

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



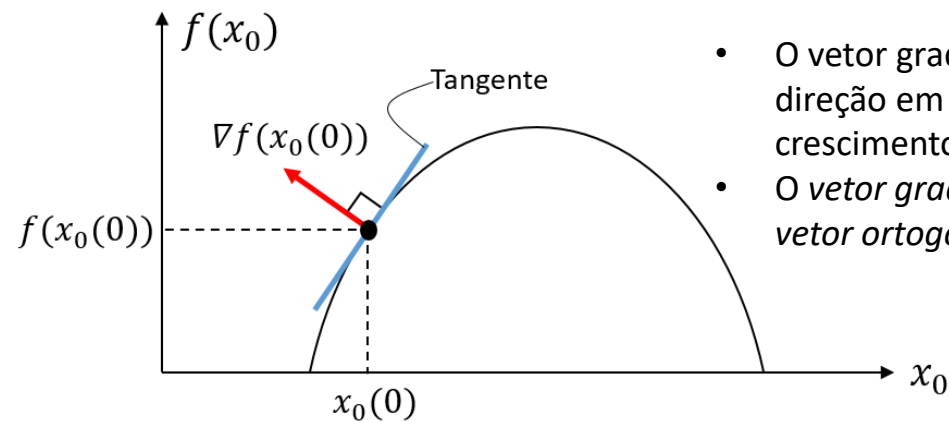
***Inatel***

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

# Recapitulando

- Vimos a motivação por trás da regressão: encontrar funções que aproximem o fenómeno gerador por trás das observações.
- Definimos o problema matematicamente.
- Vimos como resolver o problema da regressão, i.e., encontrar os pesos do modelo, através da equação normal.
- Aprendemos o que é uma superfície de erro.
- Discutimos algumas desvantagens (e.g. complexidade, regressão não-lineares) da equação normal e vimos uma solução para essas desvantagens, a qual discutiremos a seguir.

# Vetor Gradiente



- O vetor gradiente,  $\nabla f$ , indica a magnitude e a direção em que a função,  $f$ , tem a taxa de crescimento mais rápida.
- O vetor *gradiente* em um *ponto específico* é um vetor *ortogonal* ao vetor *tangente* àquele ponto.

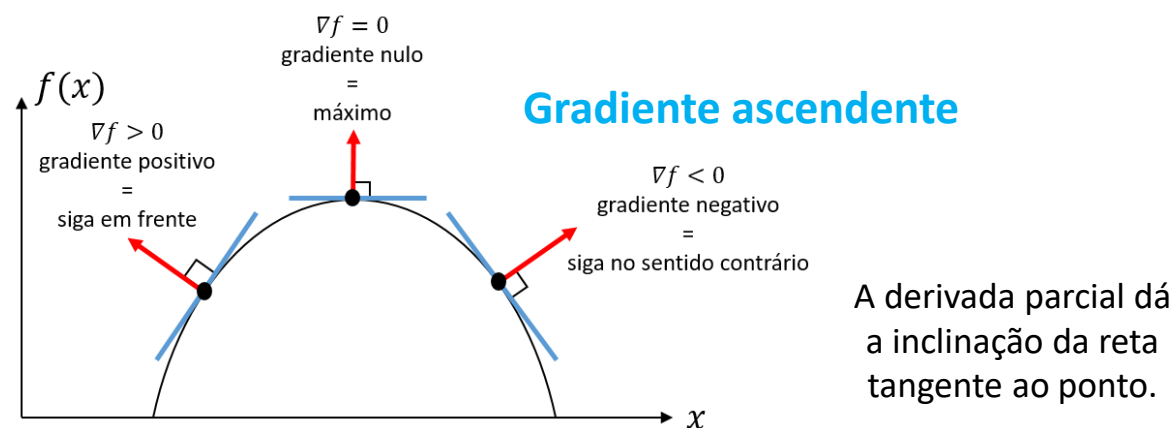
- Vocês se lembram das aulas de cálculo vetorial, onde vocês aprenderam sobre o **vetor gradiente**?
  - **Vetor gradiente** é um vetor que indica a **magnitude** (i.e., **taxa**) e a **direção** na qual, por deslocamento a partir de um ponto específico, obtém-se o maior incremento possível no valor de uma função,  $f$ .
- O **vetor gradiente** de uma função  $f(x_0, x_1, \dots, x_K)$  é definido pela derivada parcial em relação a cada um de seus argumentos  $x_k, k = 0, \dots, K$ :

$$\nabla f(x_0, x_1, \dots, x_K) = \left[ \frac{\partial f(x_0, x_1, \dots, x_K)}{\partial x_0} \quad \frac{\partial f(x_0, x_1, \dots, x_K)}{\partial x_1} \quad \dots \quad \frac{\partial f(x_0, x_1, \dots, x_K)}{\partial x_K} \right]^T.$$

