

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



Inatel

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

Recapitulando

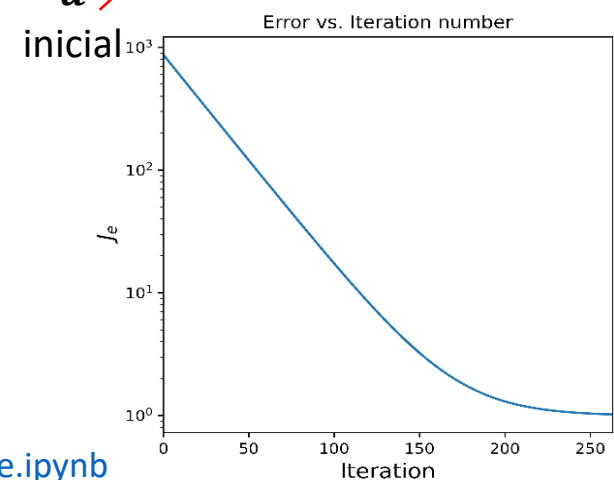
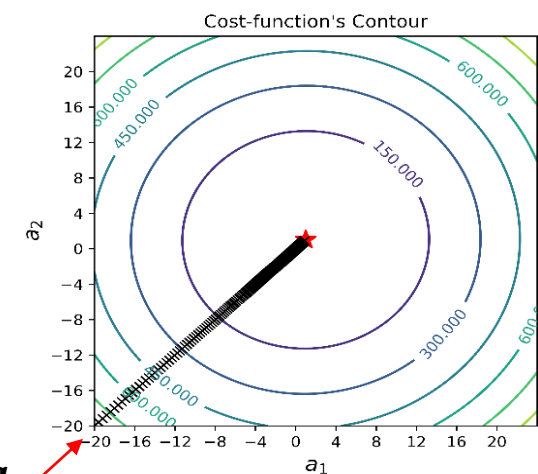
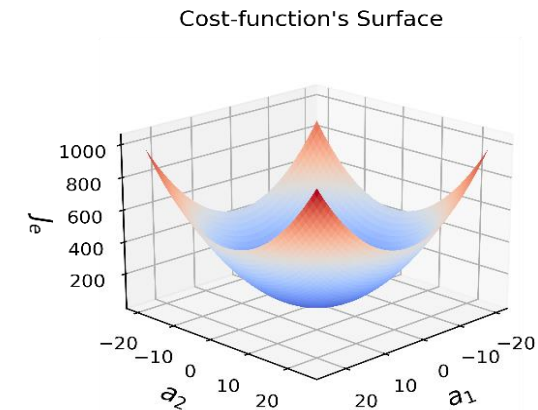
- Discutimos sobre o vetor gradiente.
- Aprendemos dois algoritmos que usam o vetor gradiente para a resolução de problemas de otimização.
 - Gradiente Ascendente para problemas de maximização.
 - Gradiente Descendente para problemas de minimização.
- Vimos as três versões do gradiente descendente e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

Escolha do Passo de Aprendizagem

- Conforme nós vimos, enquanto a **taxa** e a **direção** de **máximo declive** da **função de erro** são determinadas pelo valor negativo do **vetor gradiente** da função, o **passo de aprendizagem** determina o quão grande a atualização dos pesos é feita naquela direção.

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

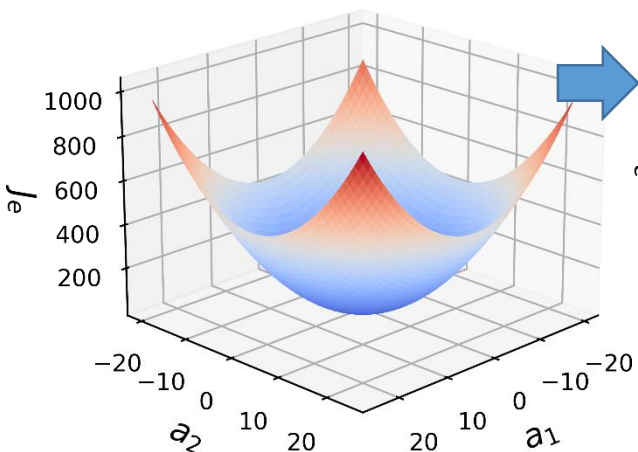
- Portanto, a **escolha do passo de aprendizagem (hiperparâmetro) é muito importante**:
 - Caso ele seja muito pequeno, a convergência do algoritmo será lenta.
 - Exemplo**: com $\alpha = 0.01$, o algoritmo atinge o valor ótimo após mais de 250 épocas.
 - Passos muito curtos, fazem com que o algoritmo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



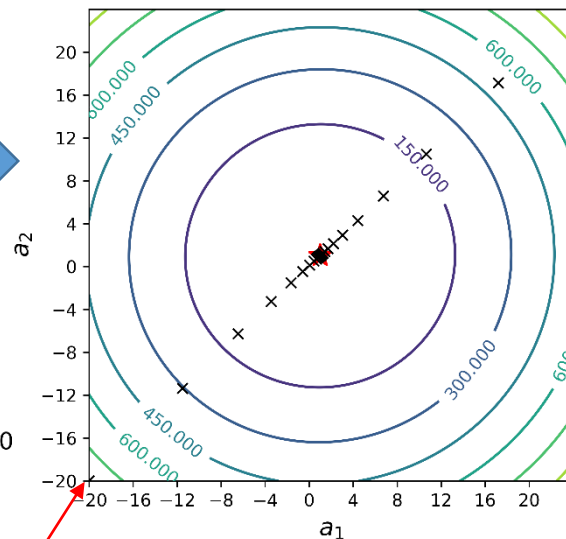
Escolha do Passo de Aprendizagem

- Caso o ***passo de aprendizagem*** seja muito grande, o algoritmo pode nunca convergir.
- Se α 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 α 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



