

# T319 - Introdução ao Aprendizado de Máquina: *Regressão Linear (Parte III)*



***Inatel***

Felipe Augusto Pereira de Figueiredo  
felipe.figueiredo@inatel.br

# Recapitulando

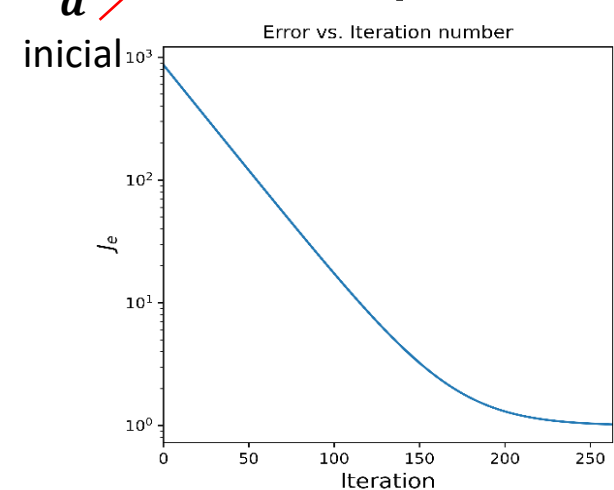
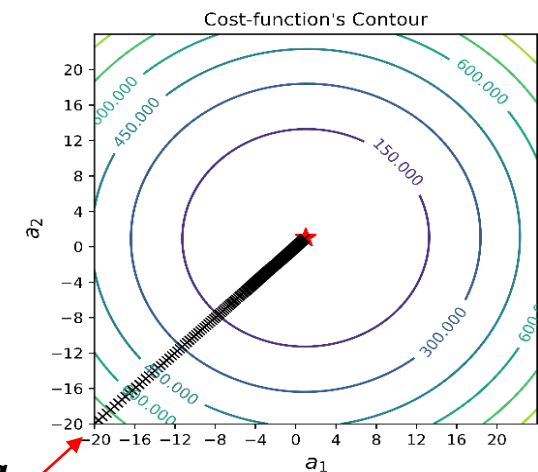
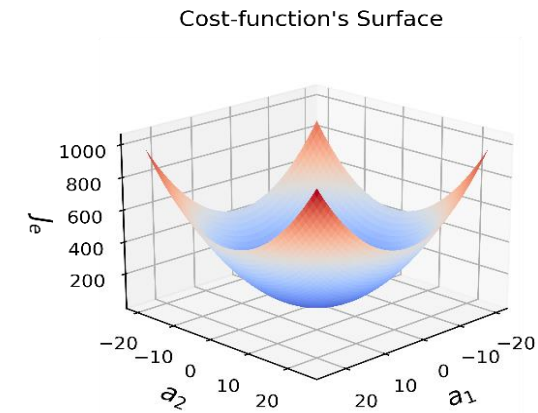
- Discutimos sobre o vetor gradiente.
- Aprendemos dois algoritmos que usam o vetor gradiente para a resolução de problemas de otimização.
  - Gradiente Ascendente para problemas de maximização.
  - Gradiente Descendente para problemas de minimização.
- Vimos as três versões do gradiente descendente e as comparamos.
- Nesta parte, discutiremos o quão importante é o ajuste do passo de aprendizagem,  $\alpha$ .

# Escolha do Passo de Aprendizagem

- Conforme nós já aprendemos, enquanto a **direção** para o ponto de mínimo é determinada pelo **vetor gradiente** da **função de erro**, o **passo de aprendizagem** determina o quão grande esse passo é dado naquela direção.

$$\mathbf{a} \leftarrow \mathbf{a} - \alpha \frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}}$$

- Portanto, a **escolha do passo de aprendizagem (hiperparâmetro) é muito importante**:
  - Caso ele seja muito pequeno, a convergência do algoritmo levará muito tempo.
  - Exemplo**: com  $\alpha = 0.01$ , o algoritmo atinge o valor ótimo após mais de 250 épocas.
    - Passos muito curtos, fazem com que o algoritmo caminhe vagarosamente em direção ao **mínimo global** da **função de erro**.

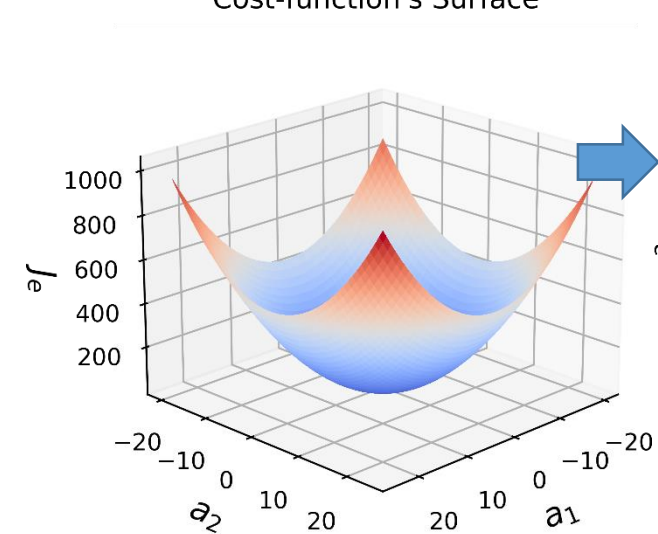


# Escolha do Passo de Aprendizagem

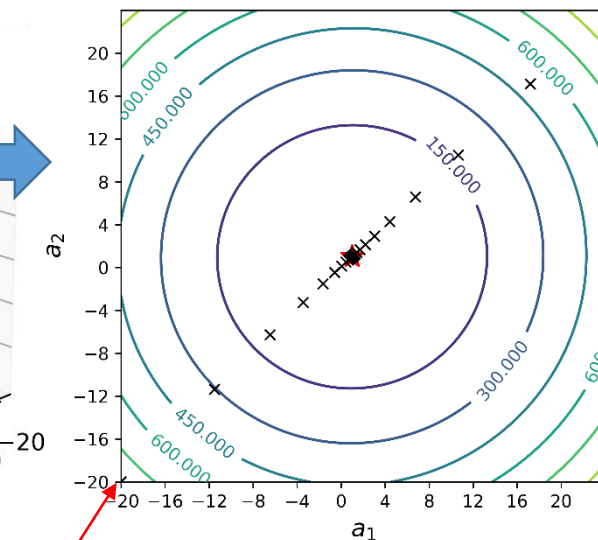
[Exemplo: linear regression selecting the learning rate.ipynb](#)

- Caso o ***passo de aprendizagem*** seja muito grande, o algoritmo pode nunca convergir.
- Se  $\alpha$  for grande, mas não tão grande assim, o algoritmo fica “pulando” ou “oscilando” de um lado para o outro da superfície de erro até que ele converge, por sorte (veja exemplo #1).
- Em outros casos, quando  $\alpha$  é muito grande, a cada iteração o algoritmo “pula” para um valor mais alto que antes, e assim, divergindo (veja exemplo #2).