- Cada elemento do **vetor gradiente** indica a magnitude e a direção de máxima variação da função em relação àquele argumento.
- Se imaginem parados em um ponto  $x_0(0), x_1(0), \dots, x_K(0)$  no domínio de  $f$ , o vetor  $\nabla f(x_0(0), x_1(0), \dots, x_K(0))$  diz em qual direção devemos caminhar para aumentar o valor de  $f$  mais rapidamente, ou seja

$$f\left(x_0 + \alpha \frac{\partial f(x_0, x_1, \dots, x_K)}{\partial x_0}, x_1 + \alpha \frac{\partial f(x_0, x_1, \dots, x_K)}{\partial x_1}, \dots, x_K + \alpha \frac{\partial f(x_0, x_1, \dots, x_K)}{\partial x_K}\right) > f(x_0, x_1, \dots, x_K).$$

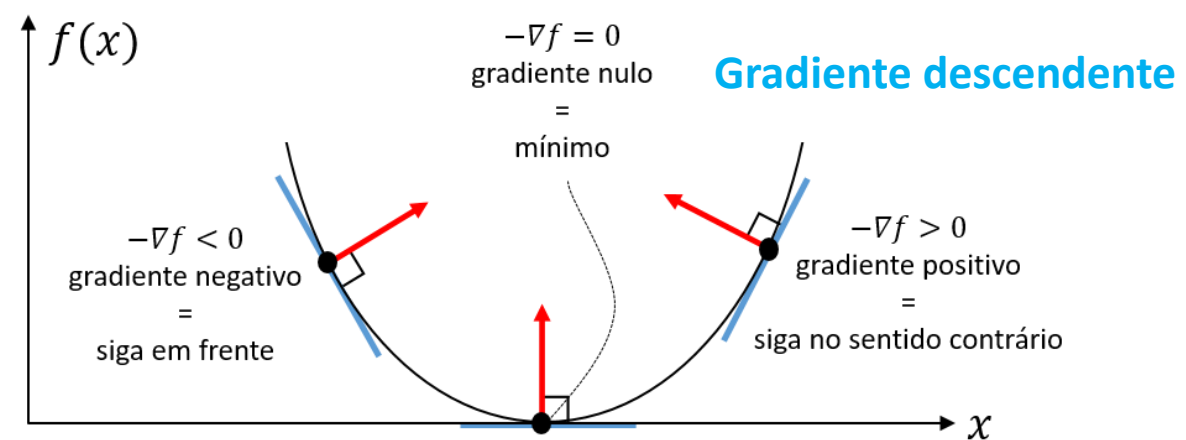
# Gradiente Ascendente



- Em um **ponto específico**, cada elemento do **vetor gradiente** com valor:
  - + (inclinação positiva) significa que o ponto de máximo está à frente.
  - - (inclinação negativa) significa que o ponto de máximo está atrás.
  - 0 (inclinação nula) significa que ponto de máximo foi encontrado.
- Portanto, seguindo na direção indicada pelo **vetor gradiente**, chegamos ao ponto de máximo da função,  $f(x_0, x_1, \dots, x_K)$ .
- Assim, um algoritmo de otimização **iterativo** que siga a direção indicada pelo **vetor gradiente** para encontrar o **ponto de máximo** de uma função  $f(x_0, x_1, \dots, x_K)$  é conhecido como **gradiente ascendente**.
- A cada iteração, calcula-se o **vetor gradiente** da função  $f(x)$  num ponto específico,  $x(l)$ , e atualiza-se os valores dos argumentos da função de tal forma, que a cada **iteração** se tenha:

$$f(x(l+1)) = f(x(l) + \nabla f(x(l))) > f(x(l)), l \geq 0.$$

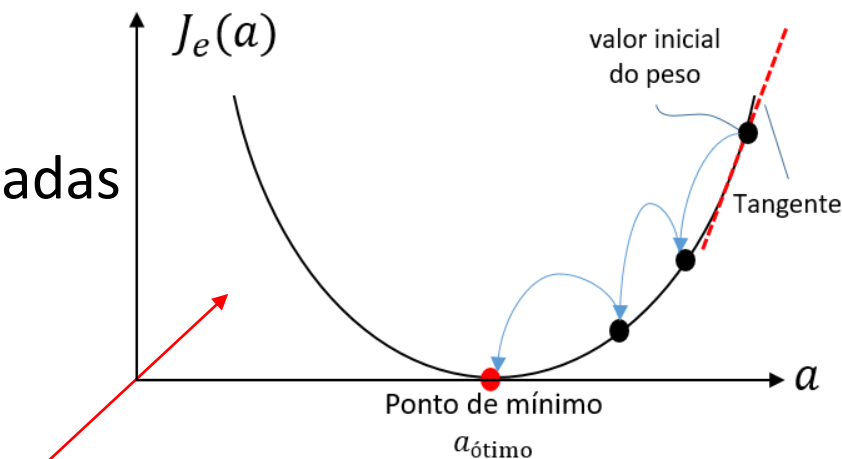
# Gradiente Descendente



- Mas e se formos na direção contrária à da máxima taxa de crescimento, dada pelo **vetor gradiente**,  $\nabla f(x_0, x_1, \dots, x_K)$ , ou seja  $-\nabla f(x_0, x_1, \dots, x_K)$ ?
  - Nesta caso, iremos na direção de **decréscimo** mais rápido da função,  $f(x_0, x_1, \dots, x_K)$ .
- Portanto, um algoritmo de otimização **iterativo** que siga a direção contrária à indicada pelo **vetor gradiente** para encontrar o **ponto de mínimo** de uma função  $f(x_0, x_1, \dots, x_K)$  é conhecido como **gradiente descendente**.
- A cada iteração, calcula-se o **vetor gradiente** da função  $f(x)$  num ponto específico,  $x(l)$ , e atualiza-se os valores dos argumentos da função de tal forma, que a cada **iteração** se tenha:
$$f(x(l+1)) = f(x(l) - \nabla f(x(l))) < f(x(l)), l \geq 0.$$
- Nesta disciplina, como precisamos minimizar o erro, iremos focar neste algoritmo.

# Características do Gradiente Descendente

- Algoritmo de **otimização iterativo** e **genérico**: encontra soluções ótimas para uma ampla gama de problemas.
  - Por exemplo, é utilizado em vários problemas de aprendizado de máquina e otimização.
- Escalona melhor do que o método da **equação normal** para grandes conjuntos de dados.
- É de fácil implementação.
- Não é necessário se preocupar com matrizes mal-condicionadas (determinante próximo de 0, i.e., quase **singulares**).
- Pode ser usado com modelos não-lineares.
- O único requisito é que a **função de erro** seja **diferenciável**.
- Quando aplicado a problemas de **regressão**, a ideia geral é atualizar os pesos,  **$a$** , **iterativamente**, a fim de **minimizar** a **função de erro**, ou seja, encontrar seu **ponto de mínimo**.
- A seguir, veremos como aplicar o algoritmo do **gradiente descendente** ao problema da **regressão linear**.

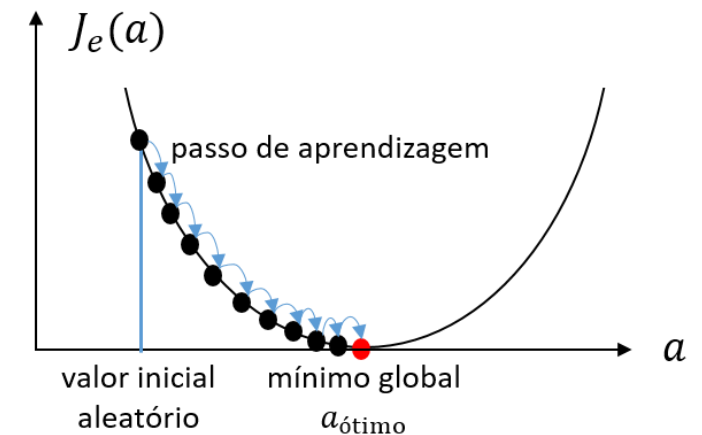


# O Algoritmo do Gradiente do Descendente (GD)

- O algoritmo inicializa os pesos,  $\mathbf{a}$ , em um ponto aleatório do **espaço de pesos** e então, os atualiza na **direção oposta** à do **vetor gradiente** até que algum critério de convergência seja atingido, indicando que um **mínimo local** ou o **global** da **função de erro** foi encontrado.

$\mathbf{a} \leftarrow$  inicializa em um ponto qualquer do espaço de pesos  
**loop** até convergir **ou** atingir número máximo de épocas **do**

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



onde  $\alpha$  é a **taxa/passo de aprendizagem** e  $\frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}}$  é o **vetor gradiente** da **função de erro**, ou seja, a derivada parcial da função em relação ao vetor de pesos,  $\mathbf{a}$ .

- O **passo de aprendizagem** dita o tamanho dos passos/deslocamentos dados na direção oposta à do **gradiente**. Ele pode ser constante ou decair com o tempo.
- Na sequência, veremos como encontrar o **vetor gradiente** da função de erro e implementar o algoritmo do **gradiente descendente**.