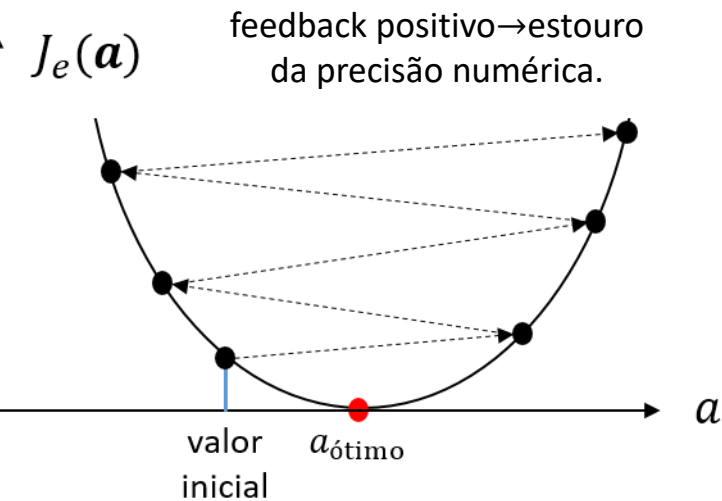
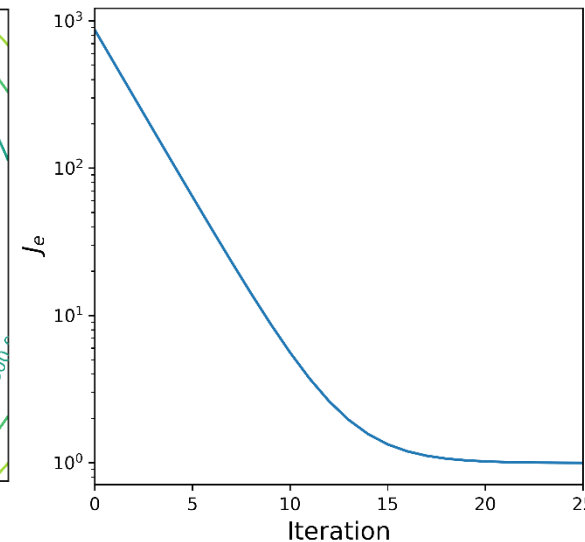
Cost-function's Contour



a inicial

Exemplo #1

Error vs. Iteration number



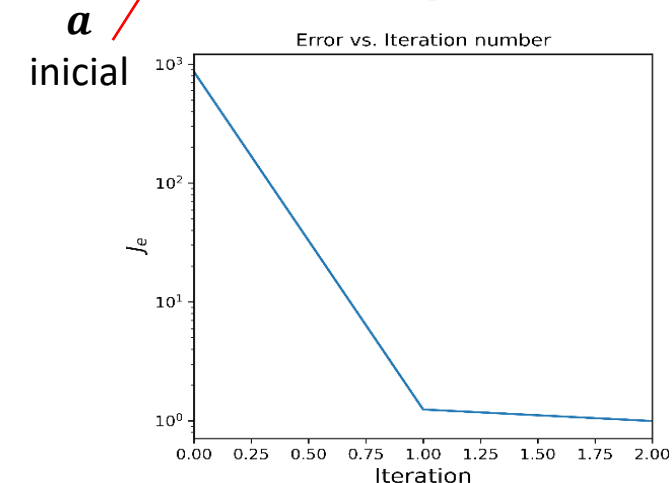
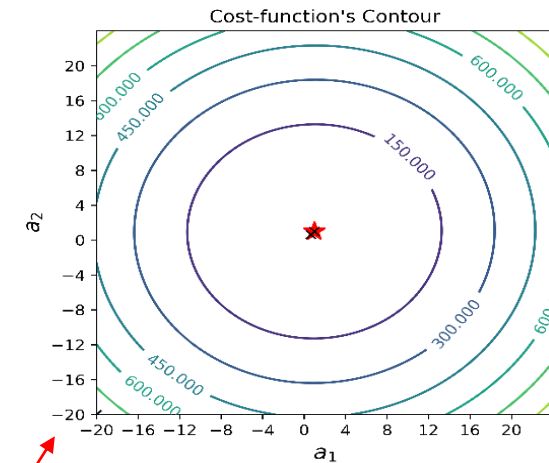
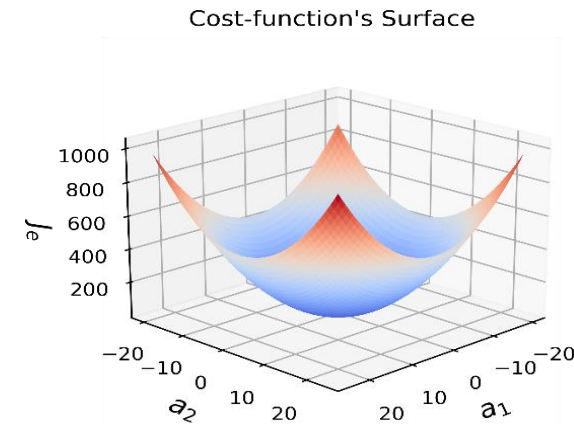
Exemplo #2

Escolha do Passo de Aprendizagem

- Portanto, o valor do ***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**, pois os pesos não são mais praticamente atualizados (mínimo da função foi atingido).
 - 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).