Cost-function's Surface



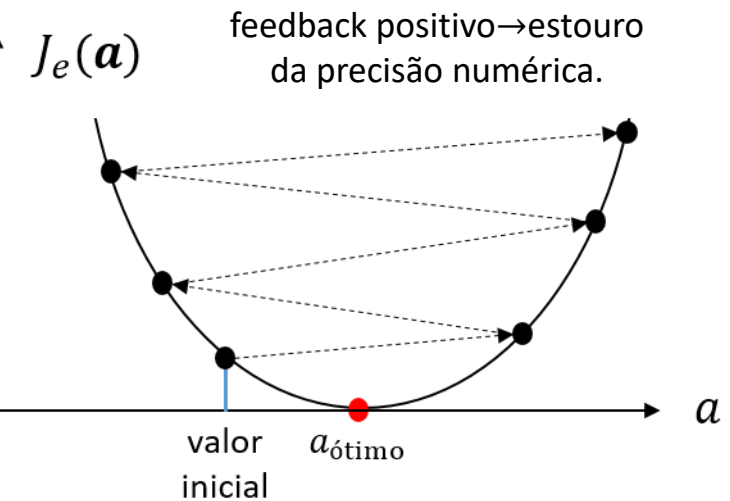
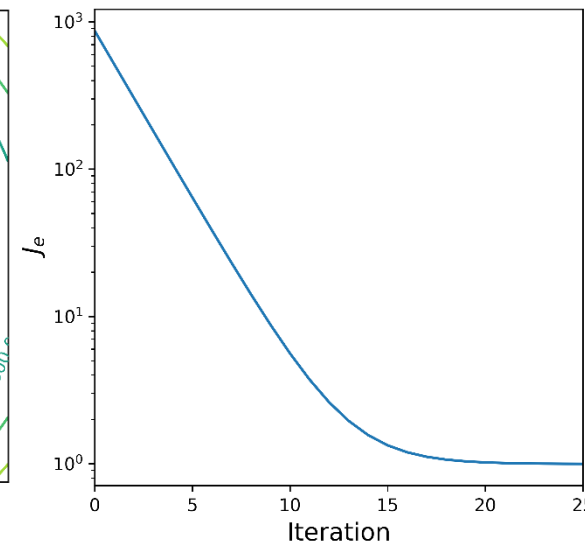
Cost-function's Contour



$a$  inicial

Exemplo #1

Error vs. Iteration number



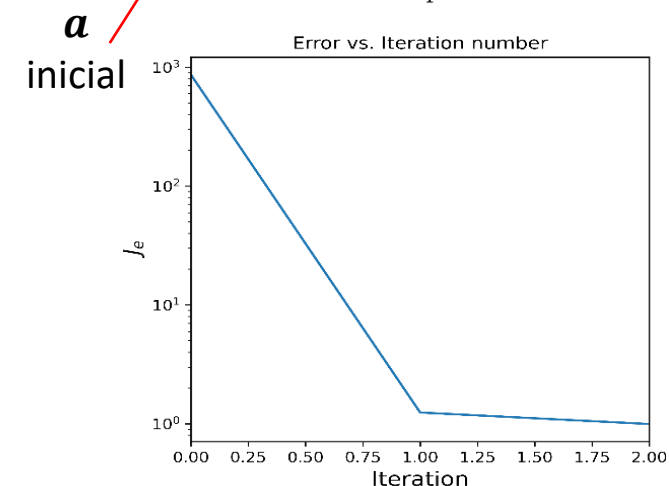
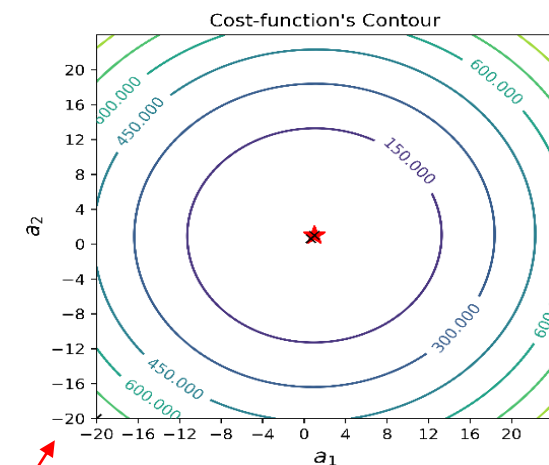
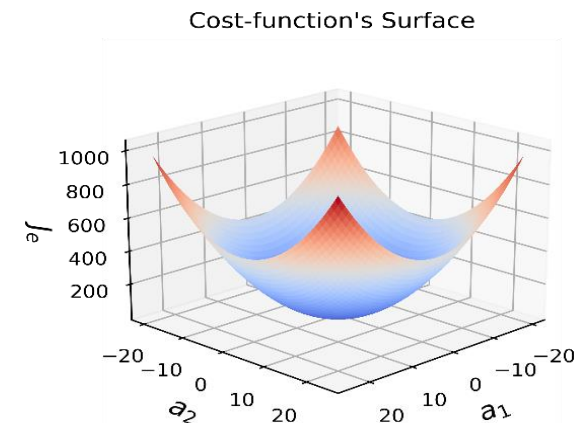
Exemplo #2

# Escolha do Passo de Aprendizagem

- Portanto, o valor ***passo de aprendizagem*** deve ser ***explorado*** para se encontrar um ***valor ideal*** que acelere a ***descida do gradiente*** de forma ***estável*** (ou seja, acelere a convergência).
  - O exemplo ao lado, converge para o ***mínimo global*** em apenas 2 iterações.
- Portanto, a escolha do ***passo de aprendizagem*** pode ser bastante demorada.
- Uma regra empírica para exploração do passo de aprendizagem é usar a seguinte sequência (***ajuste manual***):

..., 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0, ...

3 ×      ≈ 3 ×      3 ×      ≈ 3 ×      3 ×      ≈ 3 ×



# Como depurar o algoritmo do GD?

- Uma maneira de se **depurar** o algoritmo do **gradiente descendente**, principalmente quando não é possível se plotar o gráfico da superfície de contorno, é plotar o gráfico do erro (EQM) em função do número de iterações.
  - Figura A  $\Rightarrow$  Passo ideal: convergência rápida
    - Erro diminui rapidamente nas primeiras épocas e depois diminui quase que a uma taxa constante.
    - Convergência pode ser declarada, por exemplo, quando o erro entre duas iterações subsequentes for menor do que um limiar pré-definido (e.g.,  $1e-3$ ).
  - Figura B  $\Rightarrow$  Passo pequeno demais: convergência lenta.
  - Figuras C e D  $\Rightarrow$  Passo grande demais: divergência (C) e oscilação (D).

