Inteligência Artificial para Robótica Móvel

Estratégias Evolutivas

Professor: Marcos Maximo

Roteiro

- Motivação.
- Revisão de Probabilidade.
- Estratégias Evolutivas.
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES).
- Estudos de Caso.

Motivação

Estratégias Evolutivas

- Inglês: *Evolution Strategies* (ES).
- Métodos baseados em população trabalham com um conjunto de candidatos à solução.
- Algoritmos Genéticos introduziram a ideia de selecionar e combinar as melhores soluções.
- Evolution Strategy (ES) também utiliza conceitos de evolução: mutação e seleção.
- ES amostra pontos de uma distribuição de probabilidade (mutação).
- Os parâmetros da distribuição são evoluídos a partir das melhores amostras (seleção).

Estratégias Evolutivas

- Uma das variantes de ES é a Covariance Matrix Adaptation Evolution Strategy (CMA-ES).
- CMA-ES trabalha com distribuições gaussianas.
- Como o nome indica, a inovação do método consiste em adaptar a matriz de covariância da distribuição.
- É um dos melhores algoritmos de otimização caixa preta conhecidos.
- **Muito** popular no futebol de robôs.
- Rivaliza com métodos modernos de Aprendizado por Reforço Profundo.
- Apesar de caixa preta, tem forte embasamento matemático.

Revisão de Probabilidade

Definições de Probabilidade

- Experimento aleatório: resultado obtido é desconhecido e imprevisível.
- Espaço amostral (S): conjunto de todos os resultados possíveis de um evento aleatório.
- Evento ($A \subseteq S$): subconjunto do espaço amostal.
- Lei (função) de probabilidade ($P(A)$): função que codifica a crença de um evento acontecer.

Axiomas da Teoria de Probabilidade

- $P(A) \geq 0$
- $P(S) = 1$
- $A_1, A_2, \ldots, A_n$ eventos tais que $A_i \cap A_j = \emptyset$:

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i)$$

Variáveis Aleatórias Discretas

- Variável aleatória (VA) X assume valores discretos.
- $X \in \{x_1, x_2, \dots, x_n\}$.
- $P(X = x_i)$ ou $P(x_i)$ é a probabilidade de X valer x_i.

Variáveis Aleatórias Contínuas

- Variável aleatório X assume valores em um contínuo.
- Probabilidade de X assumir um único valor é nula.
- Trabalha-se com o conceito de função densidade de probabilidade: $p(X)$ ou $p(X = x)$.
- Probabilidade é a integral:

$$P(a \leq x \leq b) = \int_{a}^{b} p(x)dx$$

- Se integrar no domínio inteiro?

Vetores Aleatórios

- Pode-se coletar várias variáveis aleatórias em um vetor:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$$

Esperança

- Resultado médio se repetir o experimento infinitas vezes.
- Matematicamente:

$$E[X] = \mu_x = \sum_x xP(x)$$

$$E[X] = \mu_x = \int_x xp(x)dx$$

Esperança Amostral

- No caso de se ter N amostras, pode-se aproximar a esperança por:

$$\bar{\mu}_x = \frac{1}{N}\sum_i x_i$$

Variância

- “Dispersão” dos resultados.
- Matematicamente:

$$Var[X] = \sigma_x^2 = E[(X - E[X])^2]$$

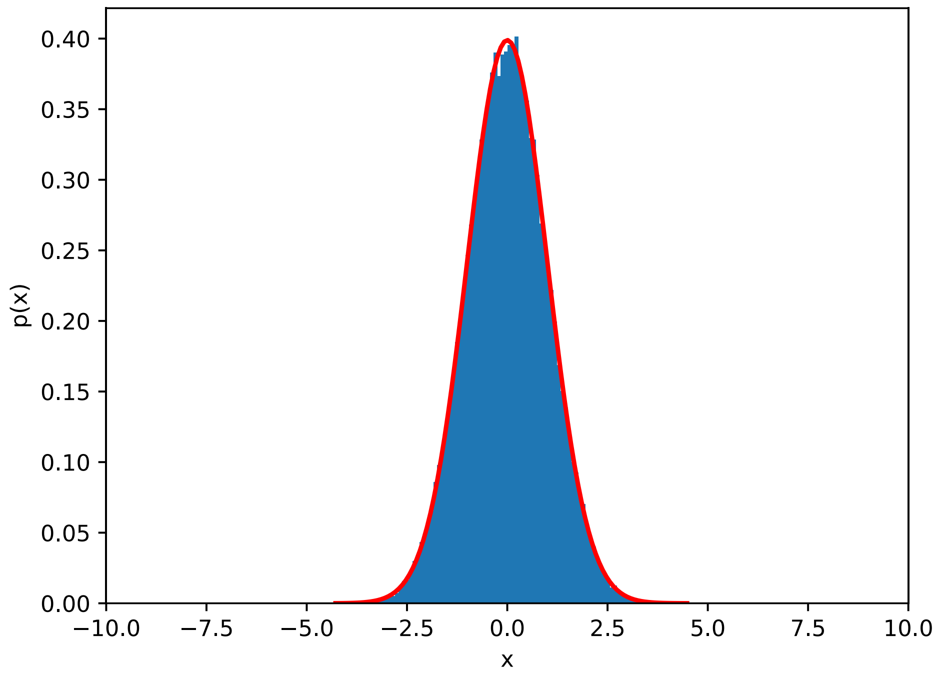
$$\sigma_x^2 = \sum_x (x - \mu_x)^2 P(x)$$

$$\sigma_x^2 = \int_x (x - \mu_x)^2 p(x) dx$$

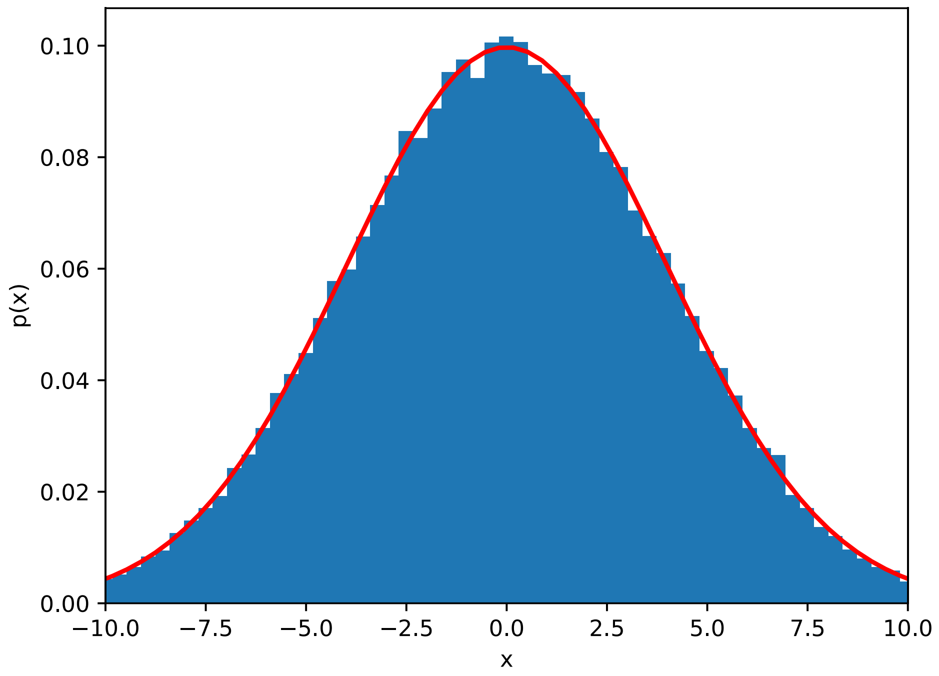
- Desvio padrão: σ_x.

Intuição de Variância

Menos disperso



Mais disperso



Matriz de Covariância

- Covariância para duas variáveis:

$$Cov[X,Y] = E[(X - E[X])(Y - E[Y])]$$

- No caso de vetor, tem-se matriz de covariância:

$$\mathbf{X} = [X_1, X_2, \dots, X_n]^T$$

$$Cov[\mathbf{X}] = \mathbf{\Sigma} = E[(\mathbf{X} - E[\mathbf{X}])(\mathbf{X} - E[\mathbf{X}])^T]$$

$$Cov[\mathbf{X}] = \begin{bmatrix} Cov(X_1) & Cov(X_1, X_2) & \dots & Cov(X_1, X_n) \\ Cov(X_2, X_1) & Cov(X_2) & \dots & Cov(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n, X_1) & Cov(X_n, X_2) & \dots & Cov(X_n) \end{bmatrix}$$

Matriz de Covariância Amostral

- A estimativa não-enviesada para a matriz de covariância a partir de N amostras é:

$$\overline{Cov}[\mathbf{X}] = \frac{1}{N-1}\sum_i (\mathbf{X}_i - \bar{\boldsymbol{\mu}}_x)(\mathbf{X}_i - \bar{\boldsymbol{\mu}}_x)^T$$

em que $\bar{\mu}_x$ é a média amostral.

