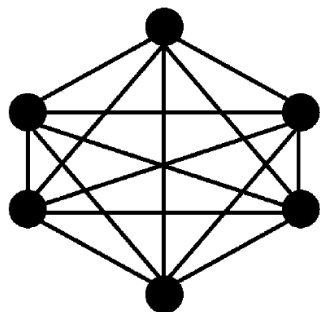
$$V_i^{t+1} = \underset{\downarrow}{w} V_i^t + \varphi_1 \cdot r_1 (P_i - X_i^t) + \varphi_2 \cdot r_2 (P_g - X_i^t)$$

- Мащабиране на предишната ст-ст на скоростта
- Управлява поведението при претърсване
 - Големи стойности → изучаване на пространството
 - Малки стойности → експлоатация (разработване)

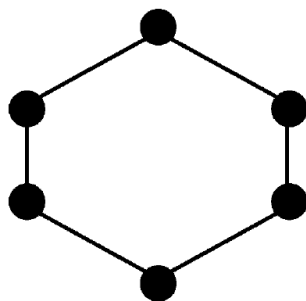
Подобряване на сходимостта

■ Два вида съседство:

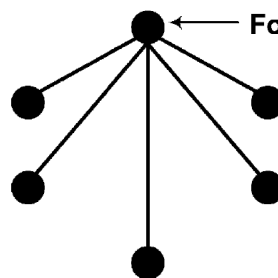
- Глобално най-добър (*gbest*): напълно свързана мрежа
- Локално най-добър (*lbest*): съгласно някъква ТОПОЛОГИЯ



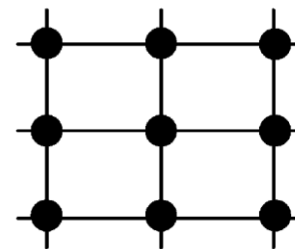
gbest



пръстен



звезда



Von Neumann

lbest

Подобряване на сходимостта

Lbest vs. Gbest

- **Gbest** предлага по-бърза сходимост но може да попадне в локален оптимум.
- **Lbest** е води до по-бавна сходимост но шансовете за откриване на оптимално решение.
- Най-добрата топология в общия случай зависи от типа на задачата.
- **Напълно информиран PSO алгоритъм (FIPS):**
 - Всяка частица е повлияна от успеха на всички свои съседни.

$$V_i^{t+1} = \chi \cdot \left[V_i^t + \sum_{p_k \in N_i} \varphi_k r_k (P_k^t - X_i^t) \right]$$

Particle Swarm Optimization

```
For each particle
  Initialize particle
END
```

```
Do
  For each particle
    Calculate fitness value
    If the fitness value is better than its personal best
      set current value as the new pBest
  End
```

```
  Choose the particle with the best fitness value of all as gBest
```

```
  For each particle
    Calculate particle velocity according equation (a)
    Update particle position according equation (b)
```

```
  End
```

```
While maximum iterations or minimum error criteria is not attained
```

$$(a) \ v[] = c0*v[] + c1*rand()*(pbest[]-present[]) + c2*rand()*(gbest[]-present[])$$

$$(b) \ present[] = present[] + v[]$$