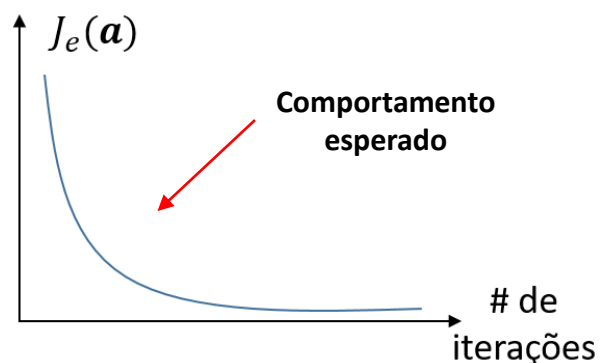


Figura A

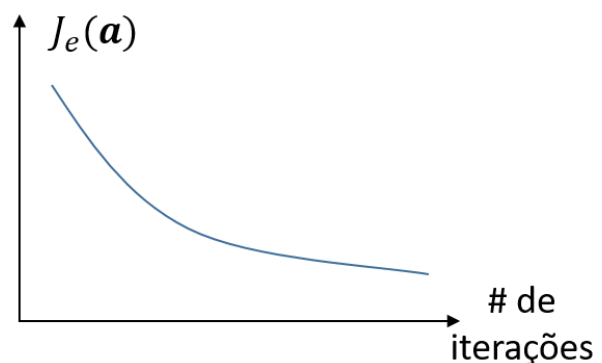


Figura B

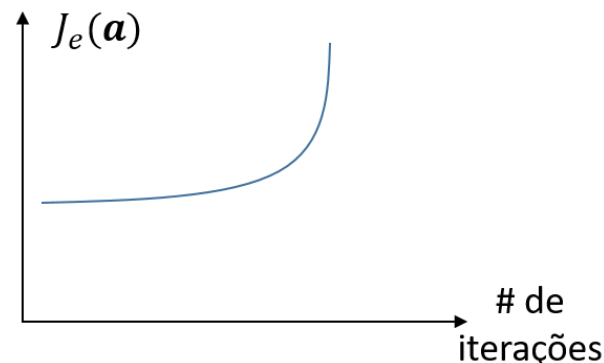


Figura C

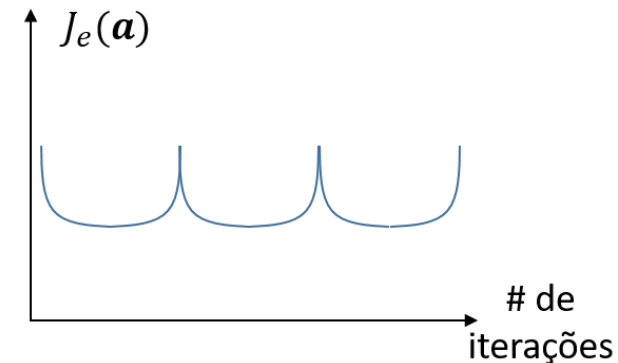
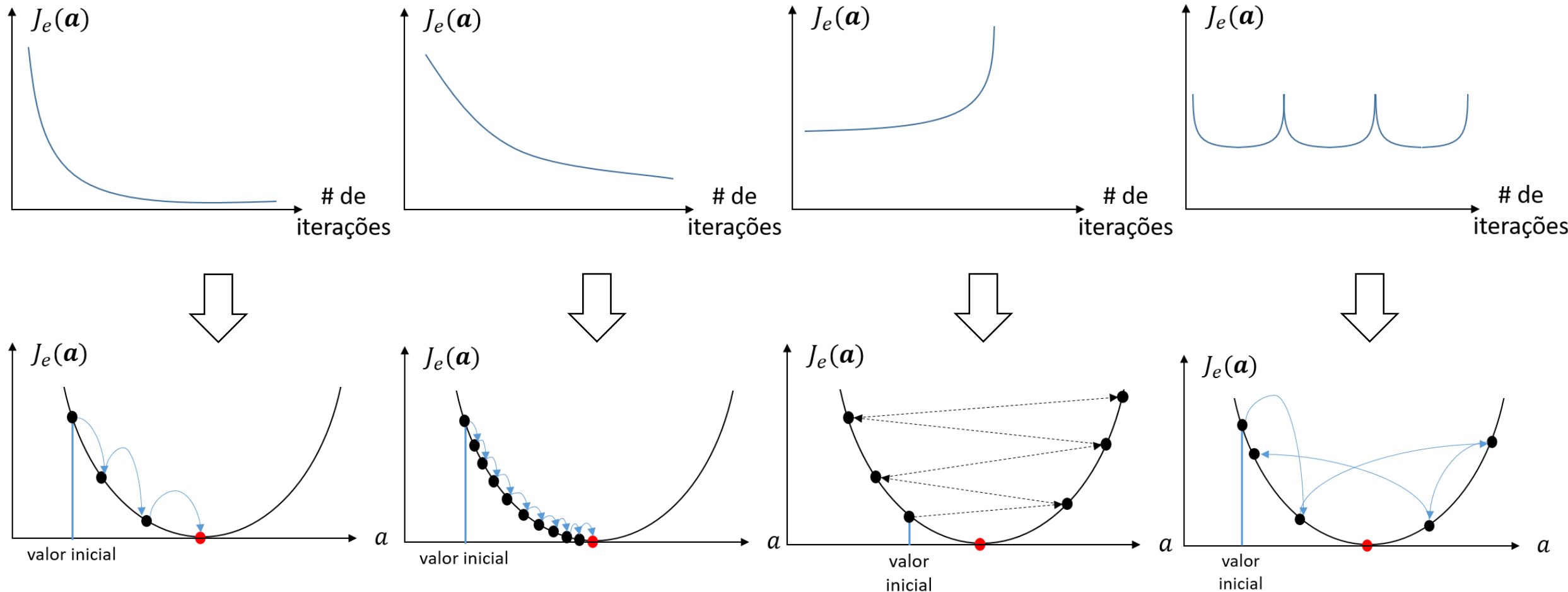


Figura D

# Como depurar o algoritmo do GD?



# Como configurar o passo de aprendizagem?

Além do ***ajuste manual*** (escolha de  $\alpha$  por tentativa e erro), podemos também usar as seguintes abordagens para configurar  $\alpha$ :

- **Redução programada:** redução de  $\alpha$  ao longo do processo de treinamento, ou seja, ao longo das iterações.
  - A forma mais simples é diminuir o passo de aprendizagem linearmente de um valor inicial grande até um valor pequeno.
  - Abordagem muito usada com GD estocástico e mini-batch para garantir a convergência para o ponto de mínimo.
- **Variação adaptativa:**  $\alpha$  é adaptativamente ajustado de acordo com a performance do modelo, além disso, pode-se ter passos diferentes para cada peso do modelo e atualizá-los de forma independente.
  - **Vantagem:** na maioria dos casos, não é necessário se ajustar manualmente nenhum ***hiperparâmetro*** como no caso dos esquemas de redução programada.