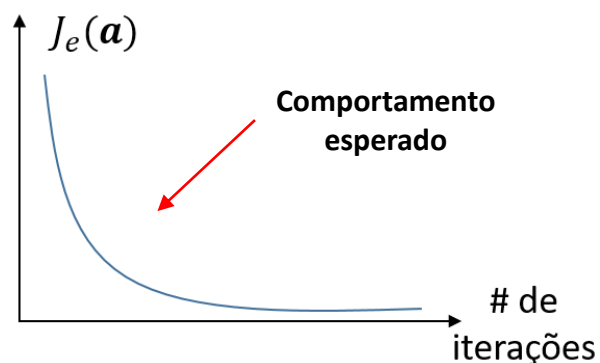


Figura A

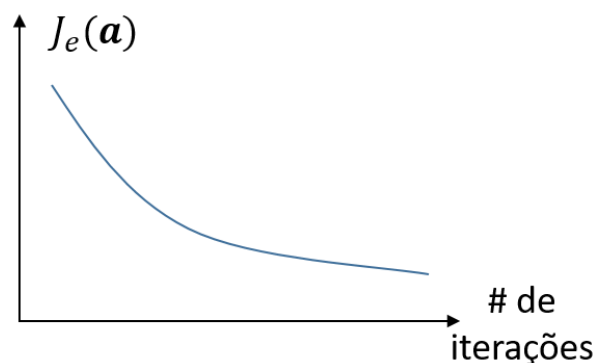


Figura B

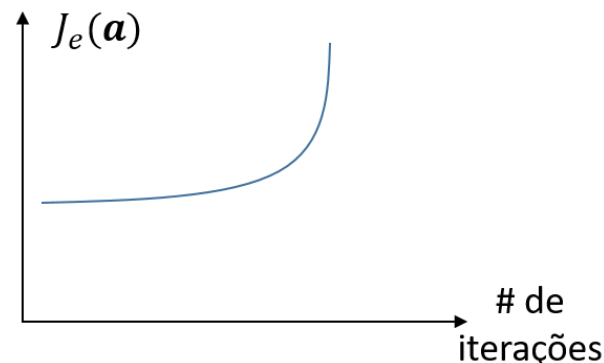


Figura C

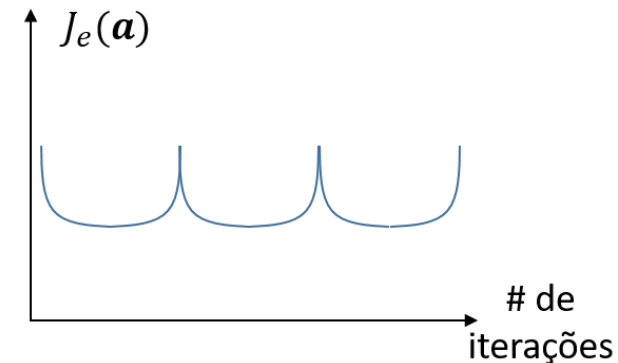
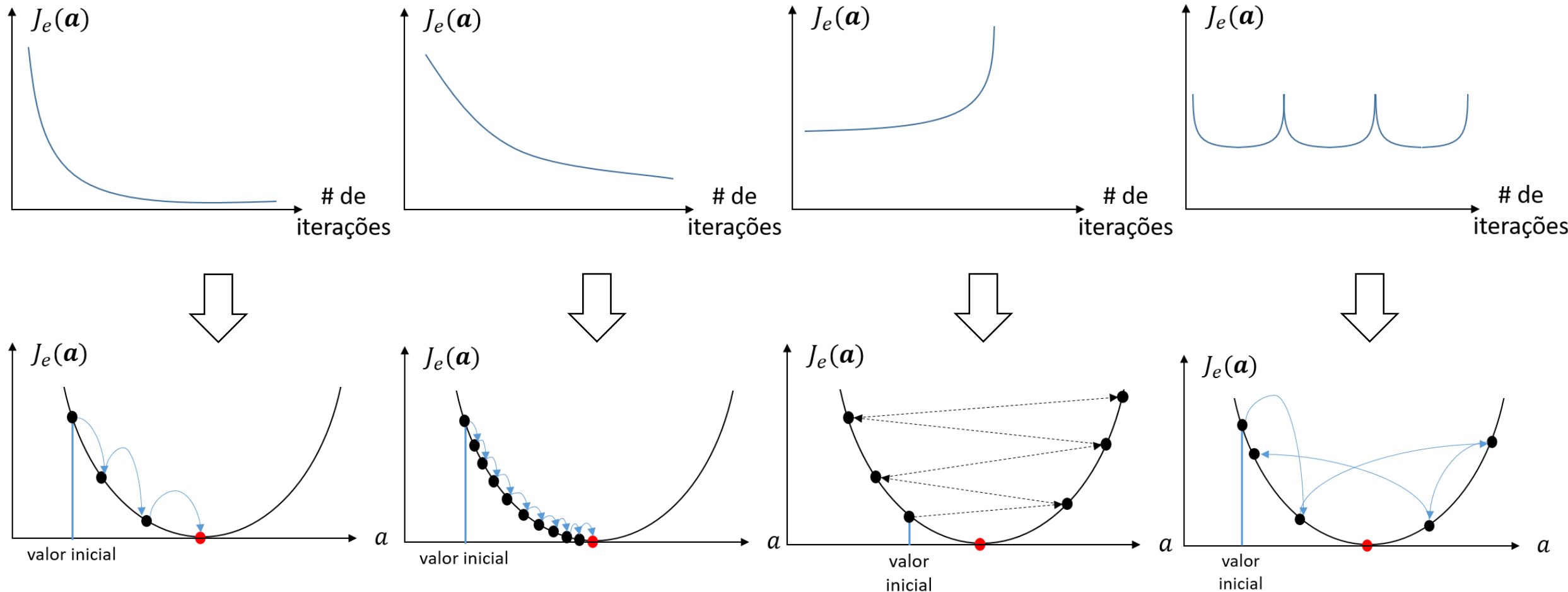


Figura D

Como depurar o algoritmo do GD?



Como configurar o passo de aprendizagem?

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

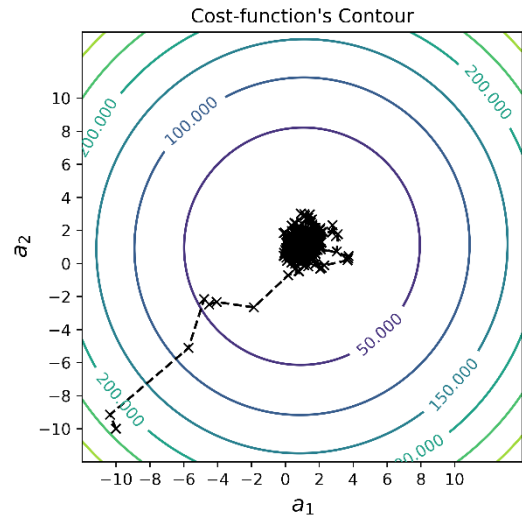
- **Redução programada**: redução de α ao longo do processo de treinamento, ou seja, ao longo das iterações, de forma a garantir a convergência.
- **Termo momentum**: adiciona a ***média do histórico de atualizações*** à equação de atualização dos pesos, tornando as atualizações menos ruidosas, e, conseqüentemente, acelerando a convergência do algoritmo.
- As duas últimas abordagens são usadas com GDE e mini-batch para garantir a convergência.
- **Variação adaptativa**: α é adaptativamente ajustado de acordo com a inclinação da superfície, além disso, usa passos diferentes para cada peso do modelo, os atualizando de forma independente de acordo com a inclinação na direção destes pesos.
 - **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

