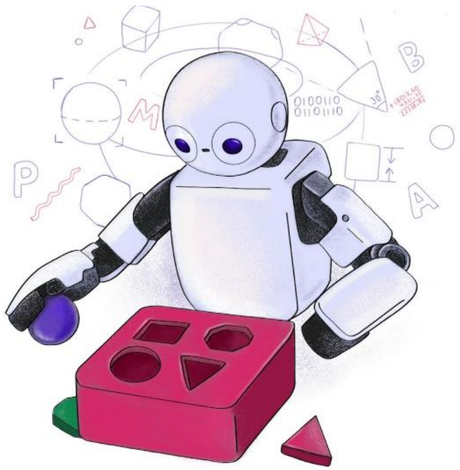
T319 - Introdução ao Aprendizado de Máquina: *Regressão Linear (Parte II)*



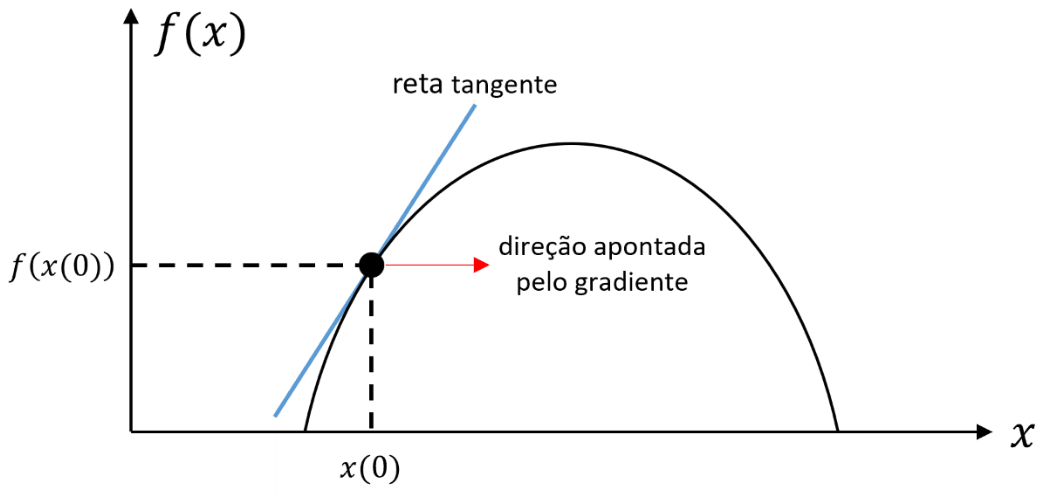
Inatel

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

Recapitulando

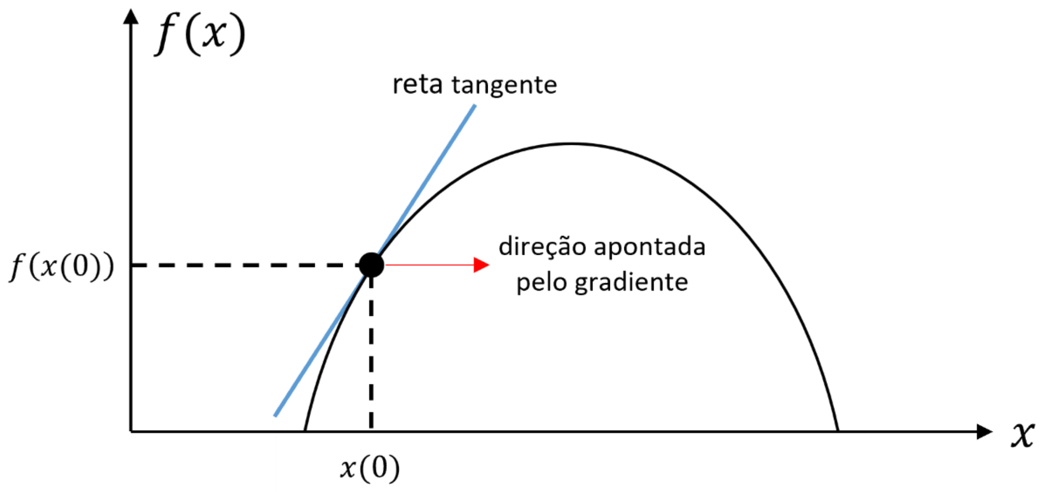
- Vimos a ***motivação*** por trás da ***regressão linear***: encontrar funções que ***aproximem o comportamento*** de um conjunto de amostras (em geral ruidosas).
- Definimos o ***problema matematicamente***.
- Vimos como resolver o problema da regressão, i.e., ***encontrar os pesos do modelo, através da equação normal e visualmente***.
- Aprendemos o que é uma ***superfície de erro***.
- Discutimos algumas ***desvantagens*** (e.g. ***complexidade e regressão não-linear***) da equação normal e vislumbramos uma solução para essas desvantagens, a qual discutiremos a seguir.