# Exemplo

[Exemplo: exemplo\\_regressao\\_linear\\_gradiente\\_descendente.ipynb](#)

Neste exemplo, usaremos uma **função hipótese** com 2 pesos,  $a_1$  e  $a_2$ , sendo  $a_0 = 0$

$$\hat{y}(n) = h(\mathbf{x}(n)) = a_1 x_1(n) + a_2 x_2(n).$$

A função de erro é dada por

$$J_e(\mathbf{a}) = \frac{1}{N} \sum_{n=0}^{N-1} [y(n) - (a_1 x_1(n) + a_2 x_2(n))]^2.$$

Operação da derivada parcial é distributiva.

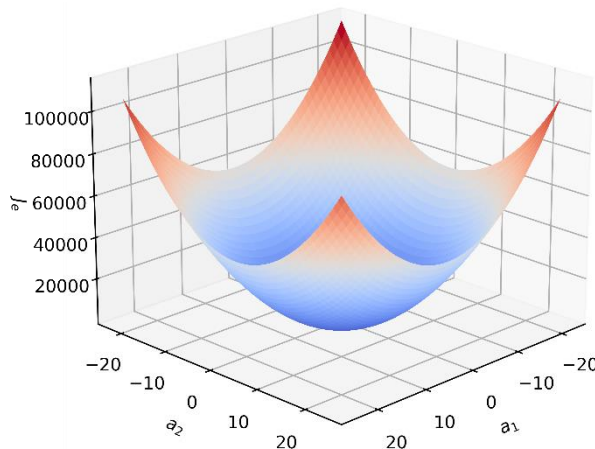
Cada elemento do vetor gradiente é dado por

$$\frac{\partial J_e(\mathbf{a})}{\partial a_k} = \frac{1}{N} \sum_{n=0}^{N-1} \frac{\partial [y(n) - (a_1 x_1(n) + a_2 x_2(n))]^2}{\partial a_k} = -\frac{2}{N} \sum_{n=0}^{N-1} [y(n) - (a_1 x_1(n) + a_2 x_2(n))] x_k(n), \quad k = 1, 2,$$

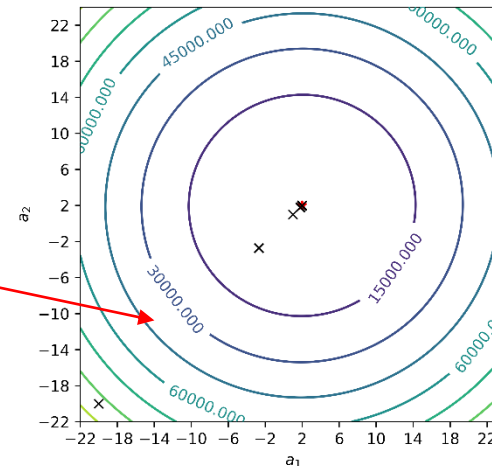
A atualização dos pesos  $a_k, k = 1$  e  $2$  é dada por

$$a_k = a_k - \alpha \frac{\partial J_e(\mathbf{a})}{\partial a_k} \therefore a_k = a_k + \alpha \sum_{n=0}^{N-1} [y(n) - (a_1 x_1(n) + a_2 x_2(n))] x_k(n), \quad k = 1, 2.$$

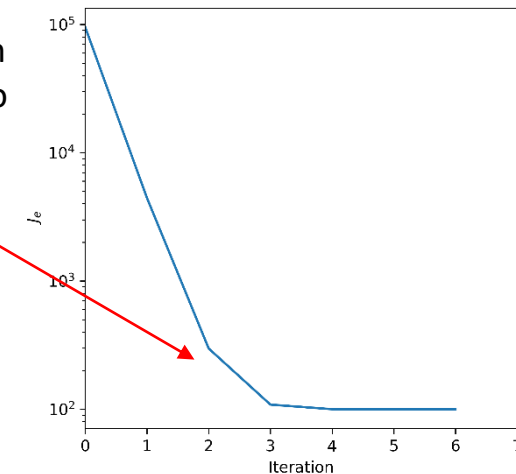
onde o termo  $\frac{2}{N}$  foi absorvido pelo **passo de aprendizagem**,  $\alpha$ .



Superfície de contorno com o caminho feito pelo algoritmo até a convergência.



Curva do EQM em função do número de épocas.





# Versões do Gradiente Descendente

Existem 3 diferentes versões para a implementação do algoritmo do Gradiente Descendente: Batelada, Estocástico e Mini-Batch.