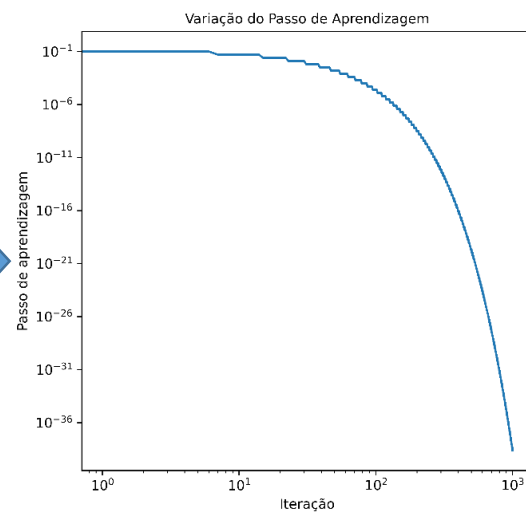
- Os três tipos mais comuns para a **redução programada** de α são:
 - **Decaimento gradual**: também conhecido como **decaimento por etapas** ou **por degraus**. Reduz a taxa de aprendizagem inicial, α_0 , de um fator, τ , a cada número pré-definido de iterações, β . Um valor típico para reduzir a taxa de aprendizado é de $\tau = 0.5$ a cada β de iterações.
 - **Decaimento exponencial**: é expresso pela equação $\alpha = \alpha_0 e^{-kt}$, onde α_0 e k são hiperparâmetros e t é o número da iteração corrente.
 - **Decaimento temporal**: é expresso pela equação $\alpha = \alpha_0 / (1+kt)$ onde α_0 e k são hiperparâmetros e t é o número da iteração corrente.
- Na prática, o **decaimento gradual** é o mais utilizado entre os 3, pois seus **hiperparâmetros** (a taxa de decaimento, τ , e o intervalo para redução, β) 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 α

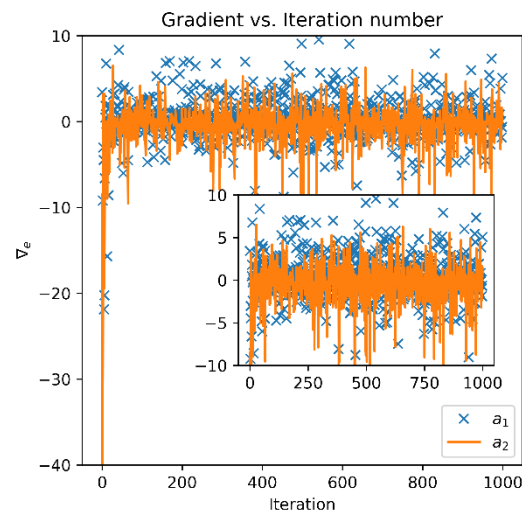
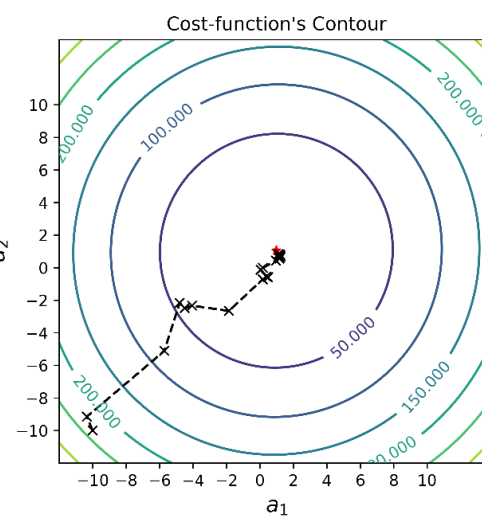
GDE sem redução programada