- Porém, se a média real é conhecida, a estimativa não-enviesada é:

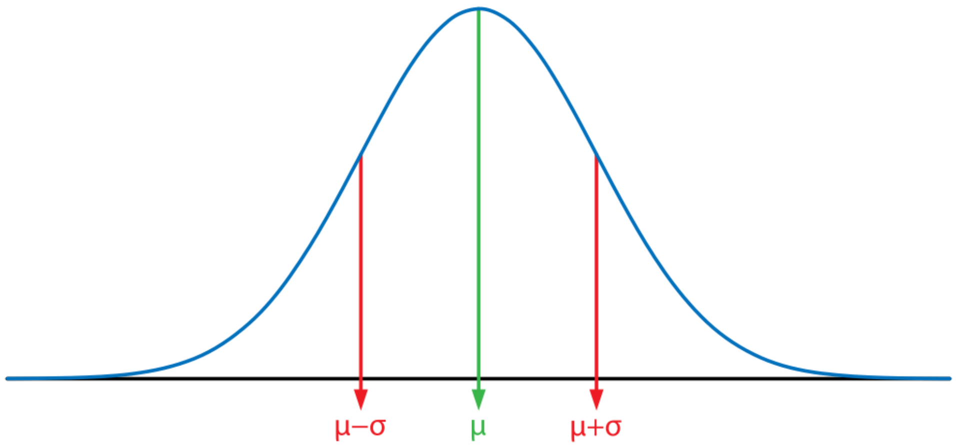
$$Cov[\mathbf{X}] = \frac{1}{N}\sum_i (\mathbf{X}_i - \boldsymbol{\mu}_x)(\mathbf{X}_i - \boldsymbol{\mu}_x)^T$$

Distribuição Gaussiana

- Para uma variável:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) = N(\mu, \sigma^2)$$

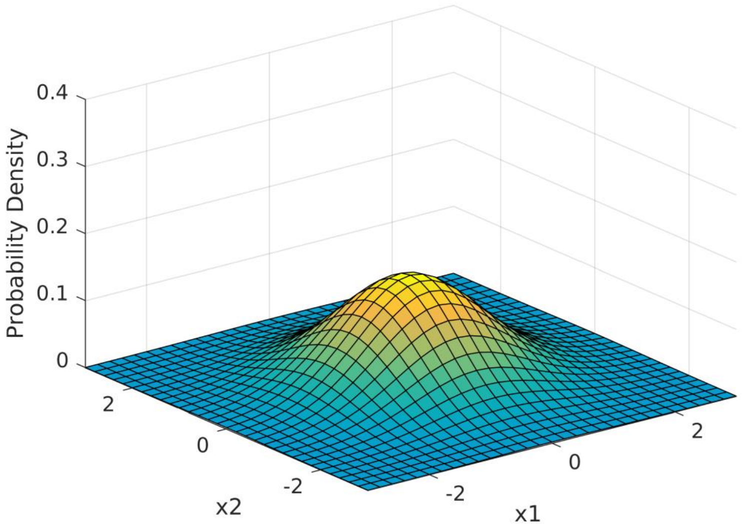
- Também chamada de distribuição normal.



Distribuição Gaussiana

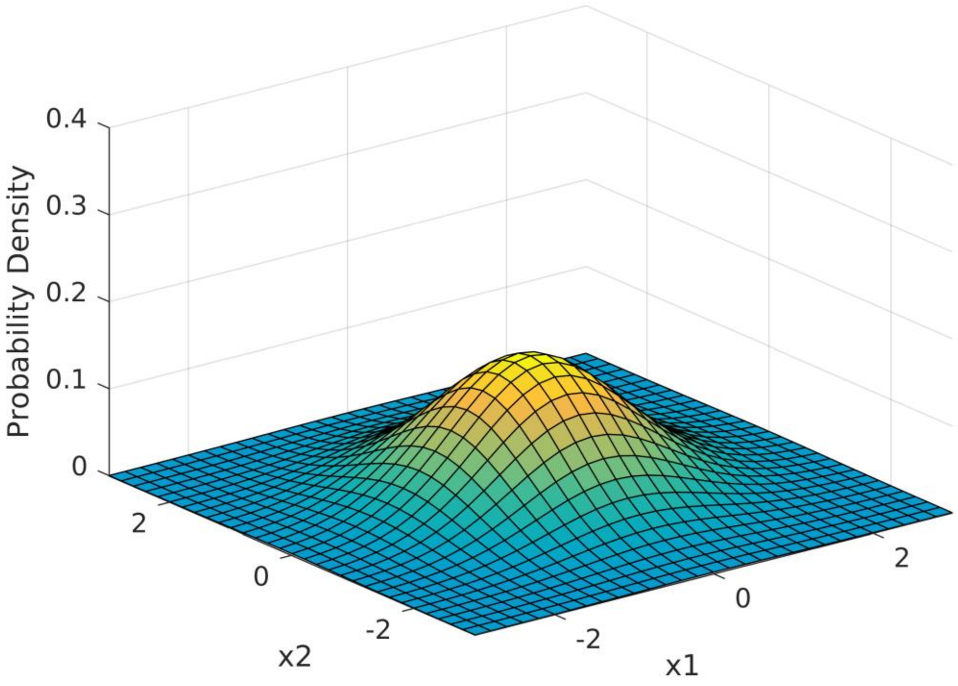
- Para vetor:

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

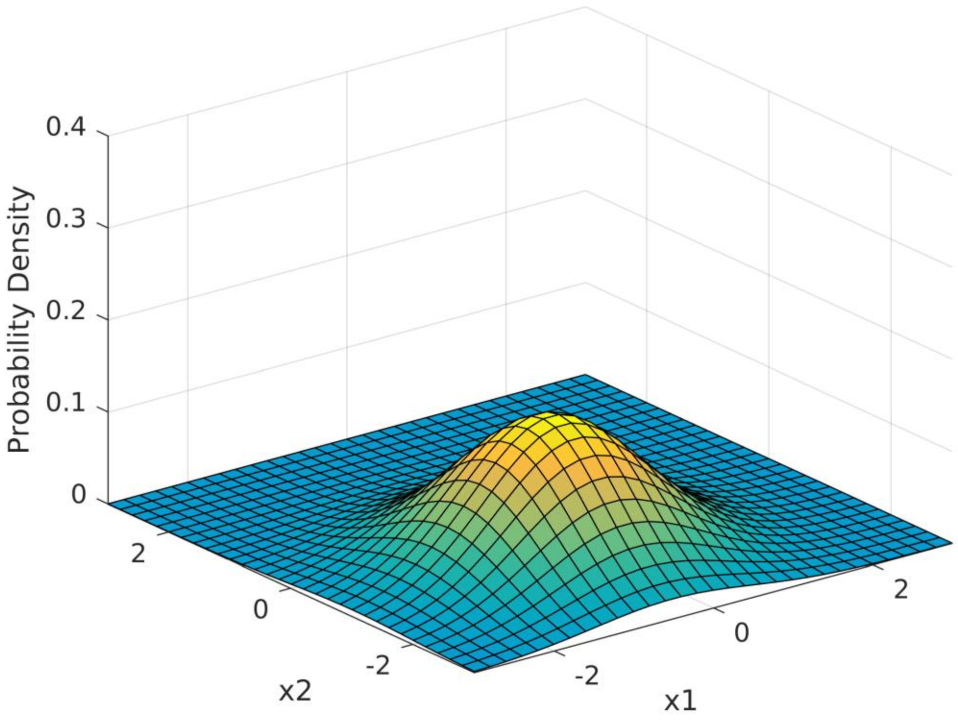


Intuição

$\mu = [0\ 0]^T, \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

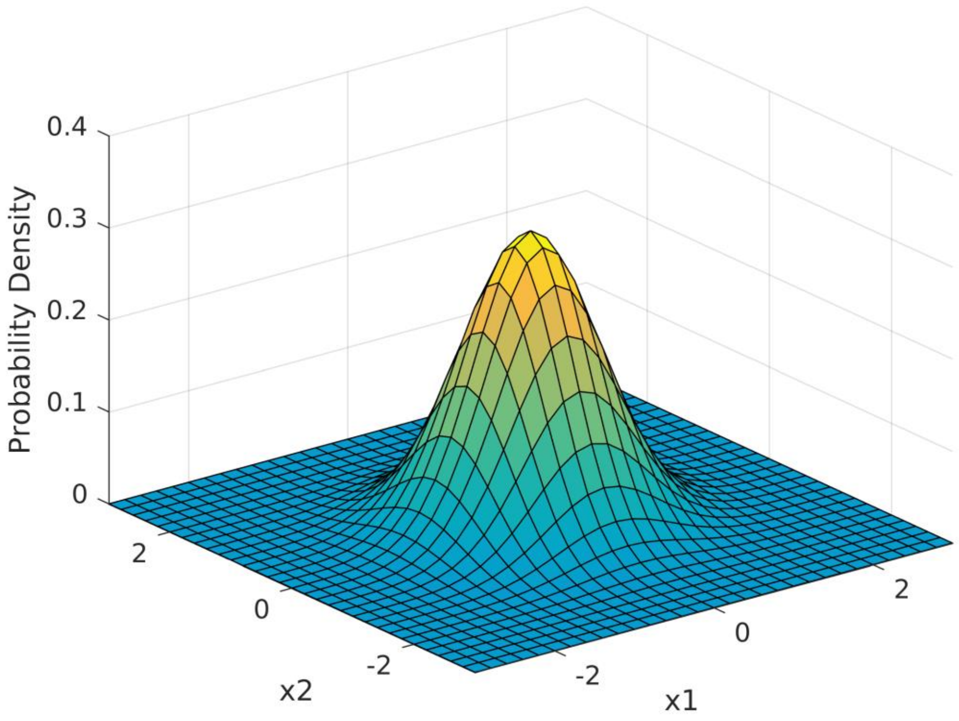


$\mu = [-0{,}5\ \ -1]^T, \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

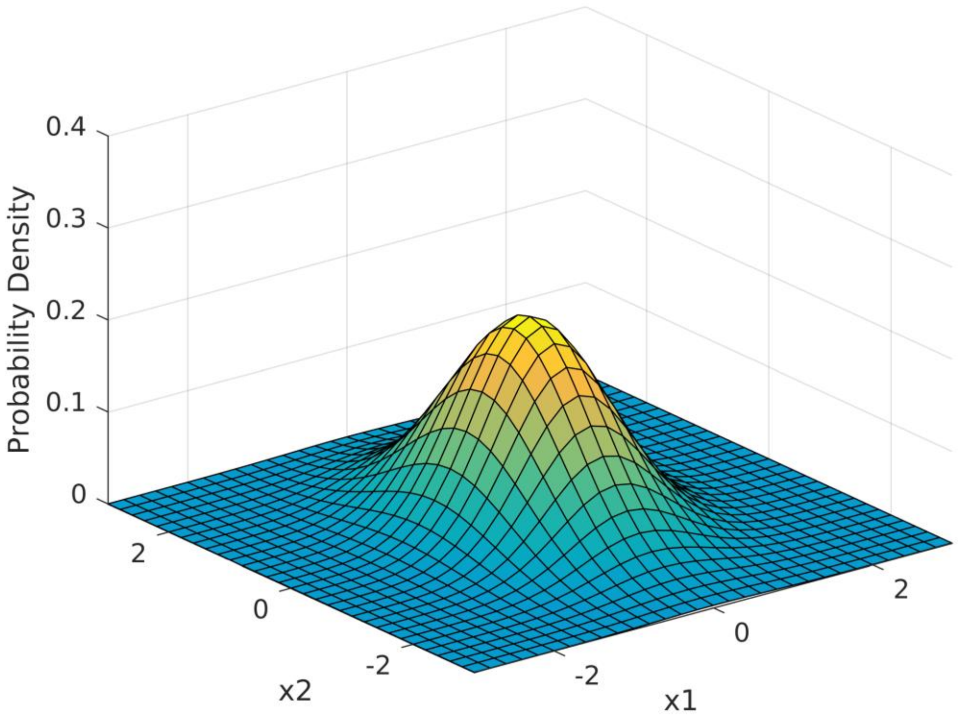


Intuição

$$\mu = [0\ 0]^T, \Sigma = \begin{bmatrix} 0{,}5 & 0 \\ 0 & 0{,}5 \end{bmatrix}$$

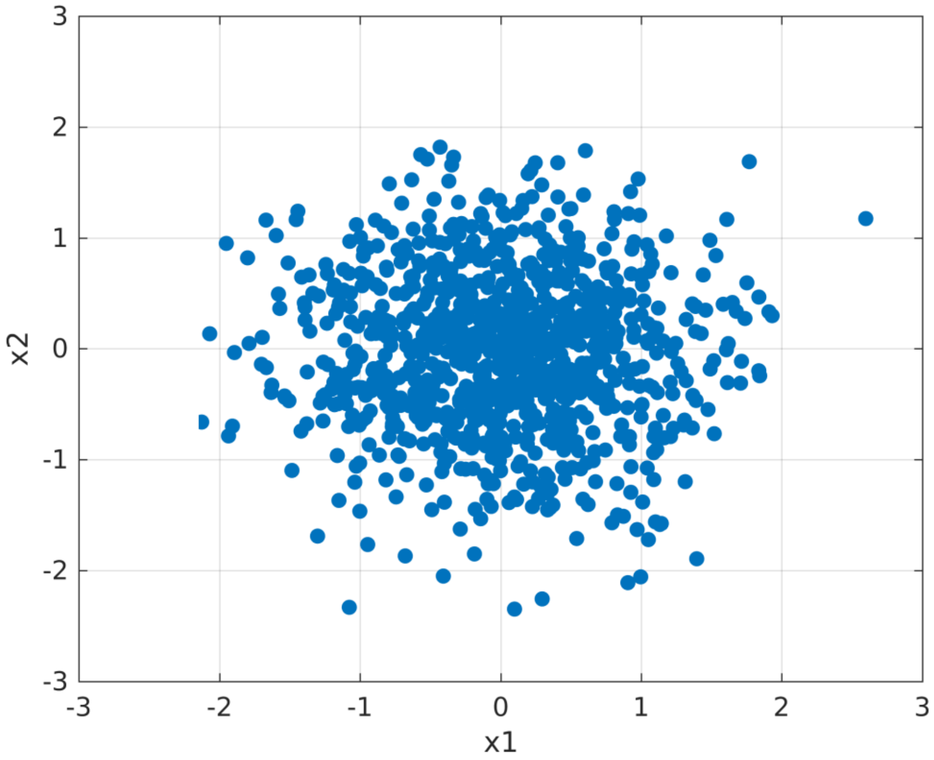


$$\mu = [0\ 0]^T, \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0{,}5 \end{bmatrix}$$

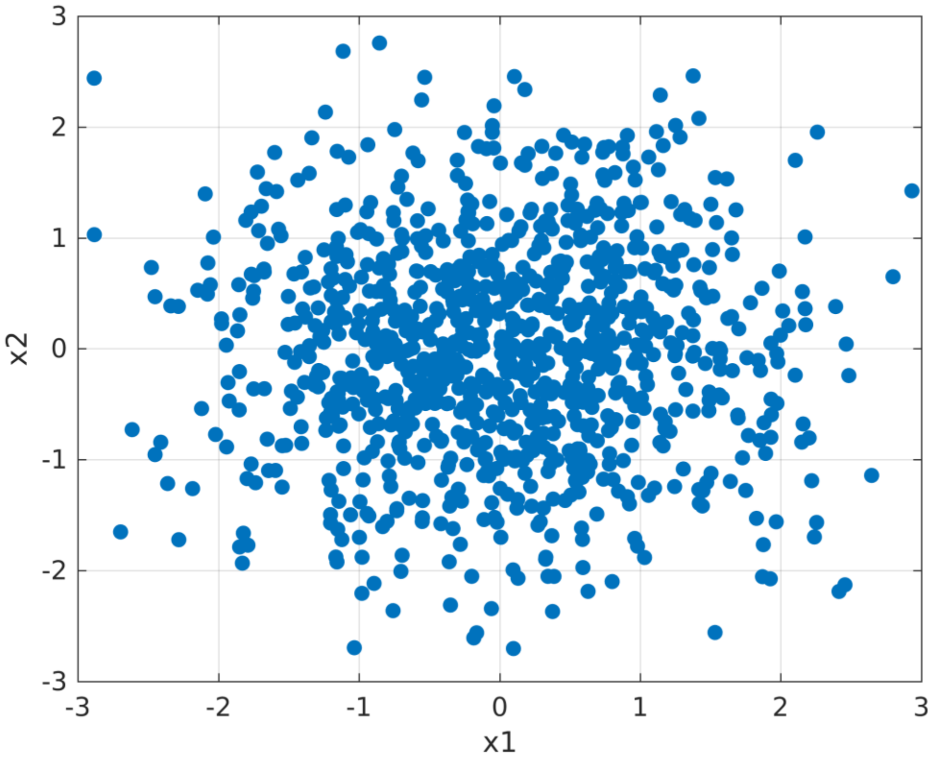


Intuição

$$\mu = [0\ 0]^T, \Sigma = \begin{bmatrix} 0{,}5 & 0 \\ 0 & 0{,}5 \end{bmatrix}$$