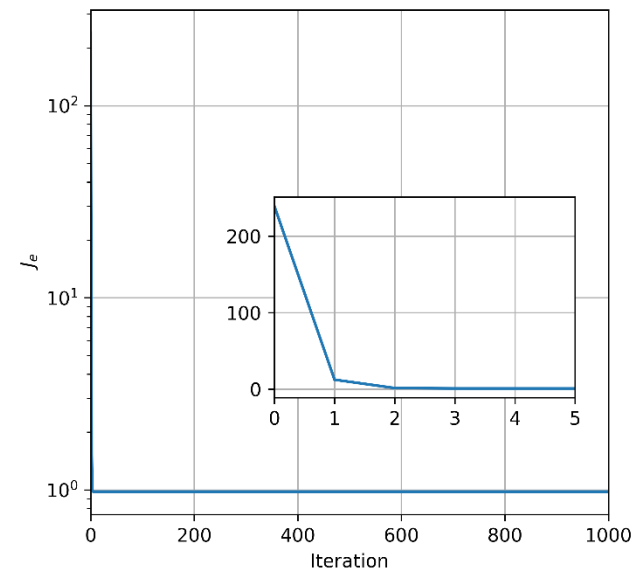
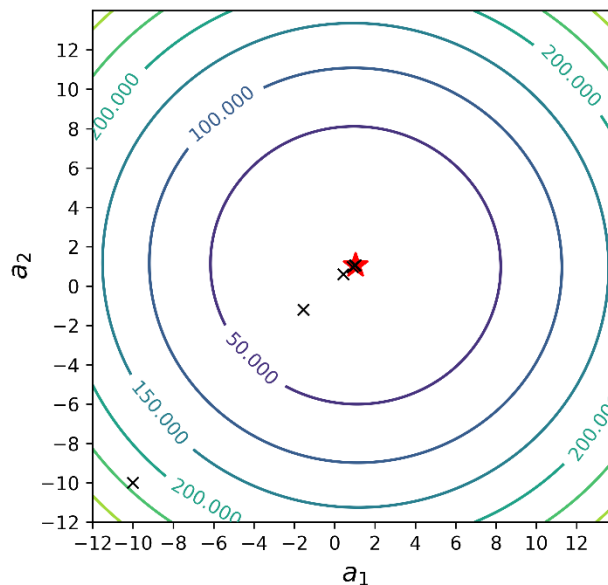
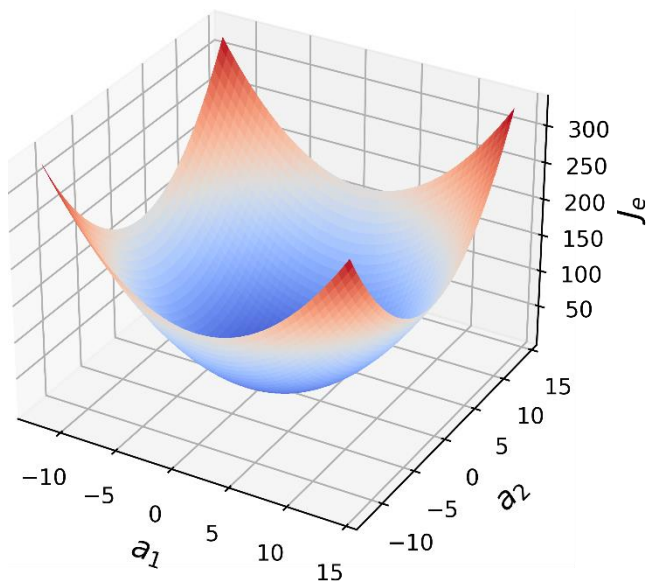
- **Batelada (do inglês *batch*):** a cada iteração (nesse caso, uma época) do algoritmo, ***todos*** os exemplos de treinamento são considerados no processo de treinamento do modelo. Esta versão foi a utilizada no exemplo anterior.

$$a_k = a_k + \alpha \sum_{n=0}^{N-1} [y(n) - (a_1 x_1(n) + a_2 x_2(n))] x_k(n), \quad k = 1, \dots, K$$

## Características:

- Utilizado quando se possui previamente todos os atributos e rótulos de treinamento, ou seja, o conjunto de treinamento.
- **Convergência garantida**, dado que o passo de aprendizagem tenha o tamanho apropriado e se espere tempo suficiente.
- **Convergência pode ser bem lenta**, dado que o modelo é apresentado a todos os exemplos a cada época.

# Características do GD em Batelada



- Segue diretamente para o mínimo global.
- Atinge o mínimo global em aproximadamente 3 épocas.
- Nesse caso específico, segue linha reta entre  $a_1$  e  $a_2$  pois a taxa de decrescimento da superfície de erro é igual para os dois pesos (contornos são circulares).
- Não fica “*oscilando*” em torno do mínimo após alcançá-lo.
- Algoritmo para no mínimo pois o vetor gradiente no ponto ótimo é praticamente nulo.