Vetor gradiente



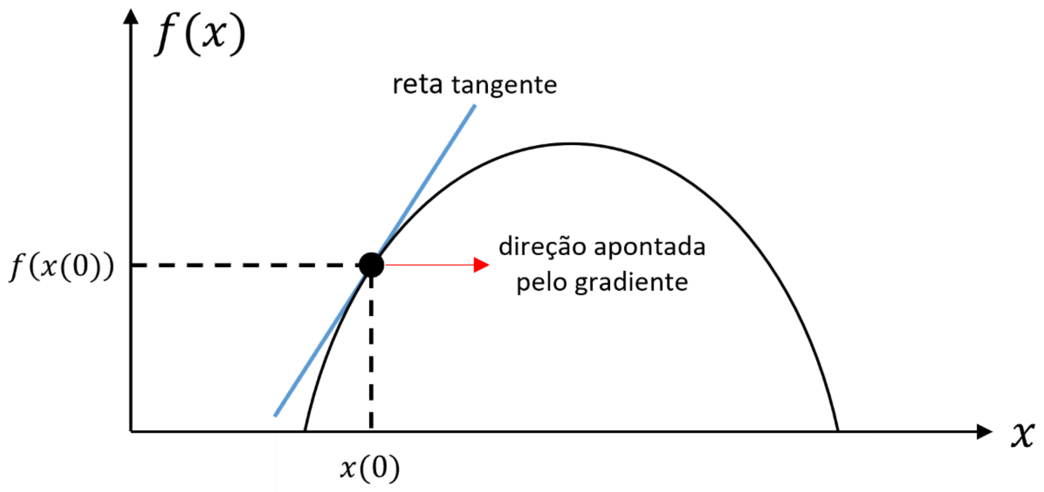
- Vocês se lembram das aulas de cálculo vetorial, onde vocês aprenderam sobre o ***vetor gradiente***?
- Qual ***informação*** ele nos dá sobre uma função?

Vetor gradiente



- O vetor gradiente aponta na ***direção*** em que a função $f(\boldsymbol{x})$ ***cresce mais rapidamente a partir do ponto em que é avaliado***.
- A ***magnitude*** do vetor gradiente indica a ***taxa de crescimento da função*** nessa direção.
    - Quanto maior a magnitude, maior a taxa de crescimento naquela direção.
- Ele diz para que "***lado***" (***aumentar ou diminuir***) os valores dos argumentos, $\boldsymbol{x}$, devem ir para que o ***valor de*** $f(\boldsymbol{x})$ ***seja maior do que o atual***.

Vetor gradiente



Obs.: No caso da função ter apenas um argumento, $f(x)$, o vetor gradiente dá a inclinação de uma ***reta*** tangente ao ponto onde o vetor é calculado.

- O vetor gradiente pode ser também interpretado como a ***inclinação de um plano tangente à função*** no ponto onde ele é calculado.
  - Quanto ***maior o valor absoluto*** do gradiente, ***mais inclinada é a reta tangente*** naquele ponto.
  - Portanto, ***um vetor gradiente igual a 0 indica inclinação nula***.
  - Ou seja, ***a função não varia mais em nenhuma direção***.
  - Onde isso ocorre? Nos ***extremos da função***, ou seja, em seus ***pontos de máximo e de mínimo***.