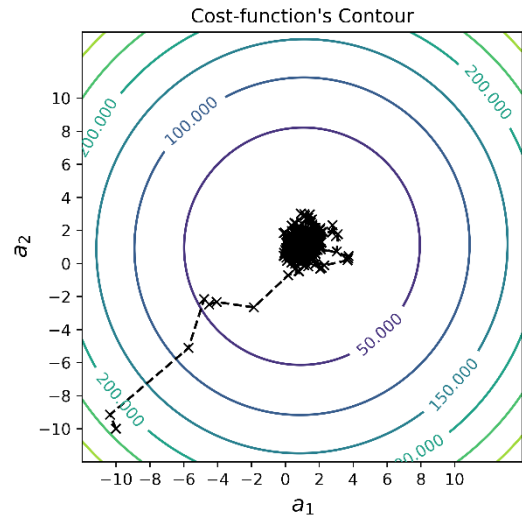


# Redução Programada do Passo de Aprendizagem

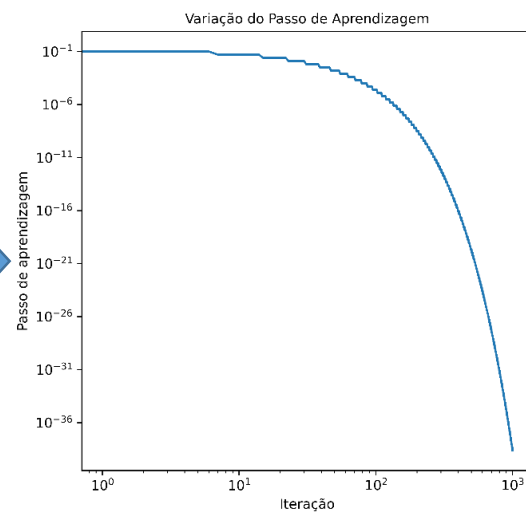
- Os três tipos mais comuns de implementação da **redução programada** do passo de aprendizagem são:
  - **Decaimento gradual**: também conhecido como **decaimento por etapas** ou **por degraus**. Ele reduz a taxa de aprendizagem inicial,  $\alpha_0$ , de um fator,  $\tau_0$ , a cada número pré-definido de iterações ou épocas,  $\beta$ . Um valor típico para reduzir a taxa de aprendizado é de  $\tau_0 = 0.5$  a cada número pré-definido de épocas.
  - **Decaimento exponencial**: é expresso pela equação  $\alpha = \alpha_0 e^{-kt}$ , onde  $\alpha_0$  e  $k$  são hiperparâmetros e  $t$  é o número da iteração ou época corrente.
  - **Decaimento temporal**: é expresso pela equação  $\alpha = \alpha_0 / (1+kt)$  onde  $\alpha_0$  e  $k$  são hiperparâmetros e  $t$  é o número da iteração ou época corrente.
- Na prática, o **decaimento gradual** é o mais utilizado entre os 3, pois seus **hiperparâmetros** (a taxa de decaimento,  $\tau_0$ , e o intervalo para redução,  $\beta$ ) são mais interpretáveis do que o hiperparâmetro  $k$ , que dita a taxa de decaimento do passo de aprendizagem.
- Mas percebam que ainda temos que encontrar os **hiperparâmetros**.

# Exemplo: GDE com Redução Programada de $\alpha$

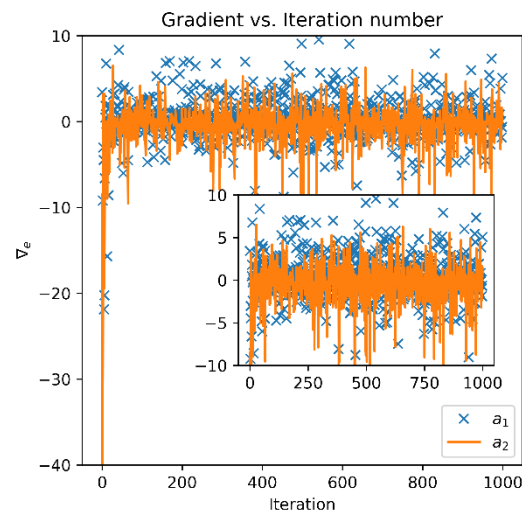
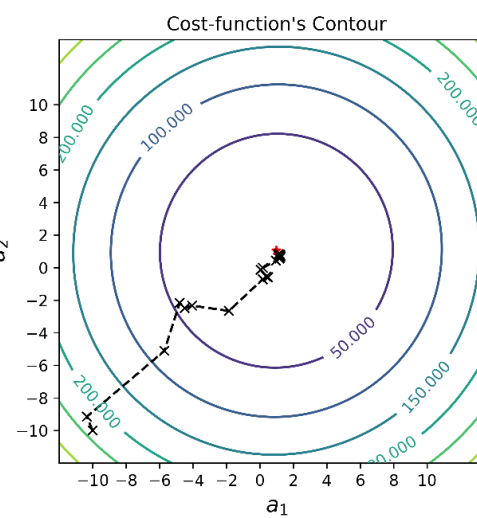
GDE sem redução programada



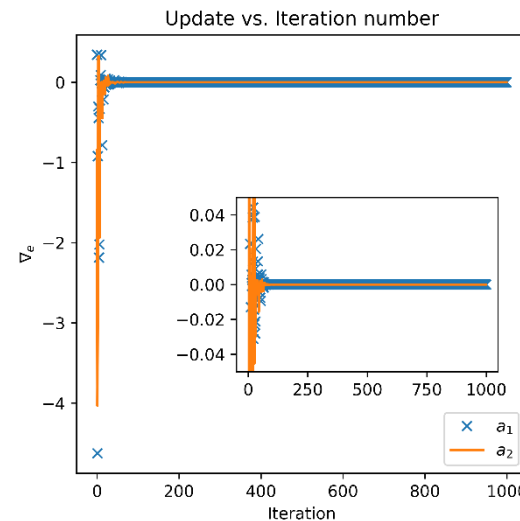
Redução programada



GDE com redução programada



```
# learning schedule: Gradual decay.  
def stepDecay(alpha_init, t, beta=8.0, drop=0.5):  
    """Gradual decay."""  
    alpha = alpha_init * math.pow(drop, math.floor((1+t)/beta))  
    return alpha
```