# Versões do Gradiente Descendente

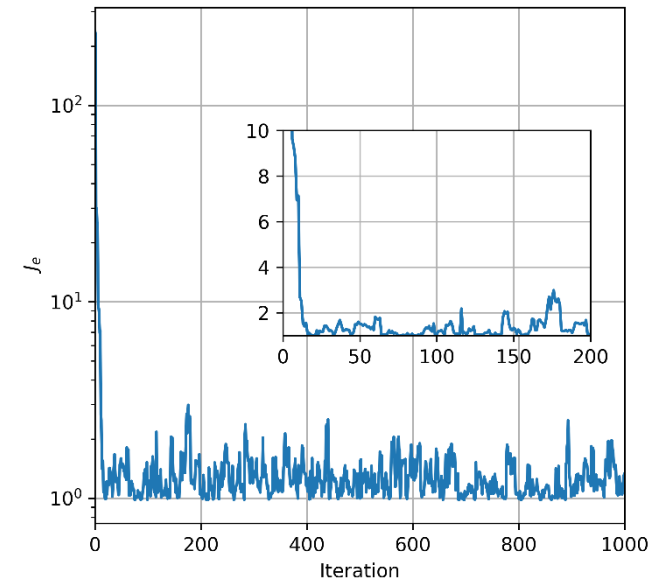
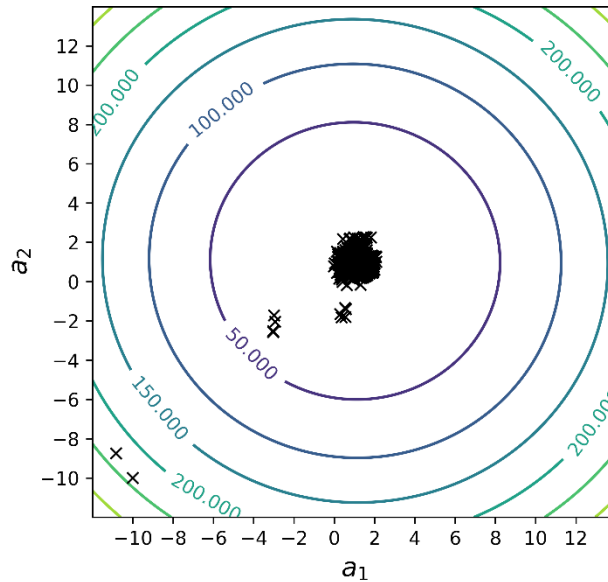
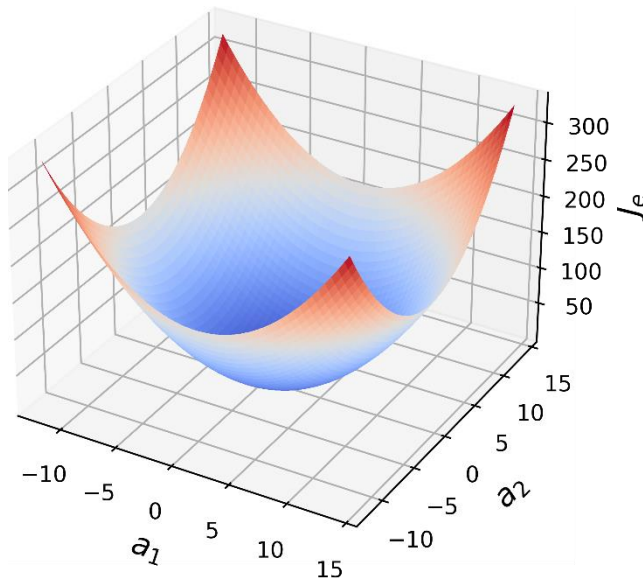
- **Gradiente Descendente Estocástico (GDE):** também conhecido como *online* ou *incremental* (exemplo-a-exemplo). Com esta versão, os pesos do modelo são atualizados a cada novo exemplo de treinamento.

$$a_k = a_k + \alpha [y(n) - (a_1 x_1(n) + a_2 x_2(n))] x_k(n), \quad k = 1, \dots, K$$

## Características:

- **Aproximação estocástica do gradiente:** gradiente calculado com um único exemplo.
- Utilizado quando os atributos e rótulos são obtidos sequencialmente.
- Ou quando o conjunto de treinamento é muito grande.
- Computacionalmente mais rápido e menos custoso em termos de memória que o GD em batelada.
- **Convergência não é garantida** com um passo de aprendizagem fixo. O algoritmo pode oscilar em torno do mínimo sem nunca convergir para o valores ótimos.
- Esquemas de variação do passo de aprendizagem podem ajudar a garantir a convergência.

# Características do GD Estocástico



- Devido à sua natureza estocástica, não apresenta um caminho regular para o mínimo, mudando de direção várias vezes.
- Por aproximar o gradiente com apenas um exemplo, nem sempre irá na direção ideal, porque as derivadas parciais são "ruidosas".
- O algoritmo não converge suavemente para o mínimo: "*oscila*" em torno dele.
- Quando o treinamento termina, os valores finais dos pesos são bons, mas não são ótimos.
- A convergência ocorre apenas na média.
- Tempo de treinamento é menor: com apenas uma época o algoritmo já se aproxima do ponto ótimo.
- Necessita de um esquema de ajuste do passo de aprendizagem,  $\alpha$ , para ficar mais "*comportado*".