Vetor gradiente

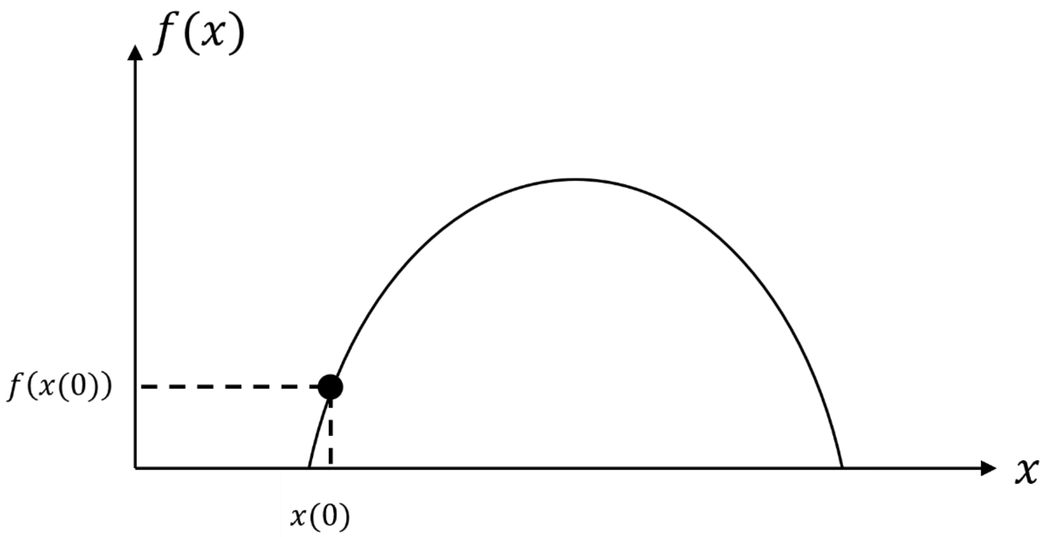
- O ***vetor gradiente*** de uma função com K argumentos, $f(x_1, x_2, \dots, x_K)$, é definido pela ***derivada parcial em relação a cada um de seus argumentos*** $x_k, k = 1, \dots, K$:

$$\nabla f(x_1, x_2, \dots, x_K) = \left[\frac{\partial f(x_1, x_2, \ \dots, x_K)}{\partial x_1} \quad \frac{\partial f(x_1, \ x_2, \dots, x_K)}{\partial x_2} \quad \cdots \quad \frac{\partial f(x_1, x_2, \dots, x_K)}{\partial x_K}\right]^T.$$

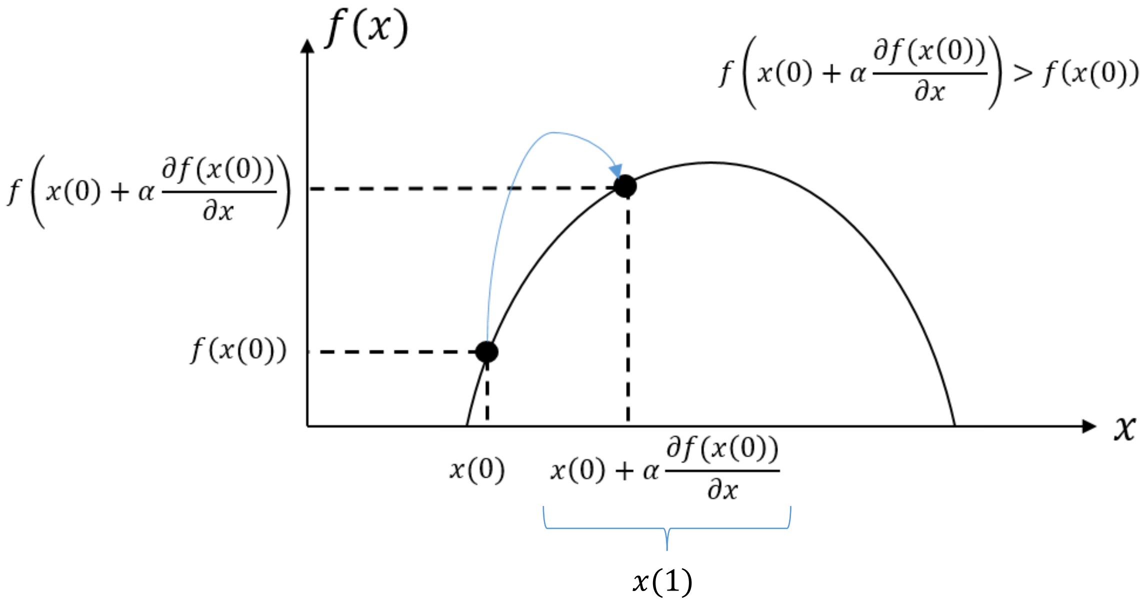
- Notem que o vetor gradiente é representado pelo símbolo *Nabla*, ∇, e é definido como um ***vetor coluna***, com número de elementos igual ao número de argumentos da função.
- **OBS.**: Na sequência, sem perda de generalidade, nós vamos assumir uma função com apenas um argumento, $f(x)$.