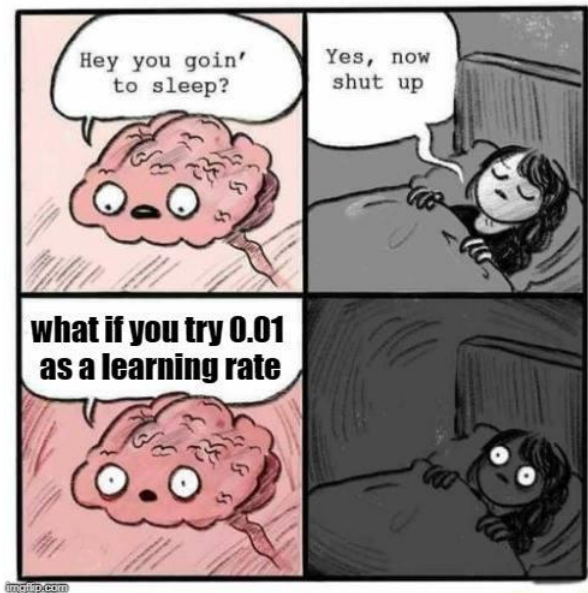


- Exemplo usando GDE com **decaimento gradual**.
- O caminho com **decaimento gradual** também não é regular para o ponto de mínimo.
- Apresenta algumas mudanças de direção ao longo do caminho.
- Porém, a oscilação em torno do mínimo é bastante minimizada devido à **diminuição gradual** de  $\alpha$ .
- Conseguimos visualizar melhor o efeito da redução de  $\alpha$  nas figuras que mostram o gradiente.
- **Conclusão:** um passo de aprendizagem que diminui ao longo das iterações permite que o algoritmo se estabilize próximo ao ponto de mínimo global.

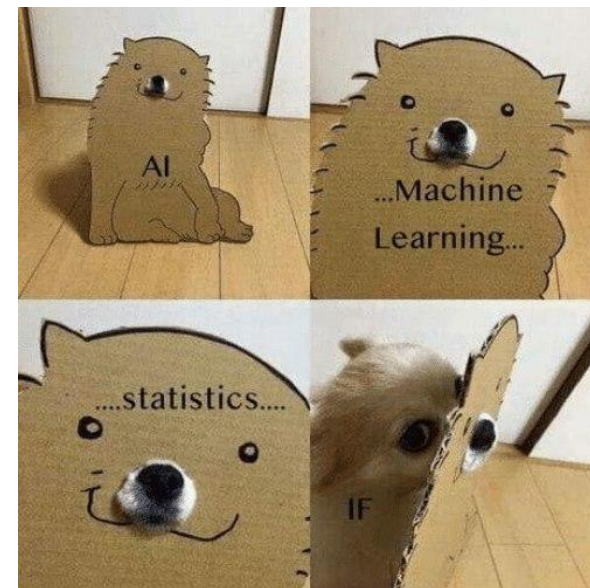
# Tarefas

- **Quiz:** “*T319 - Quiz - Regressão: Parte III*” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #4](#).
  - Pode ser baixado do MS Teams ou do GitHub.
  - Pode ser respondido através do link acima (na nuvem) ou localmente.
  - [Instruções para resolução e entrega dos laboratórios](#).
  - **Laboratórios podem ser feitos em grupo, mas as entregas devem ser individuais.**

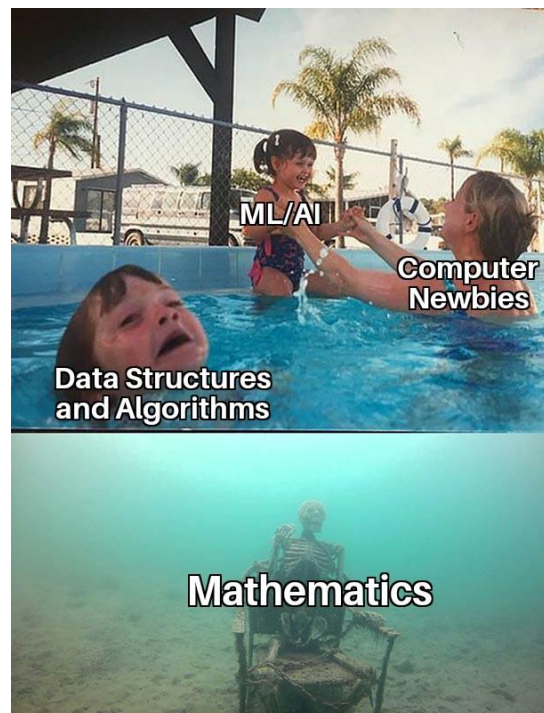
Obrigado!



**When someone asks why you never stops talking about machine learning**

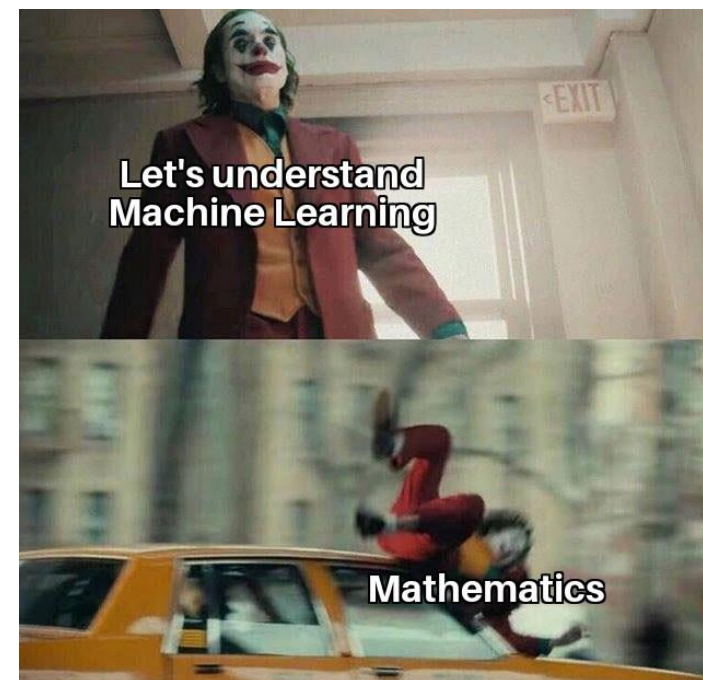


IF IF IF IF IF IF IF IF IF IF WE!