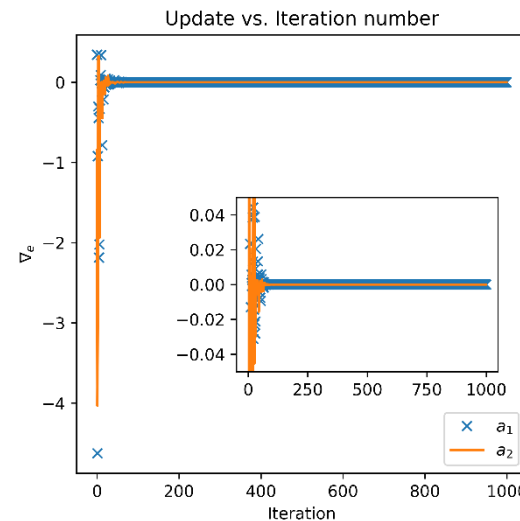
Redução programada



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
```

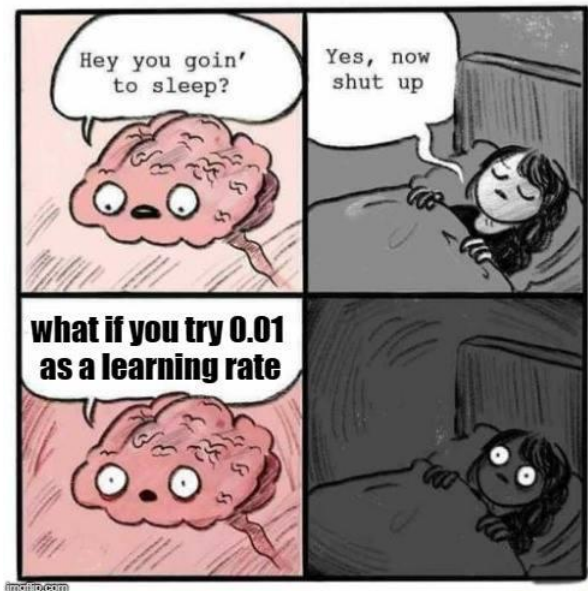


- Exemplo usando GDE com **decaimento gradual**.
- O caminho com **decaimento gradual** também não é regular para o ponto de mínimo.
- Apresenta **algumas mudanças de direção** ao longo do caminho.
- Porém, a **oscilação em torno do mínimo é bastante reduzida** devido à **diminuição gradual** de α .
- Conseguimos visualizar melhor o efeito da redução de α 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.