O vetor gradiente indica o caminho para o máximo da função

- Imaginem o ponto $x(0)$ com valor $f(x(0))$ na figura abaixo.
- Se quisermos que o valor de $f(x)$ aumente, devemos aumentar ou diminuir o valor de $x(0)$?
- Ou seja, qual ***direção*** devemos seguir para maximizar o valor de $f(x)$?



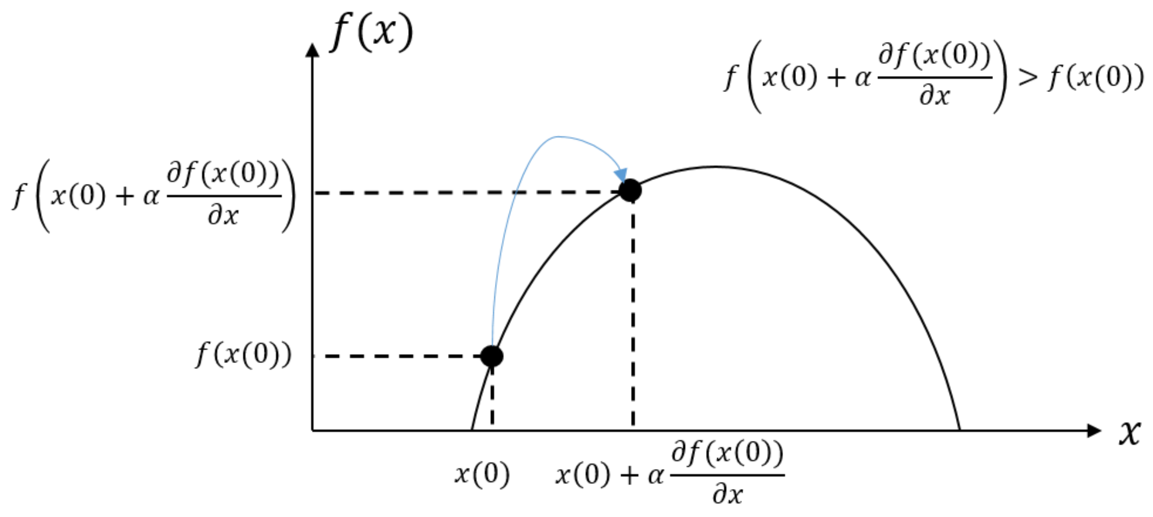
O vetor gradiente indica o caminho para o máximo da função



- O vetor gradiente calculado no ponto $x(0)$, $\nabla f(x(0))$, diz ***em qual direção*** devemos caminhar para ***aumentar o valor da função*** $f(x)$ mais rapidamente.
- Se ***adicionarmos*** uma porcentagem, α, do gradiente ao valor de $x(0)$, teremos que o ***novo ponto***, $x(1)$, terá um valor de $f(x)$ ***maior do que o anterior***, ou seja

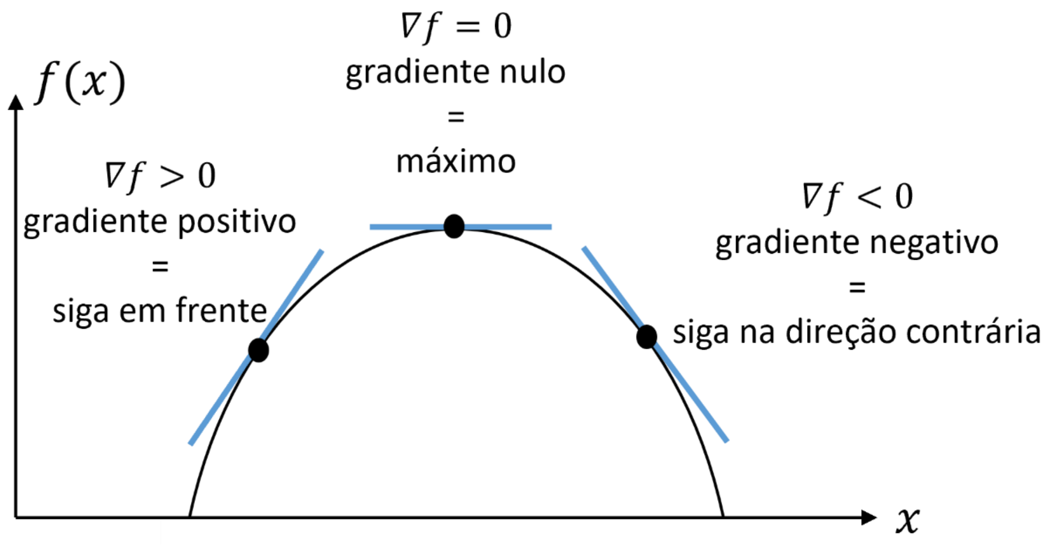
$$f\left(\underbrace{x(0)+\alpha\frac{\partial f(x(0))}{\partial x}}_{x(1)}\right) > f(x(0)).$$