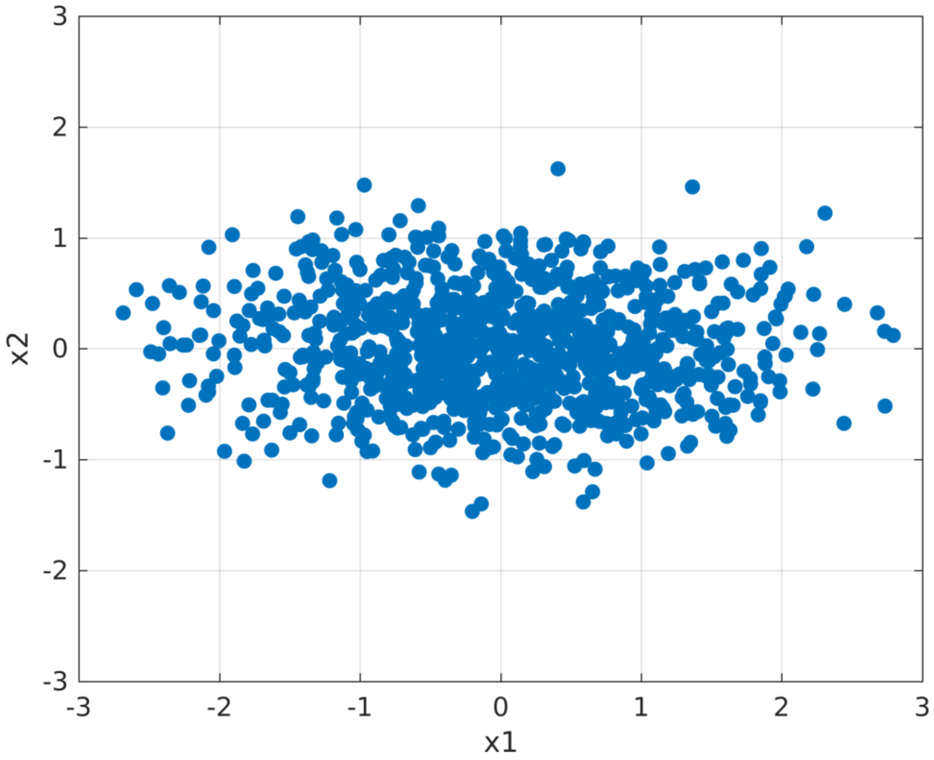


$$\mu = [0\ 0]^T, \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

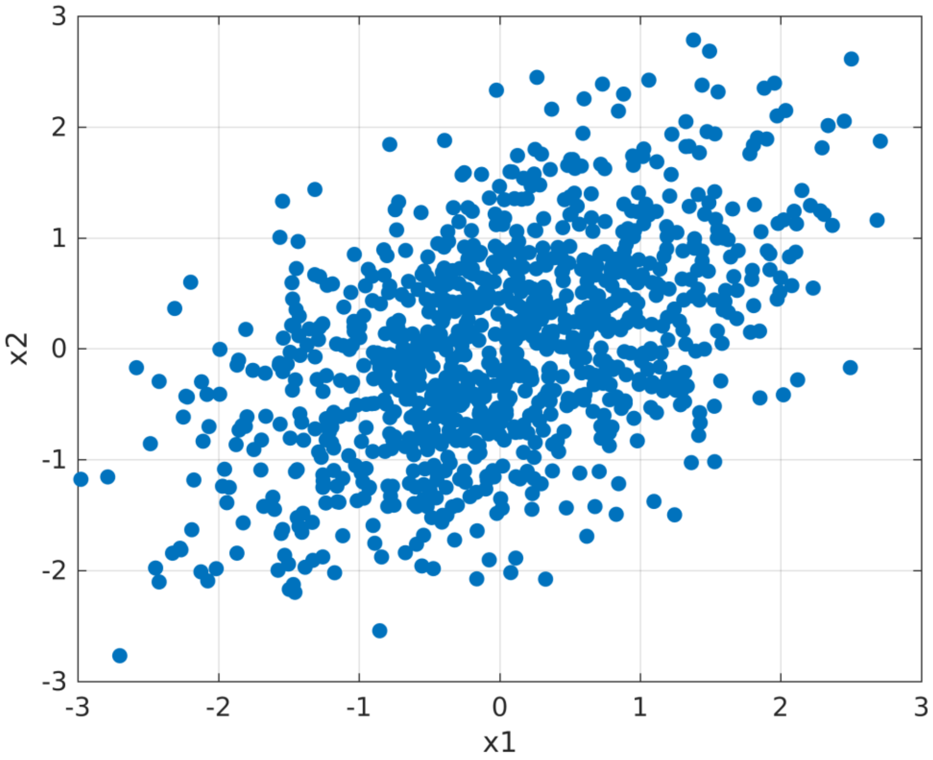


Intuição

$$\mu = [0\ 0]^T, \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0{,}25 \end{bmatrix}$$



$$\mu = [0\ 0]^T, \Sigma = \begin{bmatrix} 1 & 0{,}5 \\ 0{,}5 & 1 \end{bmatrix}$$



Demonstração NumPy

```
import numpy as np
import matplotlib.pyplot as plt

num_samples = 500
mu = np.array([1.0, 0.0])
C = np.array([[1.0, 0.5], [0.5, 2.0]])
X = np.random.multivariate_normal(mu, C, num_samples)
plt.figure()
plt.plot(X[:, 0], X[:, 1], '.')
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```

Estratégias Evolutivas

Evolution Strategy

- Amostra candidatos a partir de distribuição de probabilidade (mutação).
- Comumente, usa-se distribuição gaussiana multivariada.
- Então, evolui-se parâmetros da distribuição (e.g. média e covariância) a partir das melhores amostras (seleção).
- Repete-se até critério de parada ser satisfeito.

Evolution Strategy

- λ: tamanho da população (hiperparâmetro).
- μ: número de melhores amostras escolhidas (hiperparâmetro).
- Trabalha-se com classificação (*rank*) das amostras ao invés do valor de *fitness:*

$$J\left(\boldsymbol{x}_{1:\lambda}^{(g)}\right) \leq J\left(\boldsymbol{x}_{2:\lambda}^{(g)}\right) \leq \cdots \leq J\left(\boldsymbol{x}_{\lambda:\lambda}^{(g)}\right)$$

- Com isso, evita-se depender do valor da função de *fitness* em si.
- Notação $\boldsymbol{x}_{i:\lambda}^{(g)}$ indica que esse é o i-ésimo melhor ponto dentre λ amostras da geração g.

Evolution Strategy

- Considere uma estratégia de evolução simples, em que apenas a média da distribuição é evoluída:

$$\boldsymbol{m}^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu} \boldsymbol{x}_{i:\lambda}^{(g+1)}$$

- Se $\mu = 1$, então a distribuição da próxima geração é centrada no melhor ponto.
- Por enquanto, a matriz de covariância permanece constante:

$$\boldsymbol{C}^{(g+1)} = \boldsymbol{C}^{(g)}$$

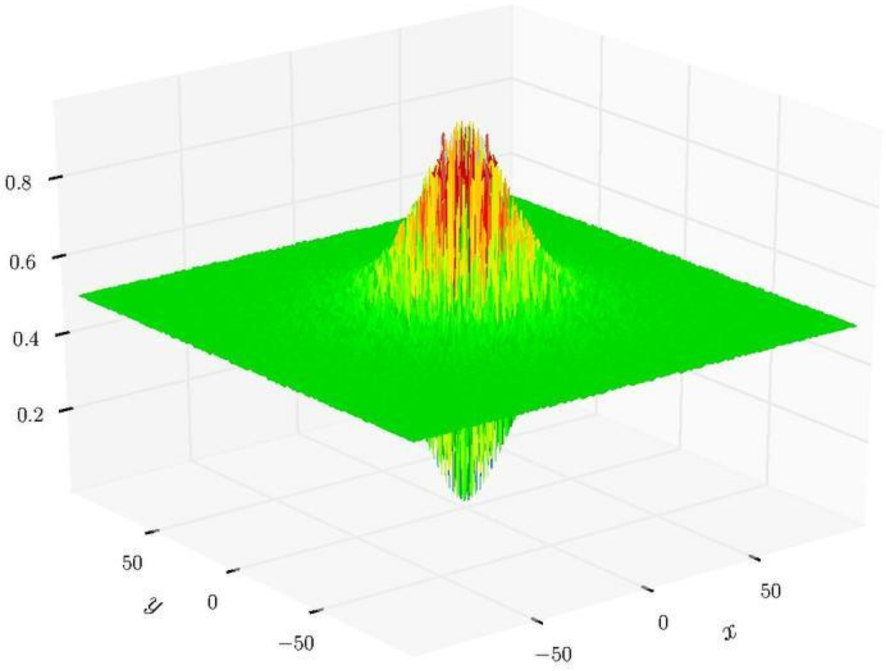
Evolution Strategy