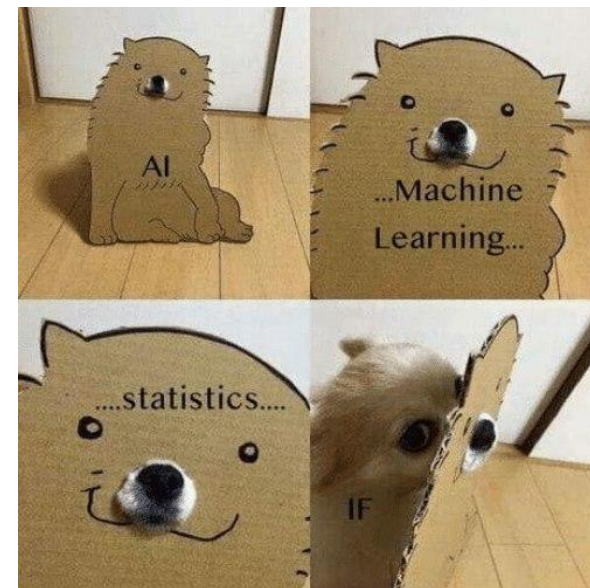
Tarefas

- **Quiz:** “*T319 - Quiz - Regressão: Parte III*” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #4](#).
 - Pode ser acessado através do link acima (Google Colab) ou no GitHub.
 - Vídeo explicando o laboratório: Arquivos -> Material de Aula -> Laboratório #4
 - Se atentem aos prazos de entrega.
 - [Instruções para resolução e entrega dos laboratórios](#).
 - **Laboratórios podem ser resolvidos em grupo, mas as entregas devem ser individuais.**