O vetor gradiente indica o caminho para o máximo da função



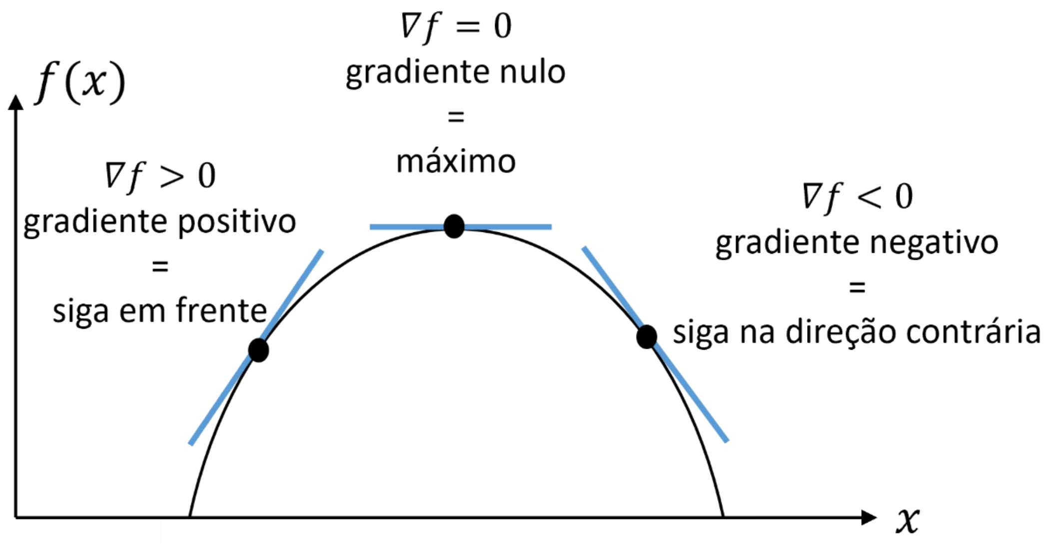
- Se, a cada ***ponto atual***, nós calcularmos o vetor gradiente e adicionarmos uma porcentagem dele ao ponto, teremos um ***novo ponto*** que leva a um ***valor da função maior do que o valor anterior***.
- Portanto, podemos criar um ***procedimento*** que vá ***iterativamente caminhando*** em ***direção ao ponto de máximo da função***.

Algoritmo do gradiente ascendente



- Se o vetor gradiente de $f(x)$ em um ***ponto*** $x(n)$ qualquer dá a ***inclinação da reta tangente à função naquele ponto***.
- Então, nesse ***ponto***, um valor de gradiente:
  - \+ (reta com inclinação positiva) indica que o ***ponto de máximo esta à frente do ponto atual***.
  - \- (reta com inclinação negativa) indica que o ***ponto de máximo está atrás do ponto atual***.
  - 0 (reta com inclinação nula) indica que ***ponto de máximo foi encontrado***.