```
# Assuming minimization
def evolution_strategy(J, m0, C0, hyperparams):
     mu, lambda = unwrap_hyperparams(hyperparams)
     m, C = m0, C0
     while not check_stopping_condition():
          population = multivariate_normal(m, C, lambda)
          population = sort_ascending(population, J)
          parents = population[0:mu]
          m = mean(parents)
     return population[0]
```

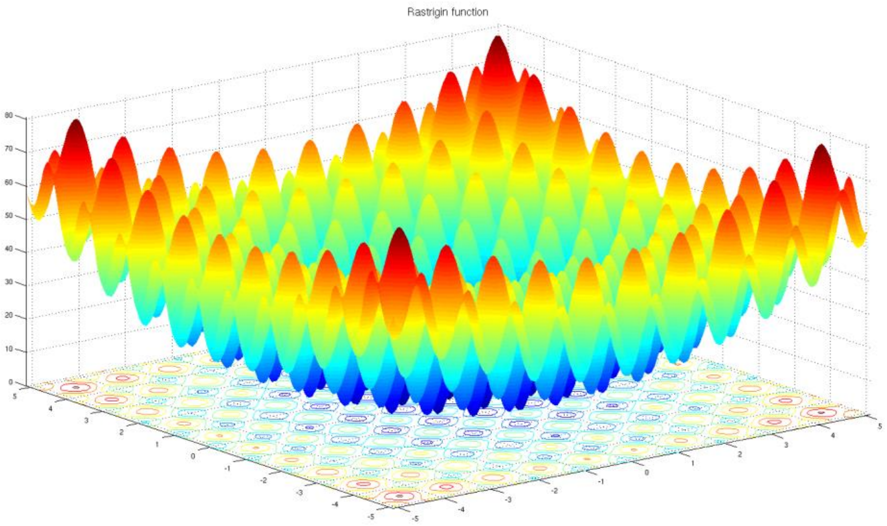
Funções de Teste para Otimização

$$f(x,y) = 0{,}5 + \frac{\sin^2(x^2 - y^2) - 0.5}{[1 + 0{,}001(x^2 + y^2)]^2}$$

$$f(x,y) = An + \sum_{i=1}^{n}\left[x_i^2 - Acos(2\pi x_i)\right]$$



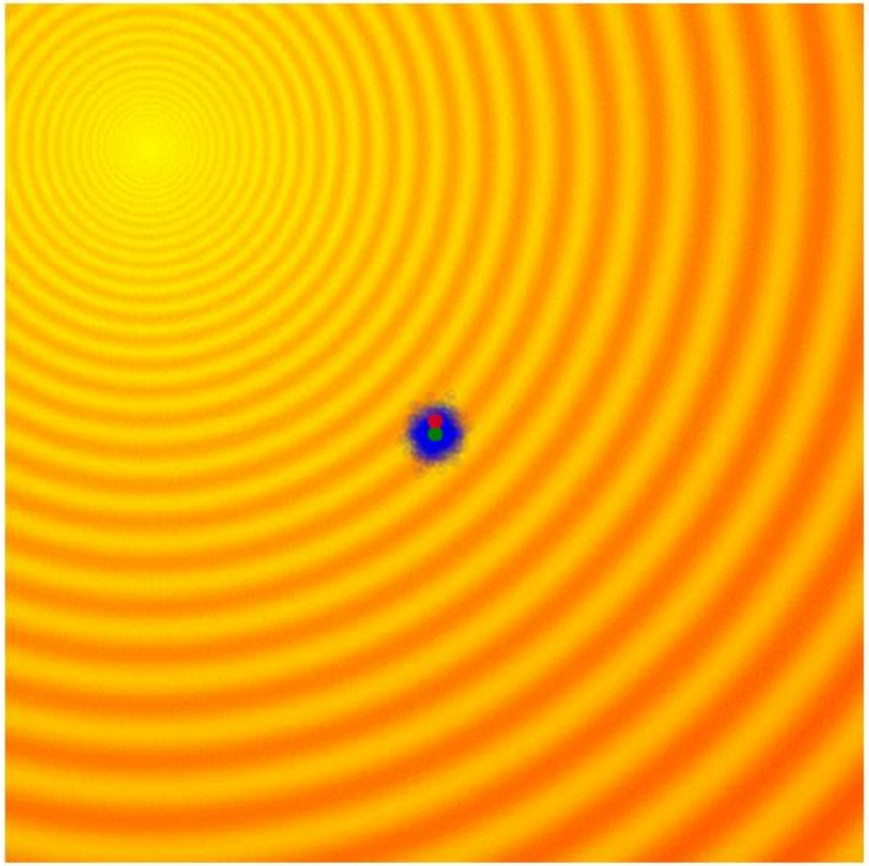
Schaffer



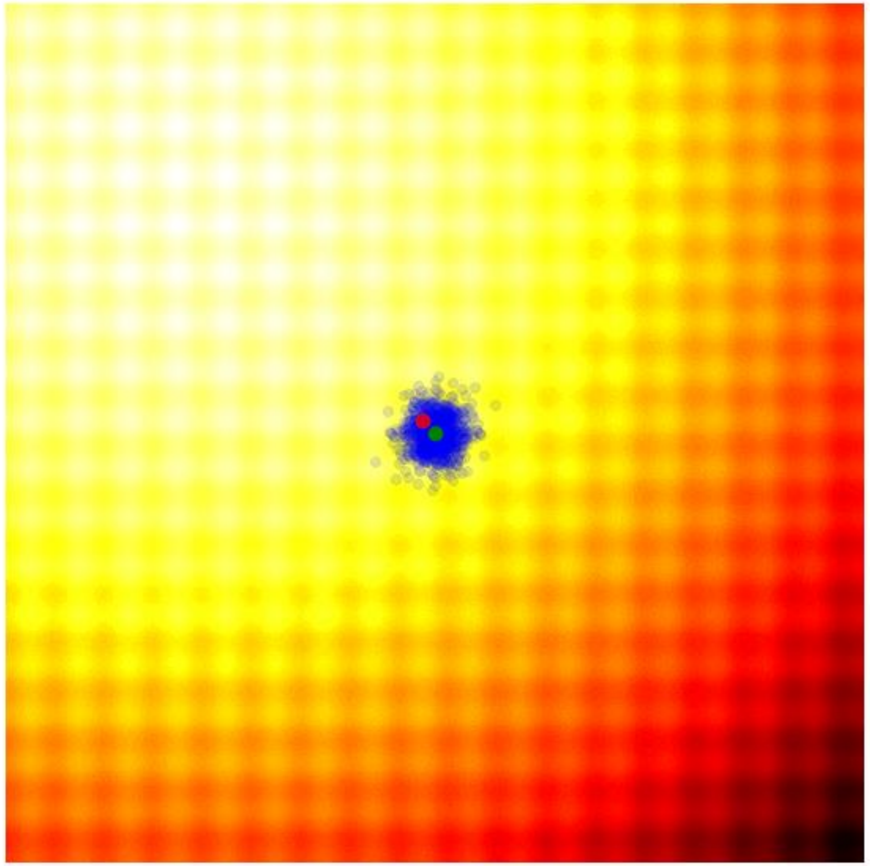
Rastrigin

Evolution Strategy

Schaffer



Rastrigin



Fonte: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/

Adaptação da Matriz de Covariância

- Uma ideia simples para evoluir a matriz de covariância é usar a covariância amostral das μ melhores amostras:

$$\boldsymbol{C}^{(g+1)} = \frac{1}{\mu - 1}\sum_{i=1}^{\mu}\left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g+1)}\right)\left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g+1)}\right)^{T}$$

em que:

$$\boldsymbol{m}^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{x}_{i:\lambda}^{(g+1)}$$

Adaptação da Matriz de Covariância

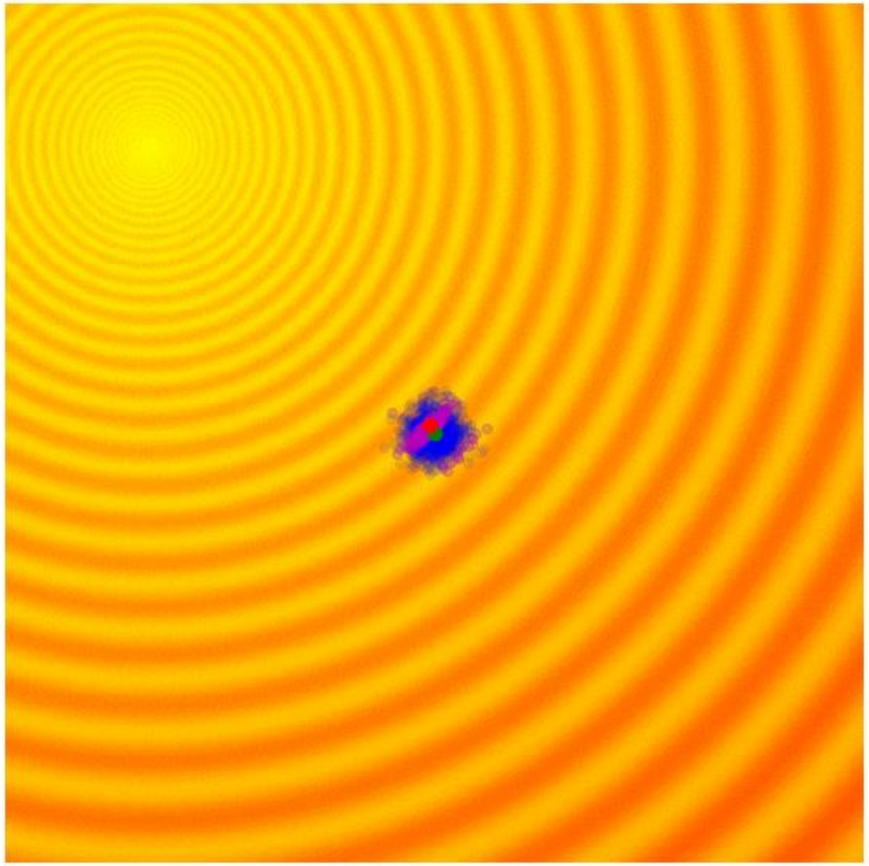
- Alternativamente, pode-se usar a média real da distribuição:

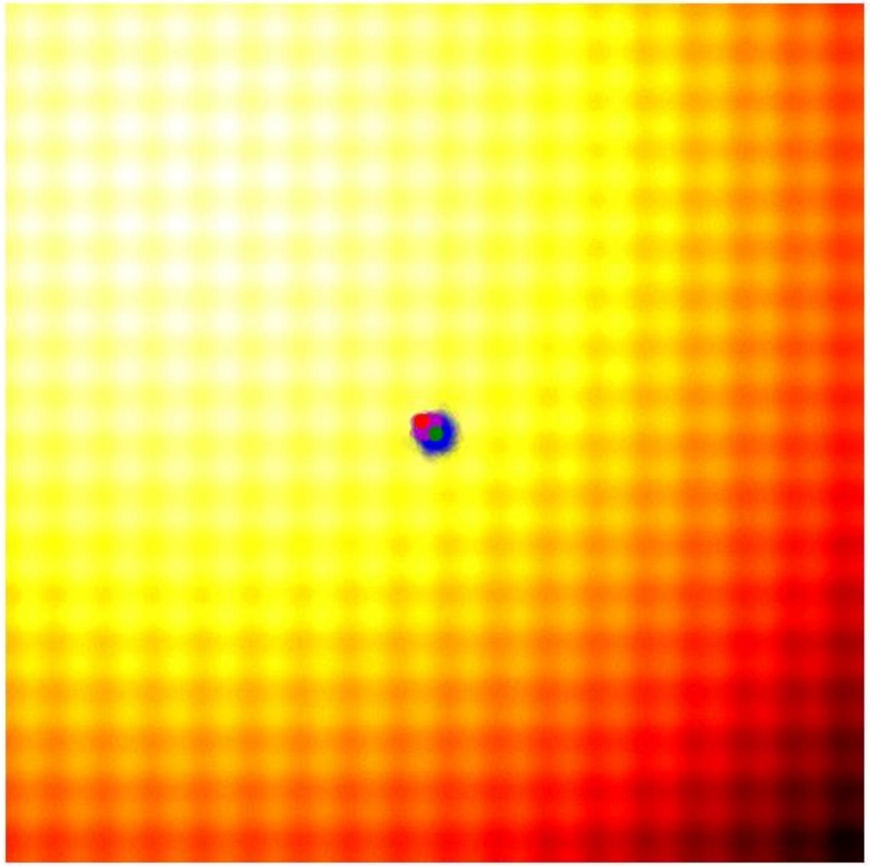
$$\boldsymbol{C}^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu}\left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}\right)\left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}\right)^T$$

em que $\boldsymbol{m}^{(g)}$ é a média real da distribuição usada para gerar as amostras $\boldsymbol{x}_{i:\lambda}^{(g+1)}$.

- Espera-se que a adaptação da matriz de covariância permita amostrar de direções melhores.

Adaptação da Matriz de Covariância





Fonte: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/

Covariance Matrix Adaptation ES (CMA-ES)

- CMA-ES é baseado nas ideias apresentadas até então.
- Traz **muitas** melhorias a essas ideias básicas (junta ideias de **muitos** artigos).
- Escolhas de técnicas embasadas por demonstrações e experimentos.
- Preocupação em prover bons valores para hiperparâmetros, de modo que funcionem em diferentes problemas.
- Mecanismos de adaptação (tamanho de passo, média e covariância) tentam fazer o algoritmo se adaptar ao problema.

CMA-ES

- Amostragem usa tamanho de passo $\sigma^{(g)}$:

$$\boxed{\boldsymbol{x}_k^{(g+1)} \sim \boldsymbol{m}^{(g)} + \sigma^{(g)} N(\boldsymbol{0}, \boldsymbol{C}^{(g)})}$$

adaptados

- CMA-ES considera pesos para cada uma das μ melhores amostras: $w_1 \geq w_2 \geq \cdots \geq w_\mu > 0$ tal que $\sum_{i=1}^{\mu} w_i = 1$. Métrica importante:

$$\mu_{eff} = \left(\frac{||\boldsymbol{w}||_1}{||\boldsymbol{w}||_2}\right)^2 = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$$

$1 \leq \mu_{eff} \leq \mu$

Bizu: $\mu_{eff} \approx 0{,}3\lambda$

$w_i \propto \mu - i + 1$

CMA-ES

- Evolução da média considerando os pesos:

$$\boxed{\boldsymbol{m}^{(g+1)} = \boldsymbol{m}^{(g)} + c_m \sum_{i=1}^{\mu} w_i \left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)} \right)}$$

taxa de aprendizado

$c_m \approx 1$

CMA-ES

- Evolução da matriz de covariância também considera os pesos:

$$\boldsymbol{C}_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i \left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}\right)\left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}\right)^T$$

- Porém, para essa estimativa ser confiável, precisa-se de μ grande.
- Rank-μ-update aproveita informação das gerações anteriores para evitar isso.

CMA-ES (Rank-μ-Update)

- Primeira ideia: média das matrizes de covariância das gerações:

$$\boldsymbol{C}^{(g+1)} = \frac{1}{g+1}\sum_{i=0}^{g}\frac{1}{\sigma^{(i)^2}}\boldsymbol{C}_{\mu}^{(i+1)}$$

- Porém, mais interessante dar maior peso para informações mais atuais. Usar média móvel exponencial:

$$\boldsymbol{C}^{(g+1)} = \left(1-c_{\mu}\right)\boldsymbol{C}^{(g)} + c_{\mu}\frac{1}{\sigma^{(g)^2}}\boldsymbol{C}_{\mu}^{(g+1)}$$

taxa de aprendizado para rank-μ-update

$$c_{\mu} \approx \min\left(1, \frac{\mu_{eff}}{n^2}\right)$$

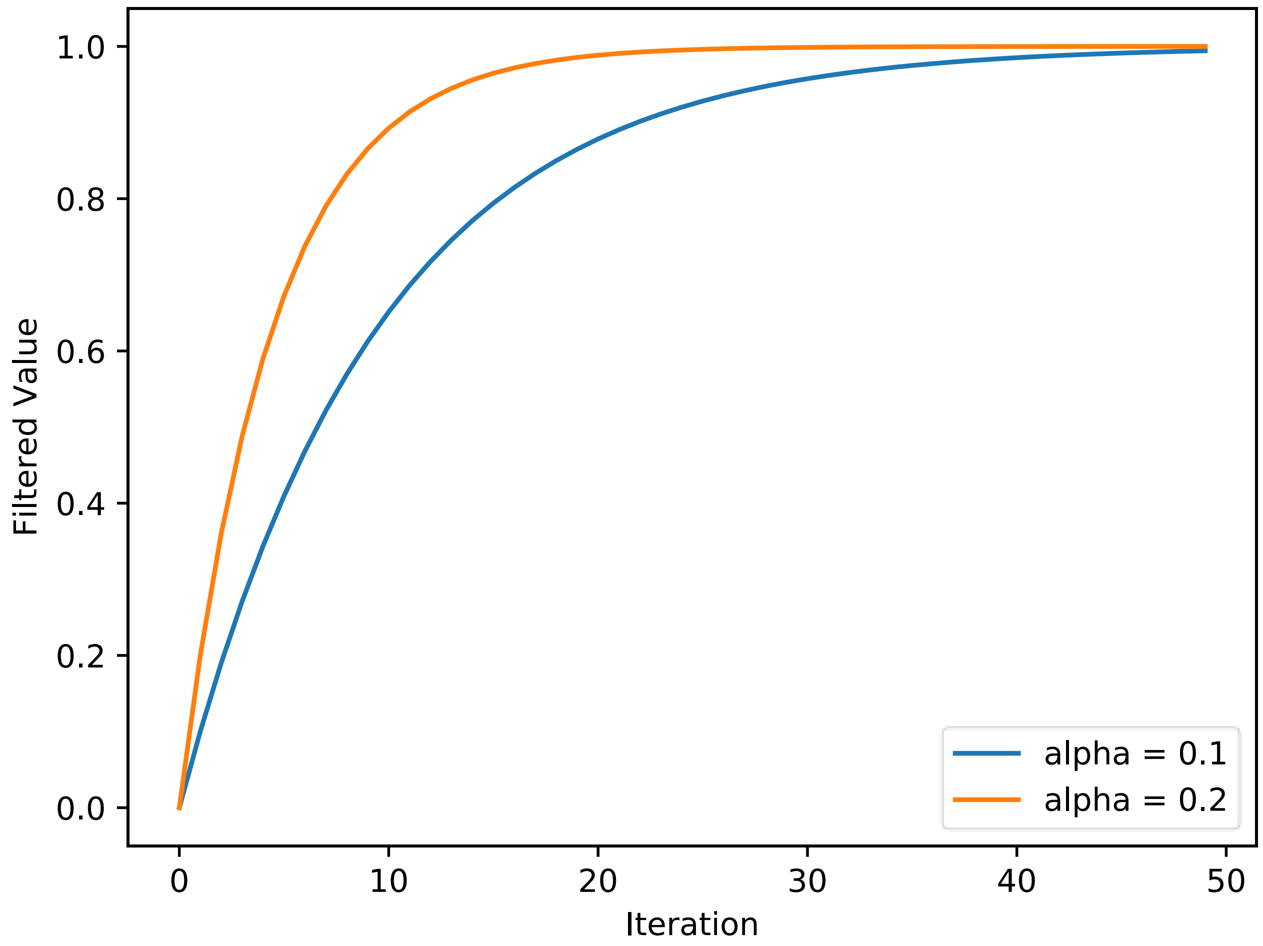
Média Móvel Exponencial

$$\bar{x} = (1 - \alpha)\bar{x} + \alpha x$$

- Uso muito comum em Engenharia, em especial Computação e IA.
- Implementa um filtro passa-baixas.
- Abrindo:

$$\bar{x}_k = (1 - \alpha)^k x_0 + \sum_{i=1}^{k} \alpha(1 - \alpha)^{k-i} x$$

Média Móvel Exponencial



CMA-ES (Rank-μ-Update)

- O autor ainda comentam de uma variação que considera λ pesos:

$$\boldsymbol{C}^{(g+1)} = \left(1 - c_\mu \sum_{i=1}^{\lambda} w_i\right) \boldsymbol{C}^{(g)} + c_\mu \sum_{i=1}^{\lambda} w_i \boldsymbol{y}_{i:\lambda}^{(g+1)} {\boldsymbol{y}_{i:\lambda}^{(g+1)}}^T$$

$$\boldsymbol{y}_{i:\lambda}^{(g+1)} = (\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}) / \sigma^{(g)}$$

$$w_1 \geq w_2 \geq \cdots \geq w_\mu > 0 \geq w_{\mu+1} \geq \cdots \geq w_\lambda$$

$$\sum\nolimits_{i=1}^{\mu} w_i = 1, \qquad \sum\nolimits_{i=1}^{\lambda} w_i \approx 0$$

CMA-ES (Rank-One-Update)

- Autores argumentam que a matriz de covariância perde informação de sinal.
- Para evitar isso, usar acumulação (*cumulation*) das atualizações:

$$\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} + \frac{\boldsymbol{m}^{(g)} - \boldsymbol{m}^{(g-1)}}{\sigma^{(g-1)}} + \frac{\boldsymbol{m}^{(g-1)} - \boldsymbol{m}^{(g-2)}}{\sigma^{(g-2)}}$$

- CMA-ES faz isso com média exponencial:

$$\boxed{\boldsymbol{p}_c^{(g+1)} = (1 - \underbrace{c_c}_{\text{taxa de aprendizado do } cumulation})\boldsymbol{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{eff}}\,\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}}$$

- $\boldsymbol{p}_c^{(g+1)}$ é chamado de *evolution path*. $c_c \approx 4/n, \boldsymbol{p}_c^{(0)} = \boldsymbol{0}$

CMA-ES

- Atualização da matriz de covariância combina rank-one-update e rank-μ-update:

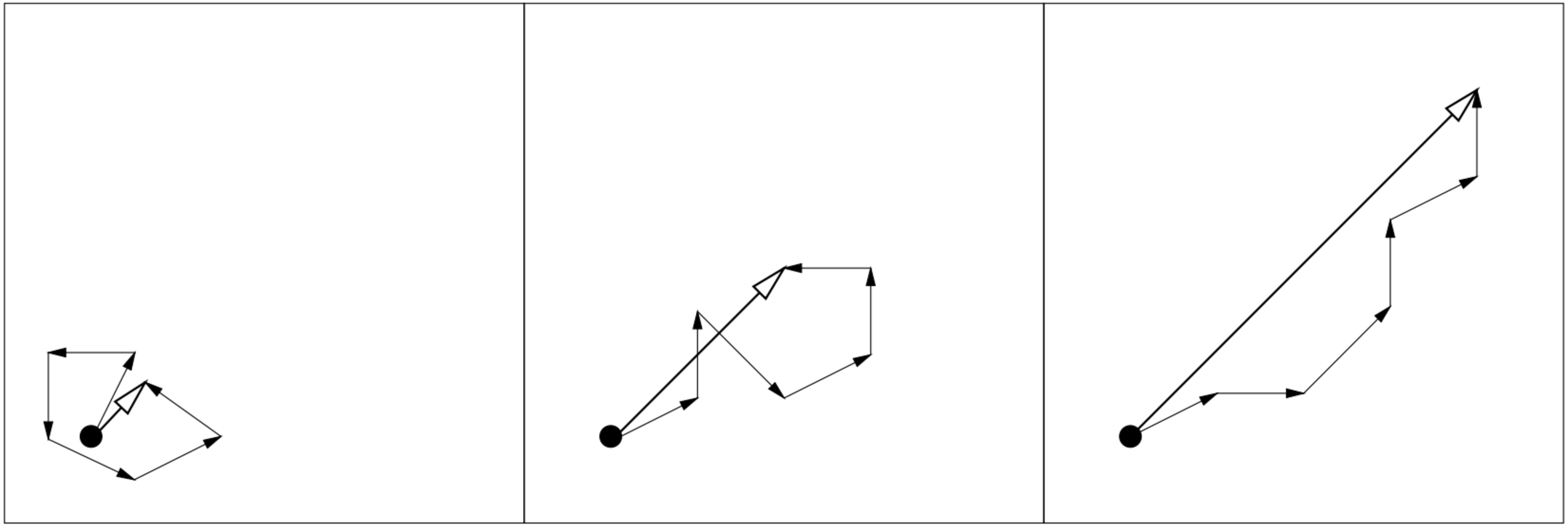
$$\boxed{\begin{aligned}&\boldsymbol{C}^{(g+1)}\\ &= \left(1 - c_1 - c_\mu\right)\boldsymbol{C}^{(g)} + c_1 \boldsymbol{p}_c^{(g+1)} \boldsymbol{p}_c^{(g+1)^T} + c_\mu \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}^{(g+1)} \boldsymbol{y}_{i:\lambda}^{(g+1)^T}\end{aligned}}$$

taxa de aprendizado do rank-one-update

$c_1 \approx 2/n^2$, $c_\mu \approx \mu_{eff}/n^2$

$$\boldsymbol{y}_{i:\lambda}^{(g+1)} = \frac{\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}$$

CMA-ES (Controle do Passo)



CMA-ES (Controle do Passo)

- Calcula-se outro *evolution path*:

$$\boxed{\boldsymbol{p}_\sigma^{(g+1)} = (1 - c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\,\boldsymbol{C}^{(g)^{-\frac{1}{2}}}\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}}$$

Taxa de aprendizado de σ

$\boldsymbol{p}_\sigma^{(0)} = \boldsymbol{0}$, $c_\sigma \approx 4/n$

evitar que dependa da "orientação" de $\boldsymbol{C}$

$$\boxed{\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|}{E\left\|N(\boldsymbol{0}, \boldsymbol{I})\right\|} - 1\right)\right)}$$

damping de σ

$d_\sigma \approx 1 + \sqrt{\frac{\mu_{eff}}{n}}$

CMA-ES (Inicialização)

- Entrada: $\boldsymbol{m}^{(0)}, \sigma, \lambda$.
- Inicialização: $\boldsymbol{C} = \boldsymbol{I}, \boldsymbol{p}_c = \boldsymbol{0}, \boldsymbol{p}_\sigma = \boldsymbol{0}$.
- Hiperparâmetros:

$c_c \approx 4/n, c_\sigma \approx 4/n, c_1 \approx 2/n^2, c_\mu \approx \mu_{eff}/n^2, c_1 + c_\mu \leq 1,$

$d_\sigma \approx 1 + \sqrt{\frac{\mu_{eff}}{n}}, w_1 \geq w_2 \geq \cdots \geq w_\mu$ t.q. $\mu_{eff} \approx 0{,}3\lambda, \mu = \lfloor \lambda/2 \rfloor$.

- Ver página 32 do tutorial de CMA-ES.

CMA-ES (*Loop*)

$$\boldsymbol{x}_i \sim \boldsymbol{m} + \sigma N(\boldsymbol{0}, \boldsymbol{C})$$

$$\boldsymbol{y}_{i:\lambda} = \frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}}{\sigma}, \boldsymbol{m}' = \boldsymbol{m}, \boldsymbol{m} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$$

$$\boldsymbol{p}_c = (1 - c_c)\boldsymbol{p}_c + \sqrt{c_c(2 - c_c)\mu_{eff}}\,\frac{\boldsymbol{m} - \boldsymbol{m}'}{\sigma}$$

$$\boldsymbol{p}_\sigma = (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\,\boldsymbol{C}^{-\frac{1}{2}}\frac{\boldsymbol{m} - \boldsymbol{m}'}{\sigma}$$

$$\boldsymbol{C} = \left(1 - c_1 - c_\mu\right)\boldsymbol{C} + c_1 \boldsymbol{p}_c \boldsymbol{p}_c^T + c_\mu \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^T$$

$$\sigma = \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\boldsymbol{p}_\sigma||}{||N(\boldsymbol{0}, \boldsymbol{I})||} - 1\right)\right)$$

Quando usar CMA-ES?

- Função de otimização não-linear, não-quadrática, não-convexa.
- Rugosa: não-suave, descontínua, multimodal (múltiplos mínimos locais) e ruidosa.
- Muitos parâmetros.
- Não-separável (um parâmetro influencia no outro).

Quando usar CMA-ES?

- Função de otimização não-linear, não-quadrática, não-convexa.
- Rugosa: não-suave, descontínua (problema para derivada), multimodal (múltiplos mínimos locais) e ruidosa.
- Muitos parâmetros.
- Não-separável (um parâmetro influencia no outro).

Quando não usar CMA-ES

- Problemas separáveis: rodar otimizações menores.
- Gradiente fácil de calcular: *Gradient Descent*.
- Dimensão pequena ($n \leq 3$): Nelder-Mead.

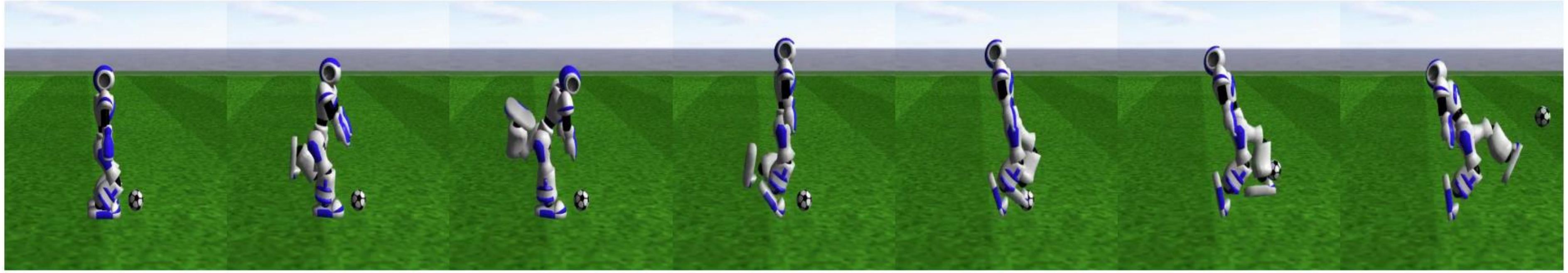
Estudos de Caso

Otimização de Keyframes

- Exemplo: fazer robô chutar a bola o mais distance possível.

f =distância que a bola andou após o robô chutar a bola.

$\mathbf{x}$=sequência de ângulos das juntas do robô humanoide.



- Quantos parâmetros?
- 22 juntas x 7 keyframes + 6 tempos = 161 parâmetros.

Otimização de Keyframes

- Problema muito difícil:
  - Muitos parâmetros.
  - Dinâmica complexa.
  - Muitas descontinuidades (e.g. robô bater o pé no chão).
  - Muitos mínimos locais.
  - Estocástico (ruído do simulador).
- Mesmo CMA-ES tem dificuldade de resolver do “zero”.
- Como conseguir um bom “chute” inicial (semente)?
- Por que não roubar movimento dos outros times? 😈