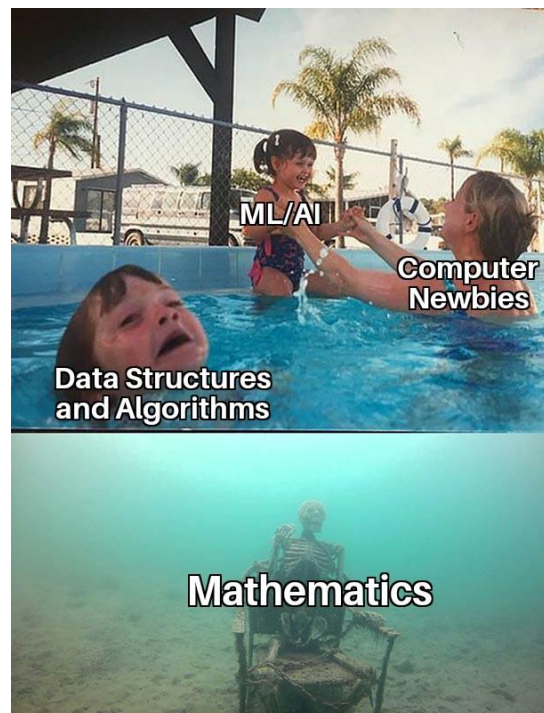
Obrigado!



When someone asks why you never stops talking about machine learning

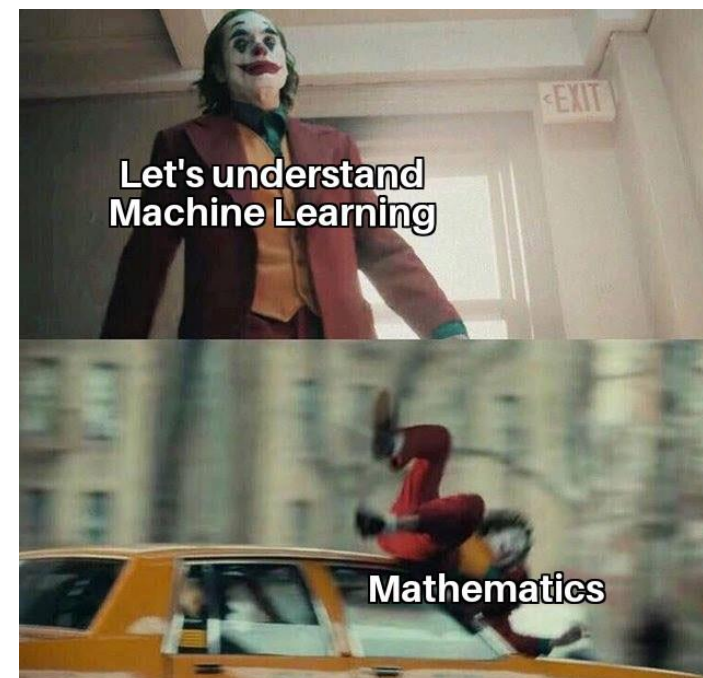


IF IF IF IF IF IF IF IF IF IF WE!

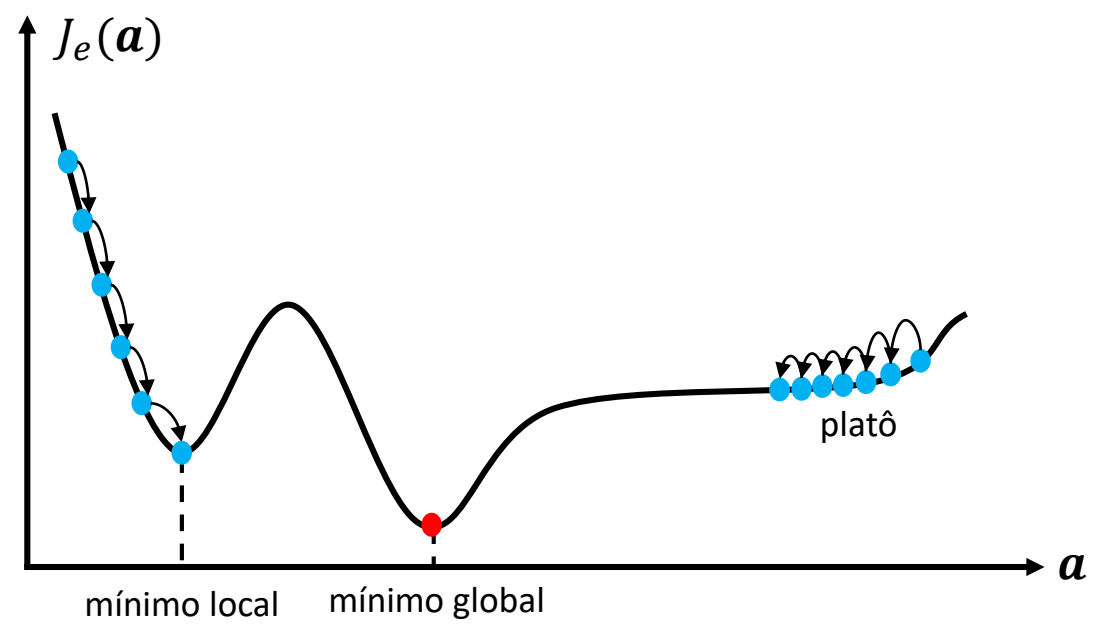


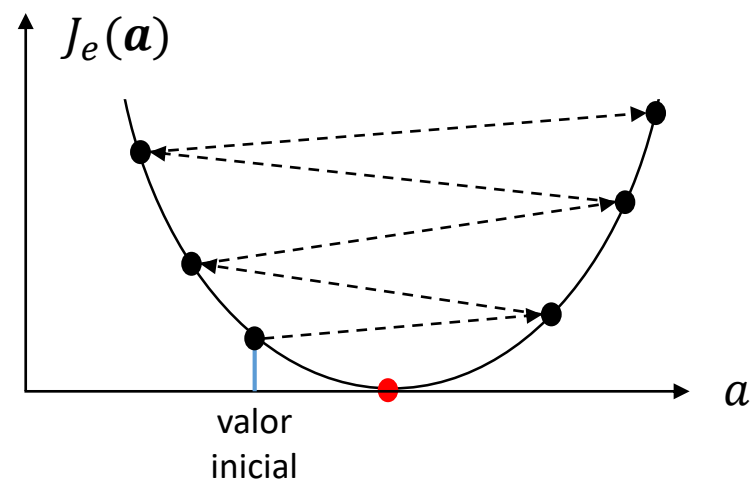
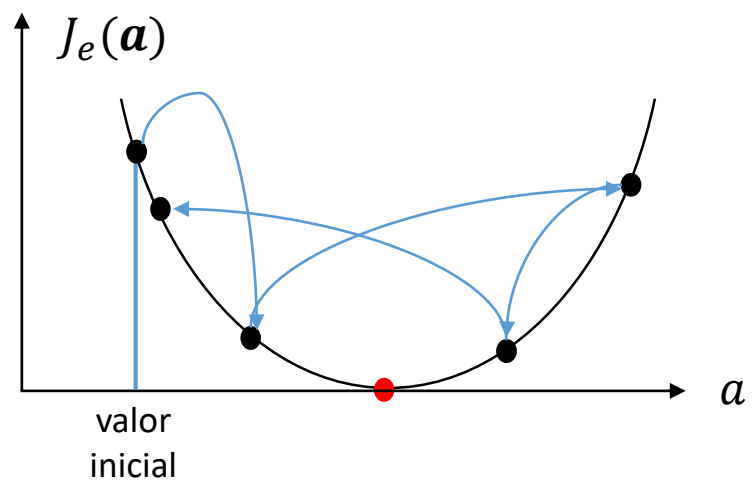
Albert Einstein: Insanity Is Doing the Same Thing Over and Over Again and Expecting Different Results

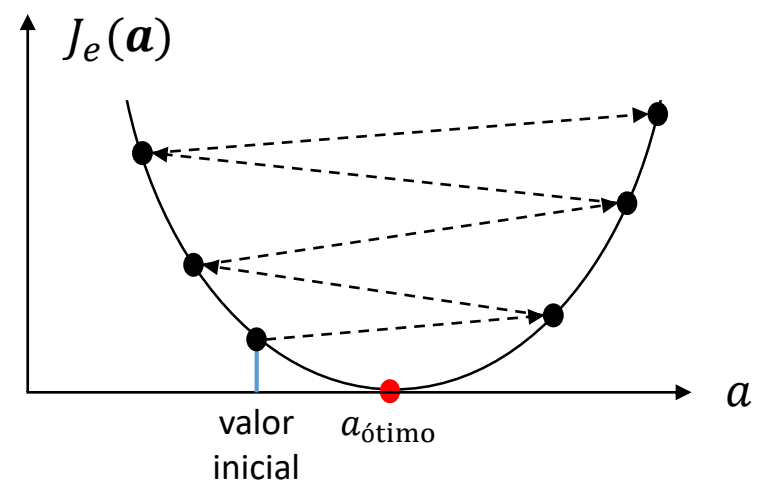
Machine learning:

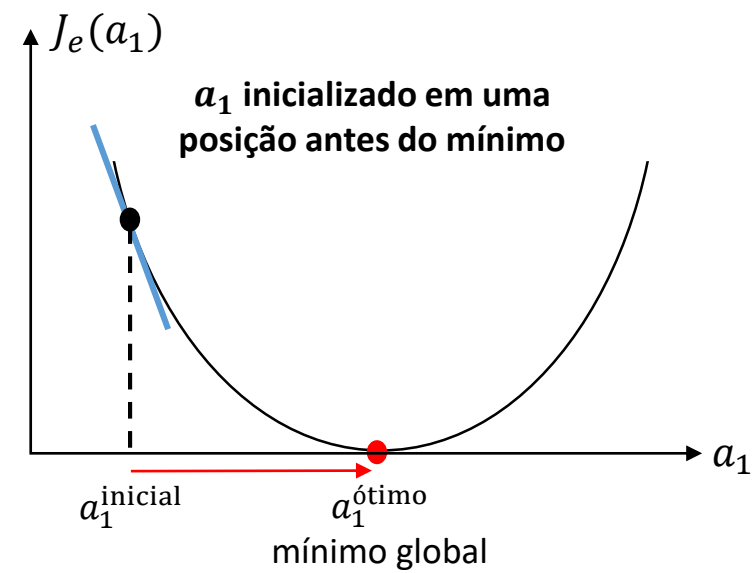


FIGURAS