Algoritmo do gradiente ascendente



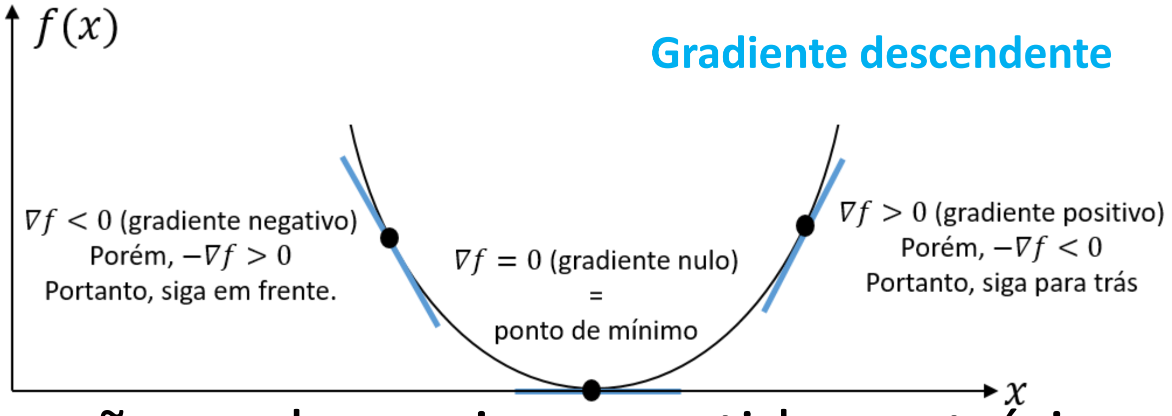
- Portanto, ***seguindo na direção*** indicada pelo ***vetor gradiente***, ***chegamos*** ao ***ponto de máximo da função***.
- Um algoritmo ***iterativo*** de otimização que ***siga a direção*** indicada pelo ***vetor gradiente*** para encontrar o ***ponto de máximo*** de uma função é conhecido como ***gradiente ascendente***.
- Mas como ele funciona?

Algoritmo do gradiente ascendente

- Inicializa-se o argumento $x(0)$ com um valor arbitrário, em geral, aleatório.
- A cada ***iteração***, i, calcula-se o ***vetor gradiente*** da função $f(x)$ no ponto atual, $x(i)$, e atualiza-se o valor do argumento usando uma porcentagem do gradiente, ou seja
$$x(i+1) = x(i) + \alpha \nabla f(x(i)), i \geq 0.$$
- De tal forma, que a cada ***iteração*** se tenha
$$f(x(i+1)) > f(x(i)), i \geq 0.$$
- As iterações se repetem até que o ponto de máximo seja atingido, ou seja, $\nabla f(x(i)) = 0$ e, consequentemente, o argumento x não sofra mais atualizações.
  - Chamamos de ***convergência*** quando o valor da função $f(x)$ fica constante.

Mas lembrando do problema da regressão linear, nós não queremos ***encontrar o ponto de mínimo da função de erro*** ao invés do seu máximo?

Gradiente Descendente



- Para encontrarmos o mínimo de uma função, podemos ir no sentido contrário ao apontado pelo vetor gradiente, ou seja, $-\nabla f(x_0, x_1, \dots, x_K)$.
- Mas e se formos na direção contrária a da máxima taxa de crescimento, dada pelo ***vetor gradiente***, $\nabla f(x_0, x_1, \dots, x_K)$, ou seja $-\nabla f(x_0, x_1, \dots, x_K)$?
  - Neste caso, iremos na direção de ***decrescimento*** mais rápido da função, $f(x_0, x_1, \dots, x_K)$.
- Portanto, um algoritmo de otimização ***iterativo*** que siga a direção contrária a 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***, l, calcula-se o ***vetor gradiente*** da função $f(\boldsymbol{x})$ num ponto específico, $\boldsymbol{x}(l)$, e atualiza-se os valores dos argumentos da função de tal forma, que a cada ***iteração***, se tenha o valor de $f(\boldsymbol{x})$ ***menor*** do que o anterior:

$$f(\boldsymbol{x}(l+1)) = f\big(\boldsymbol{x}(l) - \alpha \nabla f(\boldsymbol{x}(l))\big) < f(\boldsymbol{x}(l)), l \geq 0.$$

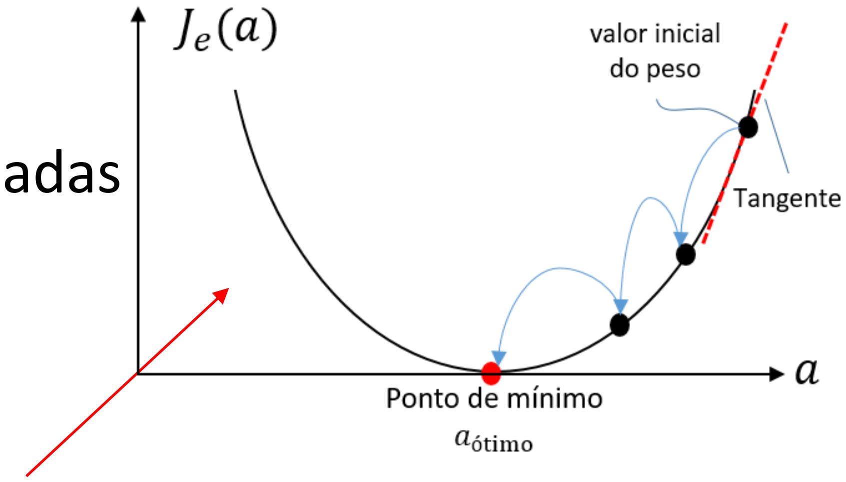
- Nesta disciplina, como queremos minimizar o erro, iremos focar neste algoritmo.