Otimização de Keyframes

- Pode-se “observar” movimento do adversário.
- UT Austin Villa na verdade modificou o servidor para enviar posições das juntas do adversário.
- Usou movimento “roubado” como semente do CMA-ES e obteve chute melhor ainda!
- Amostragem das juntas a cada 60 ms.

Otimização de Keyframes

- Com amostragem de 60 ms, o chute ficou com 1958 parâmetros!
- Parâmetros demais para o CMA-ES.
- Redução de parâmetros:
  - Remoção das juntas da cabeça.
  - “Fusão” de parâmetros de juntas que variavam menos de 0,5° entre dois *frames* consecutivos.
  - Remoção de keyframes antes do chute estar “preparado”.
- Com redução ficou com 59 parâmetros. Mais 3 parâmetros para a posição inicial do robô (x, y e ângulo): 62 parâmetros.

Chute Inicial (FCPortugal)


RoboViz
UTAustinVilla 0:0 <Right> 0.0
Playmode: BeforeKickOff
Semente Amostrada
1

Chute Otimizado (Distância)

Chute Otimizado (Precisão)

Chute Otimizado (Ar)

Chute Otimizado (Kickoff)

Funções de *Fitness* (Maximização)

- Distância:

$$f_{distance} = \begin{cases} -1, se\ falha \\ finalBallLocation.x, caso\ contrário \end{cases}$$

Falha: queda, chutar para trás ou tocar na bola antes de chutá-la.

- Precisão:

$$f_{accuracy} = \begin{cases} -1, se\ falha \\ finalBallLoc.x * e^{-angleOffset^2/180}, caso\ contrário \end{cases}$$

Funções de *Fitness* (Maximização)

- Bola no ar:

$$f_{air} = \begin{cases} -1, falha \\ 0, errou\ o\ gol \\ 100 + finalBallLoc.x + 2 * airDist, caso\ contrário \end{cases}$$

airDist: distância percorrida pela bola antes de ficar abaixo de 0,5 m acima do chão.

Funções de *Fitness* (Maximização)

- Kickoff (multi-agente):

$$f_{touch} = \begin{cases} -1, se\ falha \\ 10 - finalBallLoc.magnitude, caso\ contrário \end{cases}$$

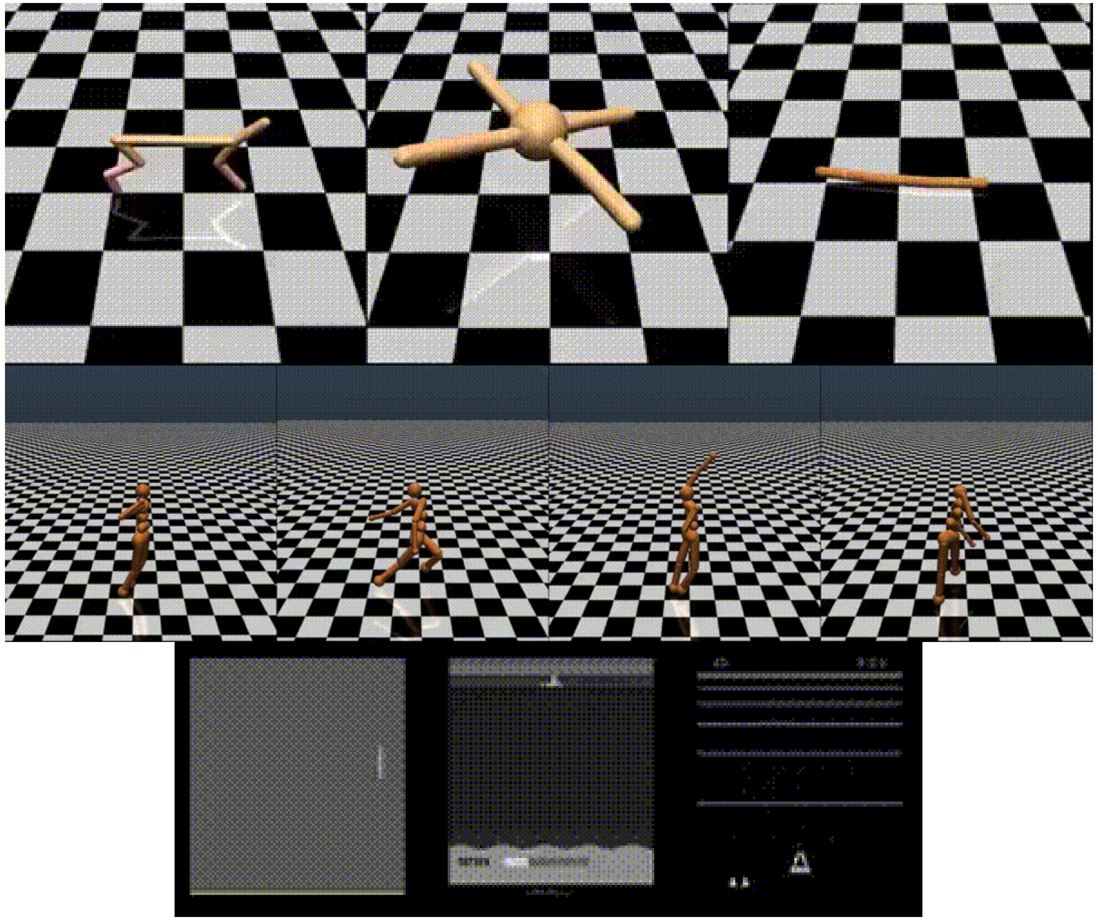
$$f_{kickoff} = \begin{cases} -1, se\ falha \\ -1, ordem\ de\ toque\ errada \\ -1, algum\ agente\ errou \\ 100 + finalBallLoc.x + 2 * airDist, caso\ contrário \end{cases}$$

Otimização com CMA-ES

- População de 200 amostras.
- Execução em *cluster*.
- Em trabalho semelhante (caminhada), disse que teve *speedup* de 150. Otimização da caminhada levaria >100 dias em único computador.
- Trabalho mais recente, diz que execução sequencial demoraria 1,5 ano.
- Patrick (UT Austin Villa) já usou CMA-ES em muitos contextos:
  - Caminhada.
  - Diferentes tipos de caminhada.
  - Levantar-se.
  - Diferentes tipos de chute.

CMA-ES x Deep Reinforcement Learning

- Artigo do OpenAI de 2017.
- CMA-ES x TRPO.
- Usou CMA-ES para treinar rede neural da política.
- CMA-ES ganhou em:
  - Muito paralelizável.
  - Não precisa tunnar hiperparâmetros.
- Resumo: precisa de mais dados para treinar, mas ganha se você tiver um *cluster* muito grande.



Para Saber Mais

- Post legal para entender ES: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/
- CMA-ES:
  - Página do CMA-ES: http://cma.gforge.inria.fr/cmaes_sourcecode_page.html
  - Tutorial do CMA-ES: https://hal.inria.fr/hal-01297037/file/tutorial.pdf
- Otimização de keyframes usando CMA-ES: http://www.cs.utexas.edu/~pstone/Papers/bib2html/b2hd-LNAI14-Depinet.html
- Estudo do CMA-ES pela OpenAI: https://openai.com/blog/evolution-strategies/

Laboratório 5

Laboratório 5

- Lab simples.
- Implementar estratégia evolutiva simples.
- Usar código do CMA-ES disponível na Internet.
- Comparar em funções *benchmark*.