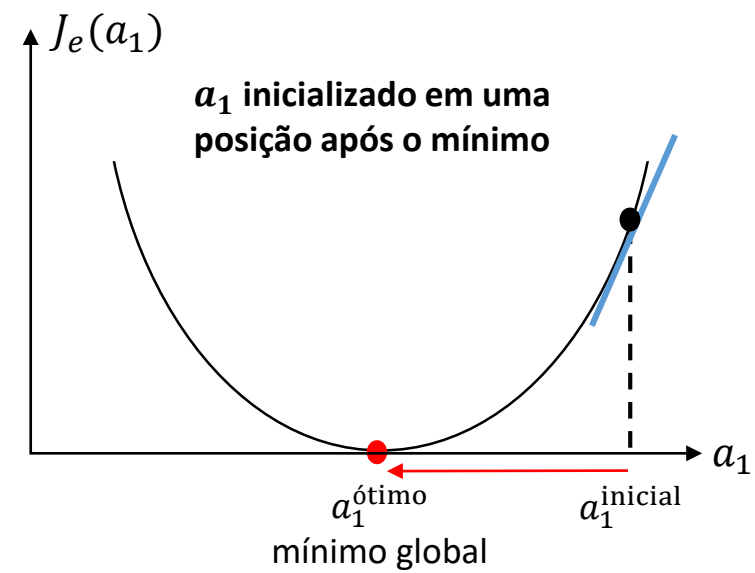




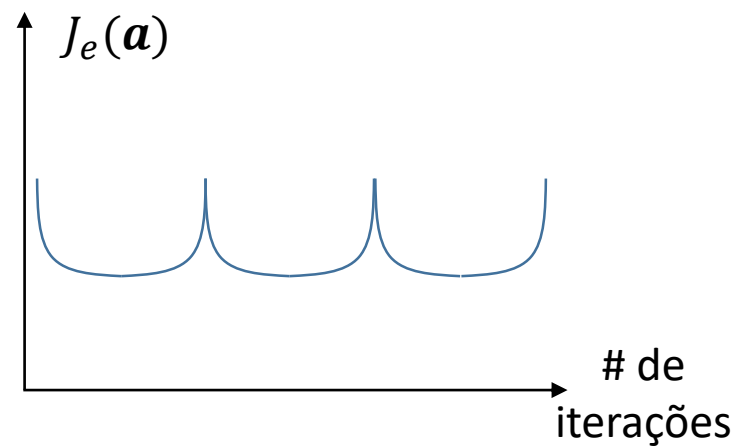
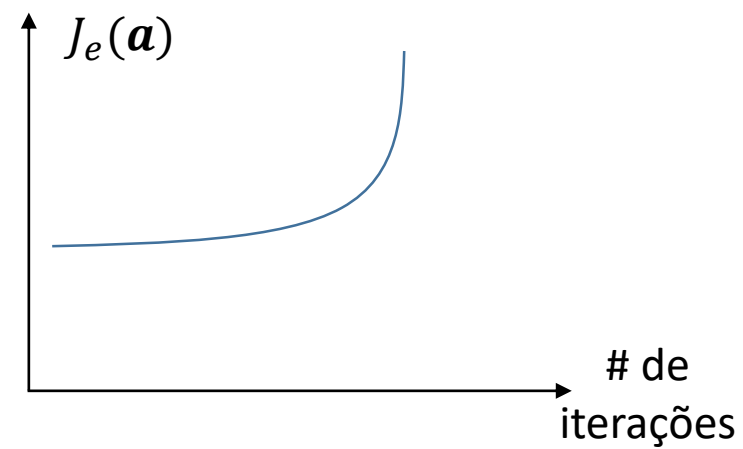
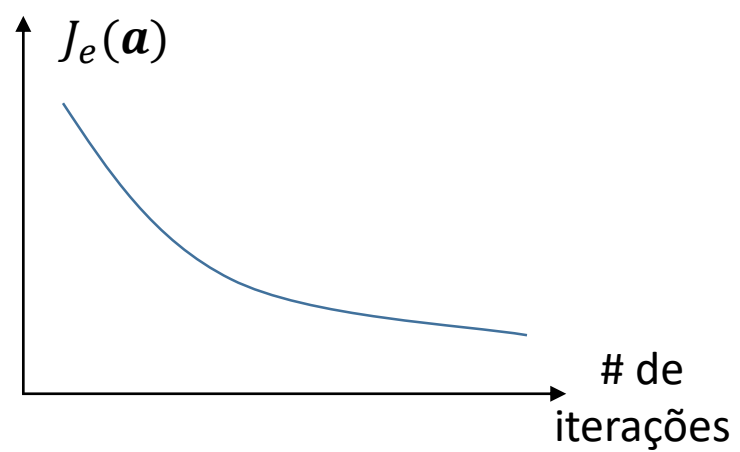
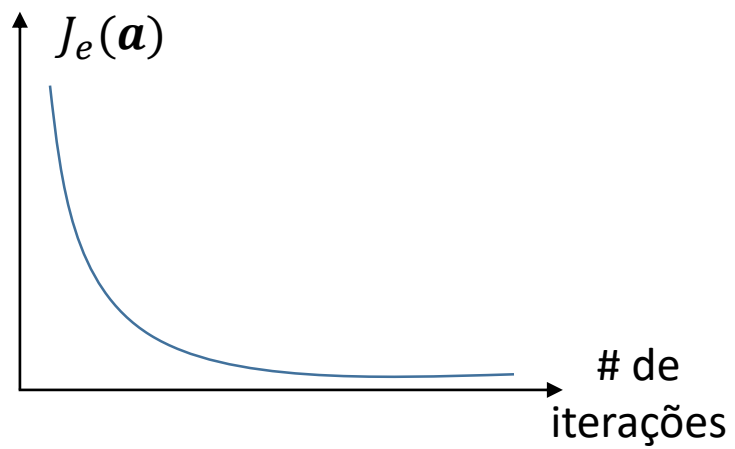


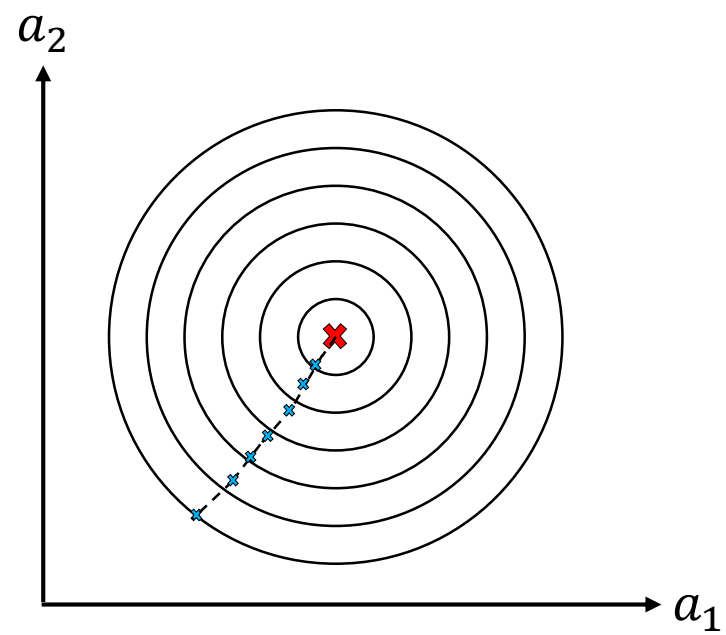
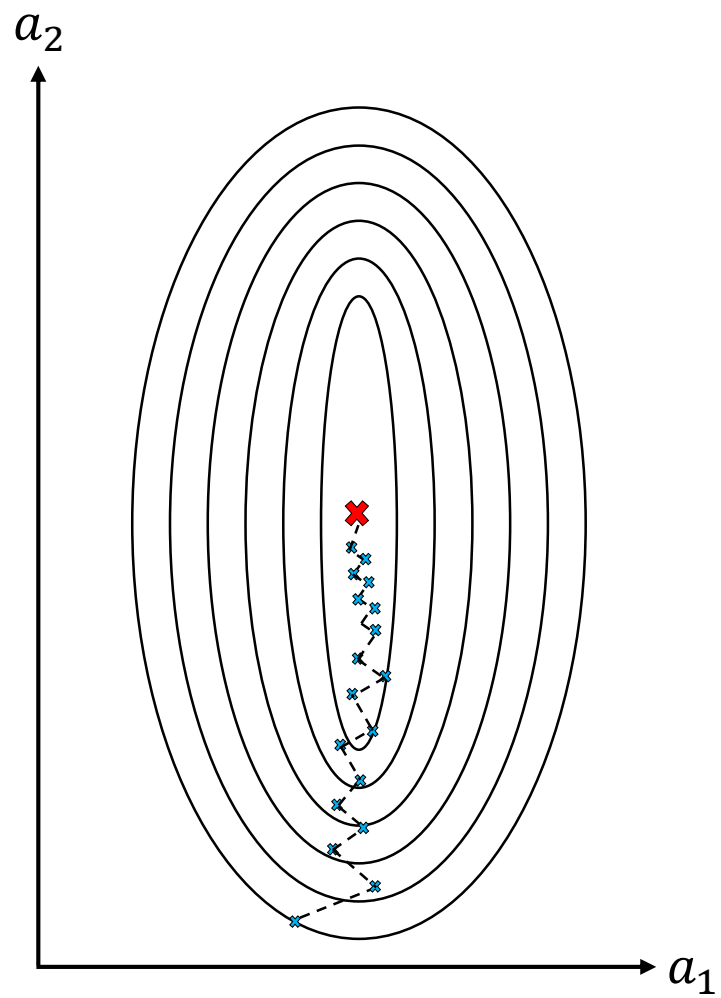


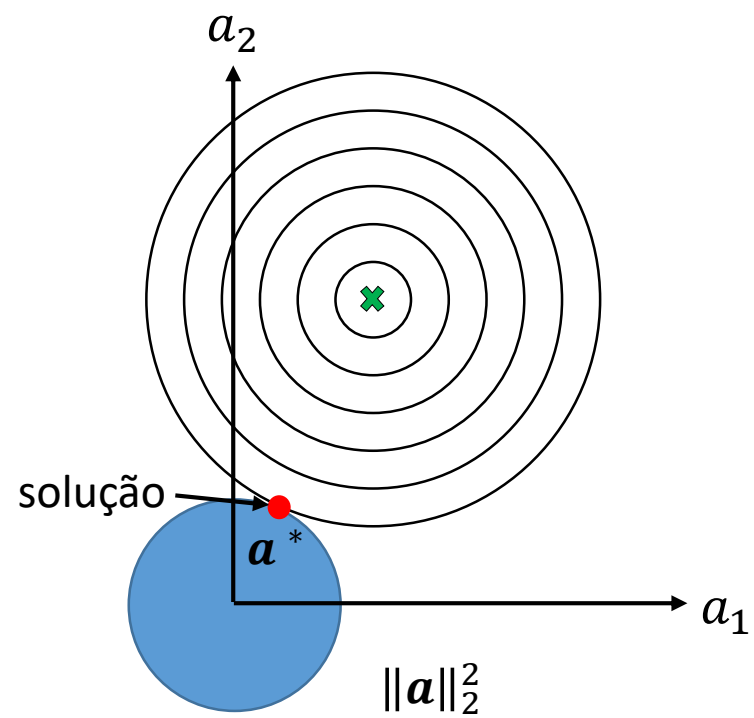
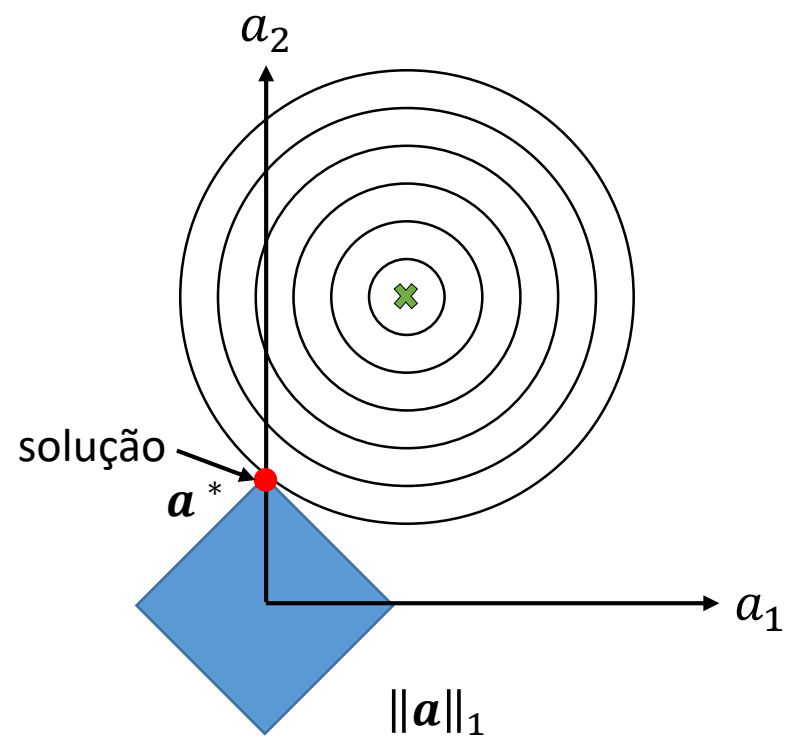
gradiente negativo: $a_1 = a_1^{\text{inicial}} + \alpha \nabla J_e(a_1)$
 a_1 aumenta e se aproxima do mínimo



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