Observação

- Os conceitos vistos até agora foram apresentados para uma função com um único argumento, $f(x)$.
- Porém todos eles são válidos para funções com vários argumentos, $f(x_0, x_1, \dots, x_K)$.

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***.



A cada nova iteração de atualização (seta azul), o peso se aproxima de seu valor ótimo, consequentemente, minimizando o erro.

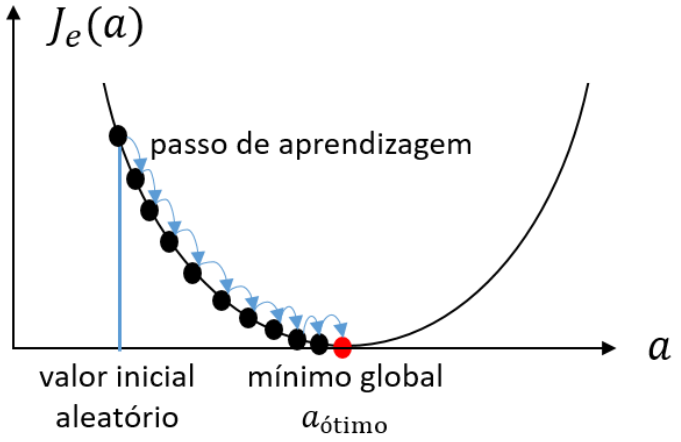
O Algoritmo do Gradiente do Descendente (GD)

- O algoritmo inicializa os pesos, $\boldsymbol{a}$, em um *ponto aleatório* do ***espaço de pesos*** e, então, aplica a ***regra de atualização dos pesos*** até que o algoritmo convirja (e.g., erro pequeno entre duas iterações subsequentes) ou o número máximo de iterações seja atingido.

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

$$\boldsymbol{a} \leftarrow \boldsymbol{a} - \alpha \frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}}$$ (*regra de atualização dos pesos*)

Os pesos são atualizados na direção oposta a do vetor gradiente.



onde $\alpha > 0$ é a ***passo de aprendizagem*** e $\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}}$ é o ***vetor gradiente***, $\nabla J_e(\boldsymbol{a})$, da ***função de erro***, ou seja, a derivada parcial da função em relação ao vetor de pesos, $\boldsymbol{a}$.

- O ***passo de aprendizagem*** dita o tamanho dos passos (i.e., deslocamentos) dados na direção oposta a do ***gradiente***.
- O ***passo de aprendizagem*** pode ser constante ou pode decair com o tempo à medida que o processo de aprendizado prossegue.
- Na sequência, veremos como encontrar o ***vetor gradiente*** da ***função de erro*** e como implementar o algoritmo do ***gradiente descendente***.

Exemplo

[Exemplo: exemplo_regressao_linear_gradiente_descendente.ipynb]()

- Usaremos uma ***função hipótese*** com 2 pesos, a_1 e a_2

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

- A função de erro é dada por

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

- Cada elemento do vetor gradiente é dado por

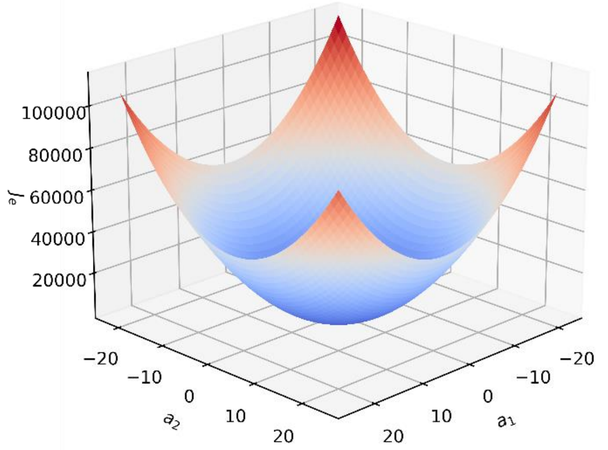
$$\frac{\partial J_e(\boldsymbol{a})}{\partial a_k} = -\frac{2}{N}\sum_{n=0}^{N-1}\left[y(n) - \left(a_1 x_1(n) + a_2 x_2(n)\right)\right] x_k(n)\,, k = 1{,}2$$