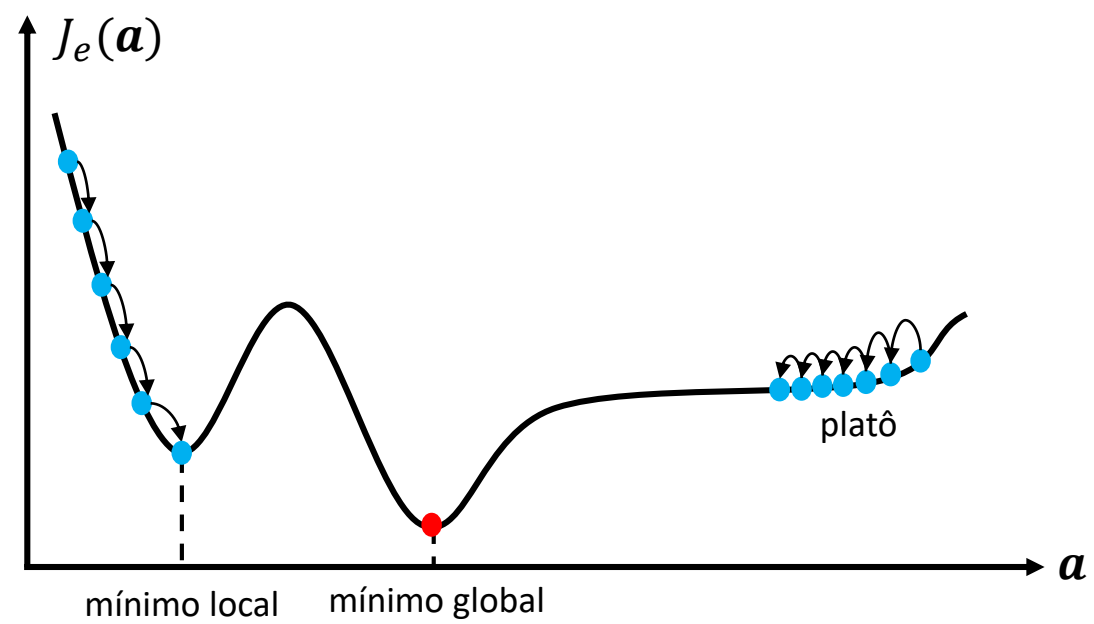


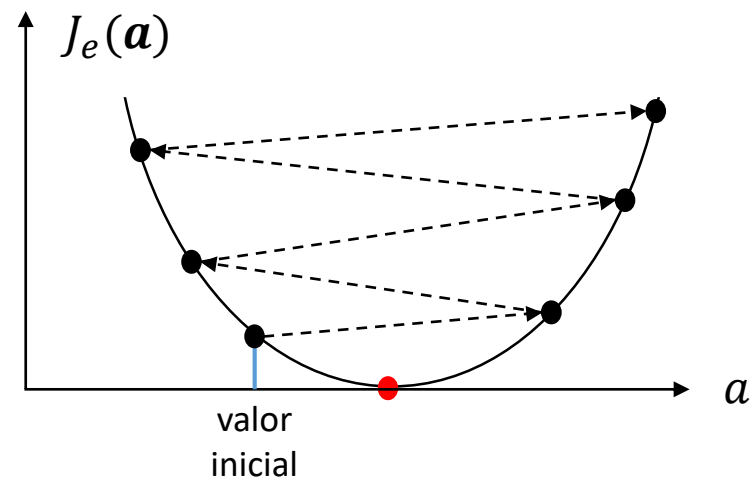
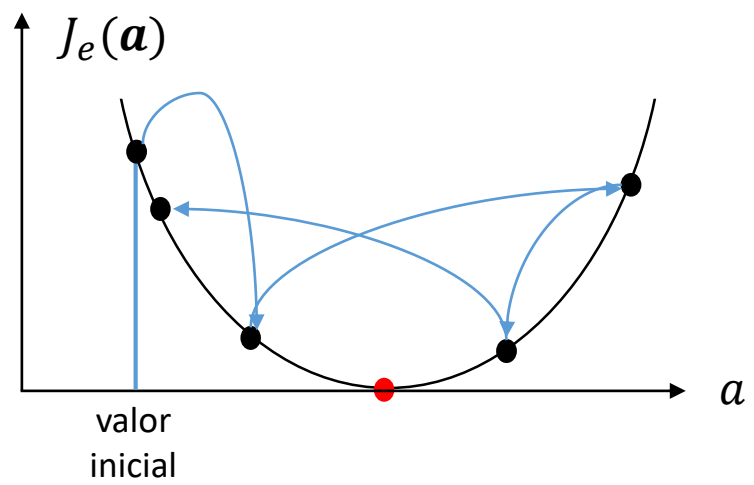
Albert Einstein: Insanity Is Doing the Same Thing Over and Over Again and Expecting Different Results

Machine learning:

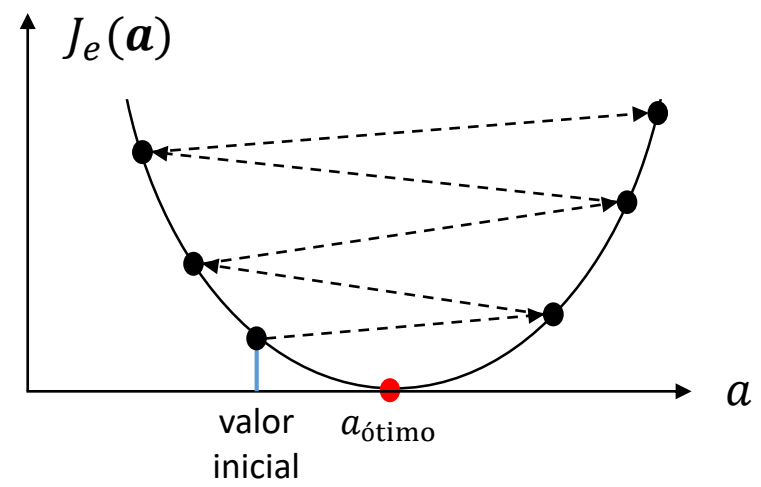


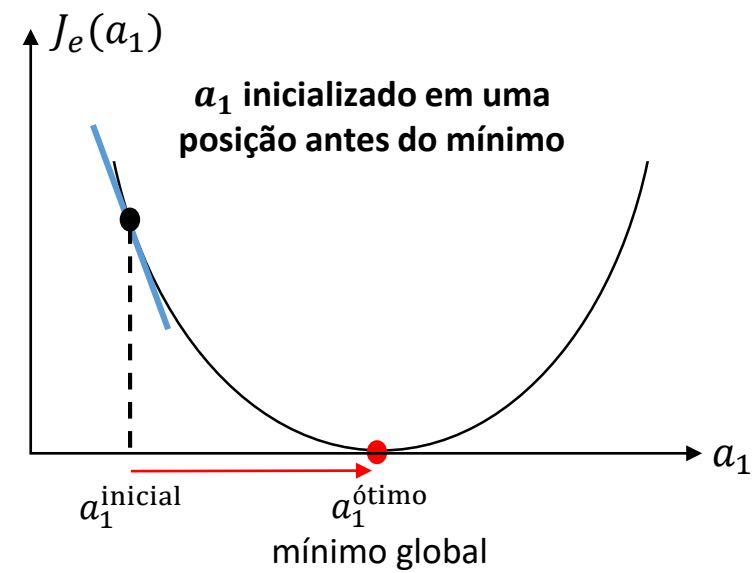
FIGURAS



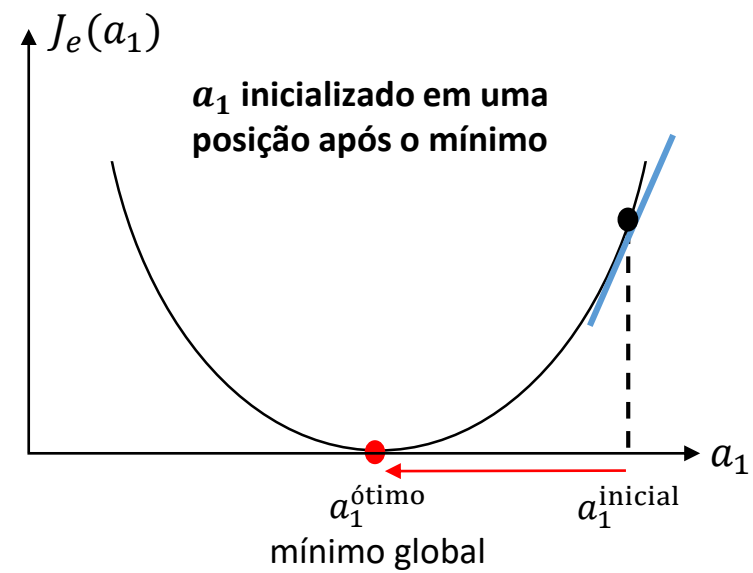




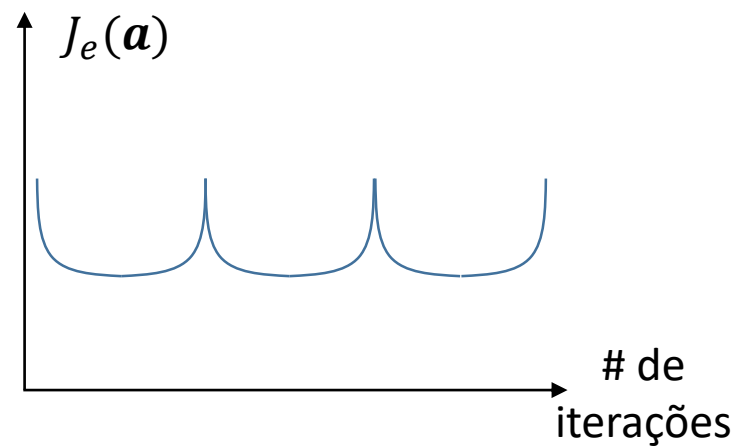
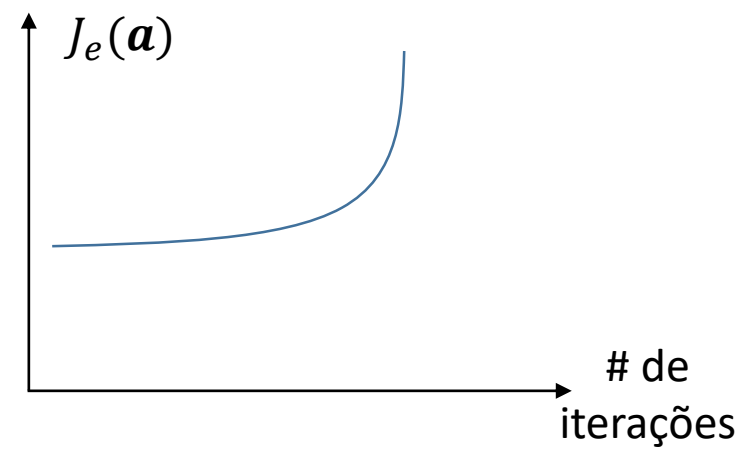
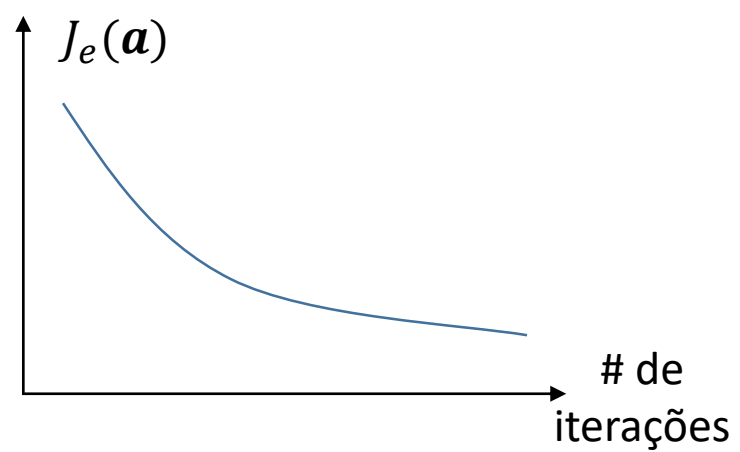
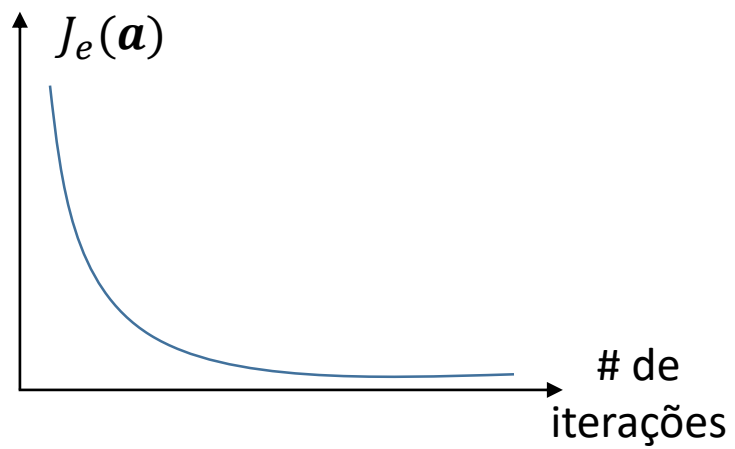