# Versões do Gradiente Descendente

- **Mini-batch:** é um meio-termo entre as duas versões anteriores. O conjunto de treinamento é dividido em vários subconjuntos (mini-batches) com elementos aleatórios (i.e., par atributo/rótulo), onde os pesos do modelo são ajustados a cada mini-batch.

$$a_k = a_k + \alpha \sum_{n=0}^{MB-1} [y(n) - (a_1 x_1(n) + a_2 x_2(n))] x_k(n), \quad k = 1, \dots, K$$

onde  $MB$  é o tamanho do mini-batch.

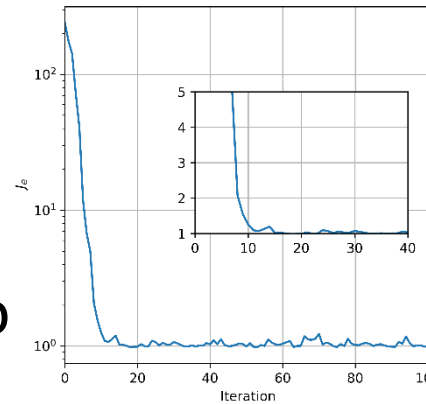
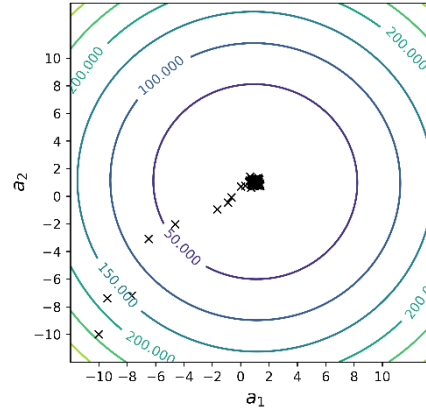
## Características:

- Pode ser visto como uma **generalização** das 2 versões anteriores:
  - Caso  $MB = N$ , então se torna o GD em batelada.
  - Caso  $MB = 1$ , então se torna o GD estocástico.
- Computacionalmente mais rápido do que o GD em batelada, mas mais lento do que o GD estocástico.
- Convergência depende do tamanho do mini-batch.
- Pode usar esquemas de variação do passo de aprendizagem para melhorar a convergência.

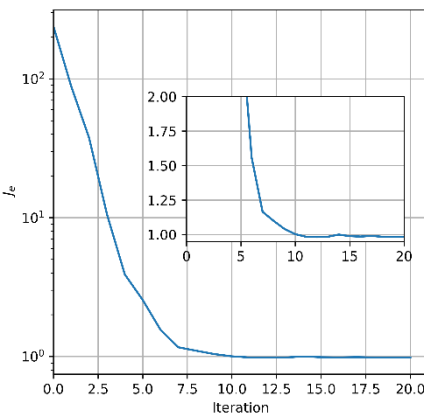
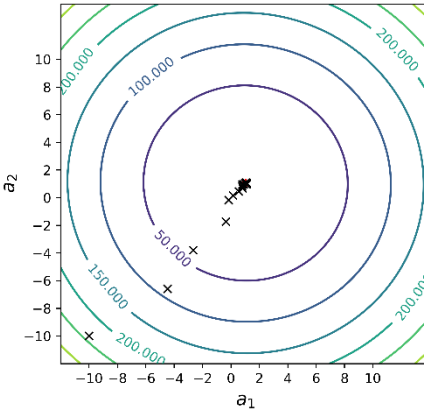
# Características do GD com Mini-Batch

- Progresso menos irregular do que com o GDE, especialmente com mini-batches maiores.
- Como resultado, essa versão oscila menos ao redor do mínimo global do que o GDE.
- Tem comportamento mais próximo do GD em batelada para mini-batches maiores.
- Oscilação em torno do mínimo diminui conforme o tamanho do mini-batch aumenta.
- Pode também ser usado com um esquema de variação do passo de aprendizagem.