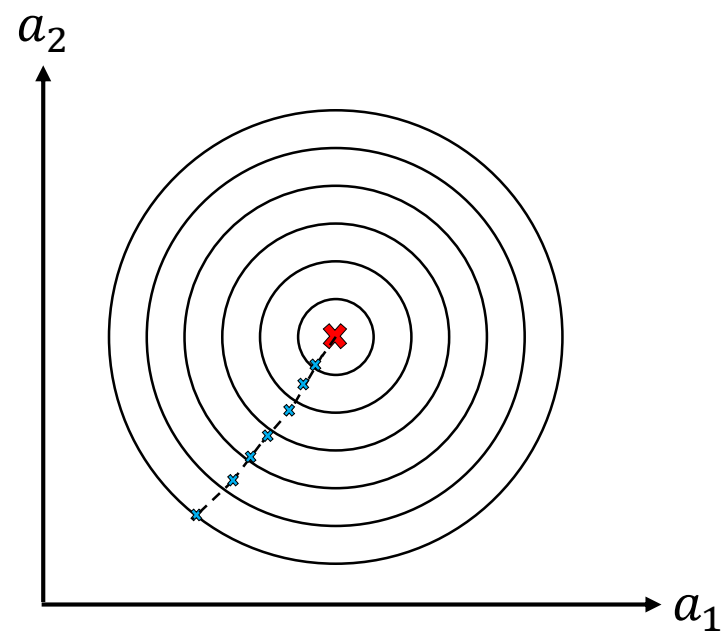
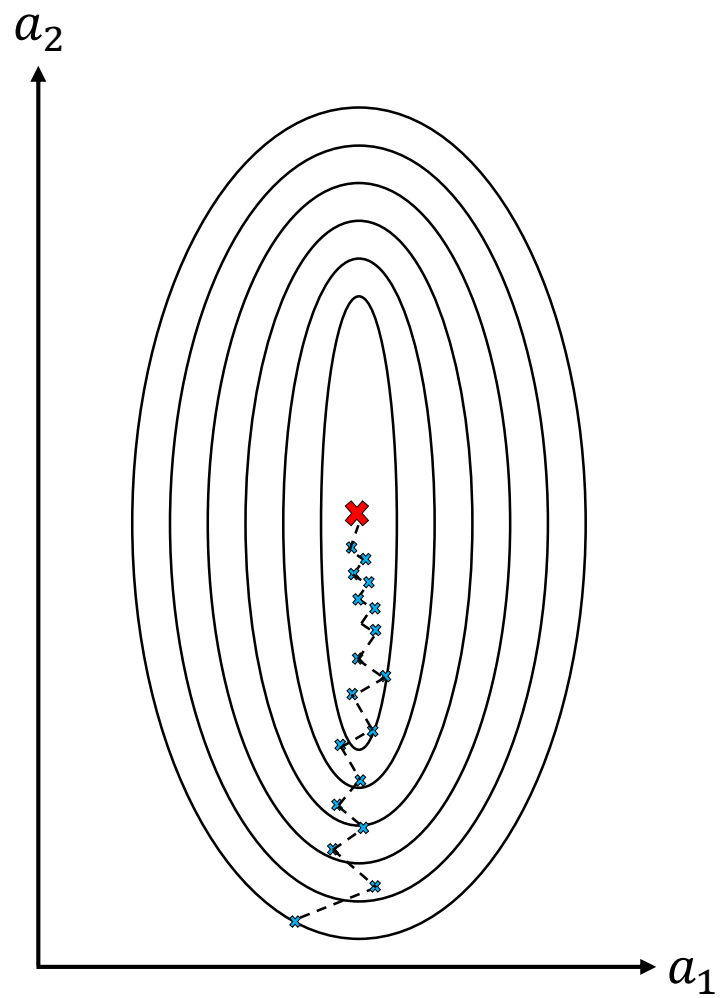


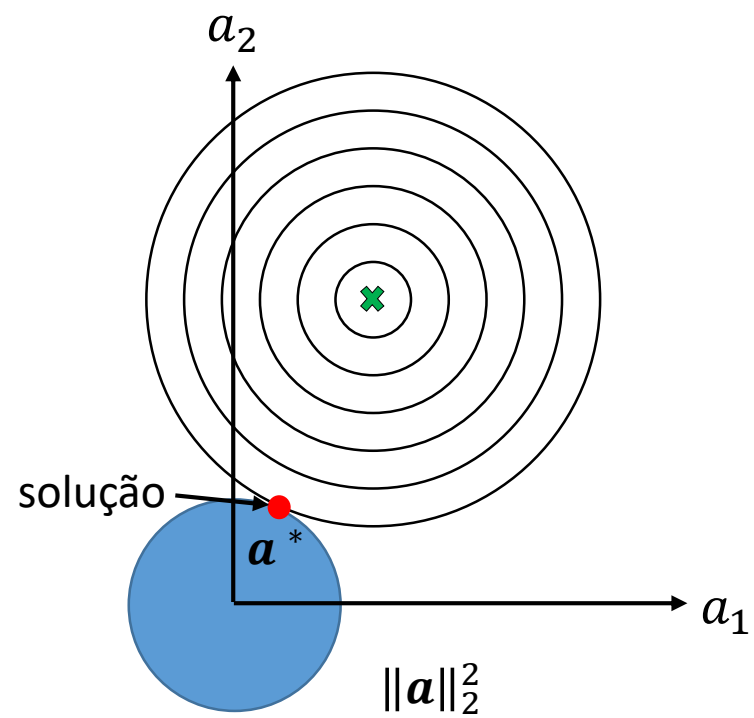
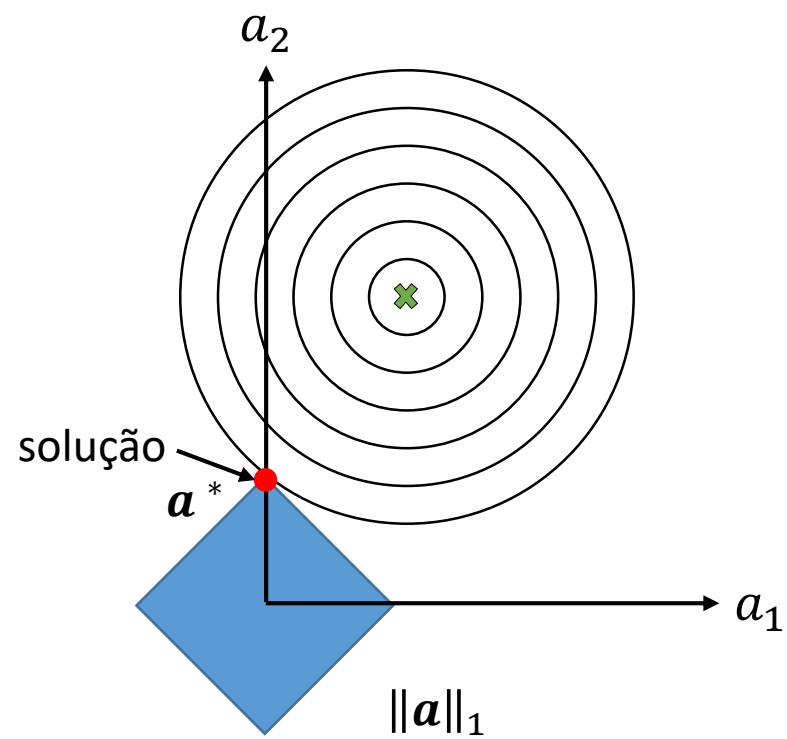
**gradiente negativo:**  $a_1 = a_1^{\text{inicial}} + \alpha \nabla J_e(a_1)$   
 $a_1$  aumenta e se aproxima do mínimo



**gradiente positivo:**  $a_1 = a_1^{\text{inicial}} - \alpha \nabla J_e(a_1)$   
 $a_1$  diminuiu e se aproxima do mínimo







## Gradiente Descendente Estocástico