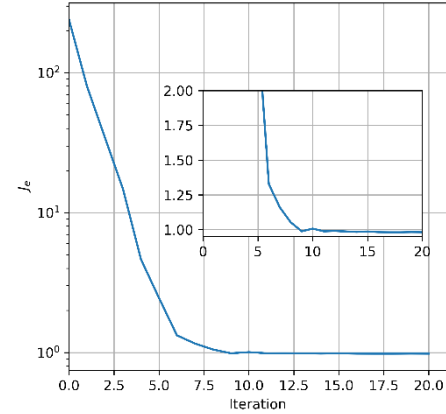
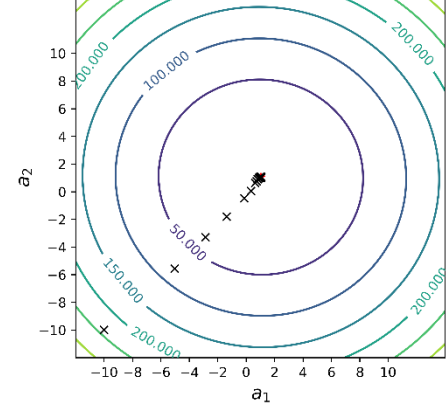
Tamanho do mini-batch: 10



Tamanho do mini-batch: 50



Tamanho do mini-batch: 100



[Exemplo: mini batch gradient descent with figures.ipynb](#)

# Tarefas

- **Quiz:** “*T319 - Quiz - Regressão: Parte II (1S2021)*” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #3](#).
  - 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.**

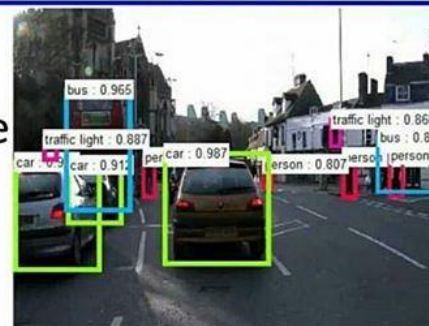
Obrigado!



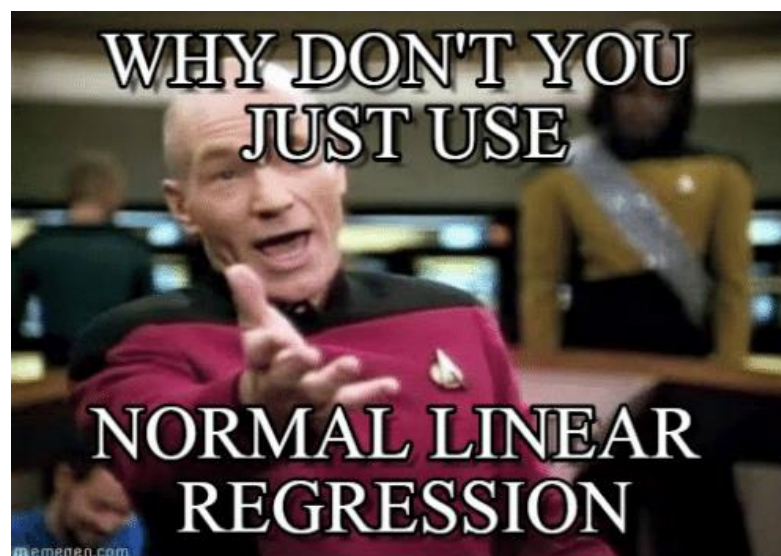
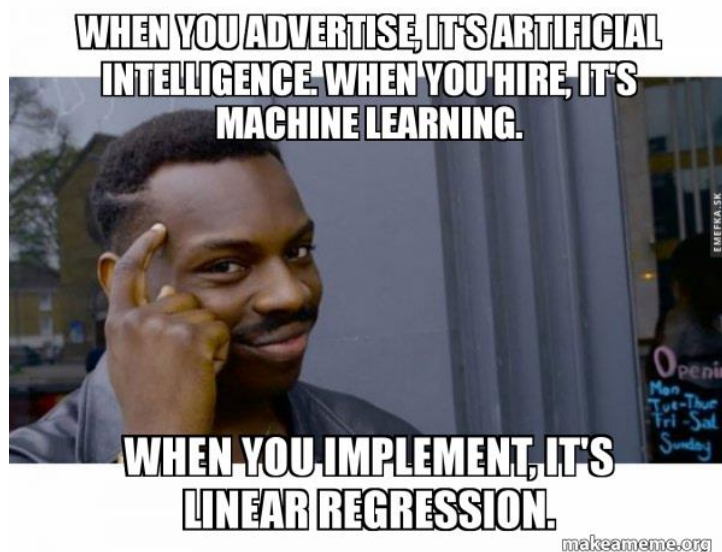
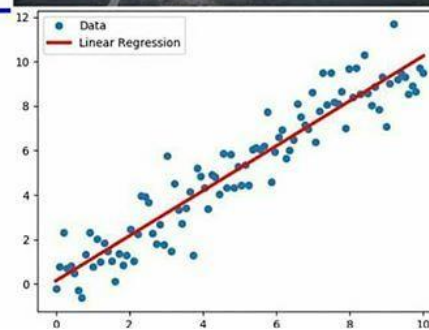


## Online Courses

What they promise  
you will learn



What you actually  
learn



LEARNING ML/DL  
FROM UNIVERSITY

ONLINE COURSES

FROM YOUTUBE

FROM ARTICLES

FROM MEMES