- A ***equação de atualização*** dos pesos $a_k, k = 1$ e 2 é dada por

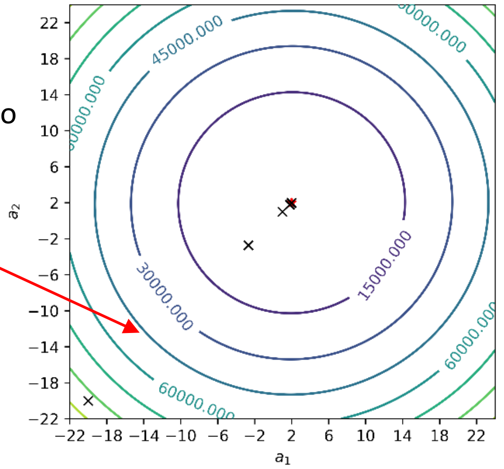
$$a_k = a_k - \alpha \frac{\partial J_e(\boldsymbol{a})}{\partial a_k}$$

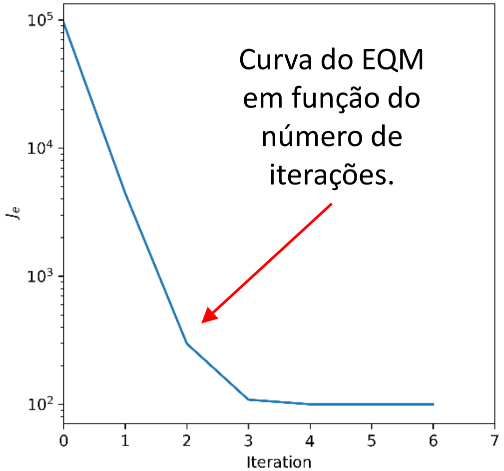
$$a_k = a_k + \alpha \sum_{n=0}^{N-1}\left[y(n) - \left(a_1 x_1(n) + a_2 x_2(n)\right)\right] x_k(n),\ k = 1{,}2.$$

- Por ser constante, o termo $2/N$ pode ser absorvido por α.
- Forma matricial da equação de atualização: $\boldsymbol{a} = \boldsymbol{a} - \alpha \boldsymbol{X}^T(\boldsymbol{y} - \hat{\boldsymbol{y}})$



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





Versões do Gradiente Descendente

- Existem três versões diferentes para a implementação do algoritmo do gradiente descendente:
  - **Batelada**: usa todas as amostras do conjunto de treinamento para calcular o vetor gradiente (versão que acabamos de ver).
  - **Estocástico**: usa apenas uma amostra do conjunto de treinamento para estimar o vetor gradiente.
  - **Mini-Batch**: usa um subconjunto de amostras do conjunto de treinamento para estimar o vetor gradiente.

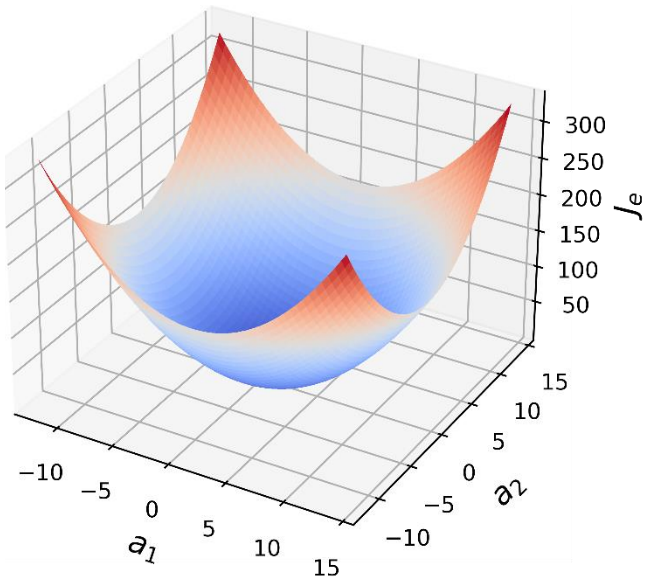
Versões do Gradiente Descendente

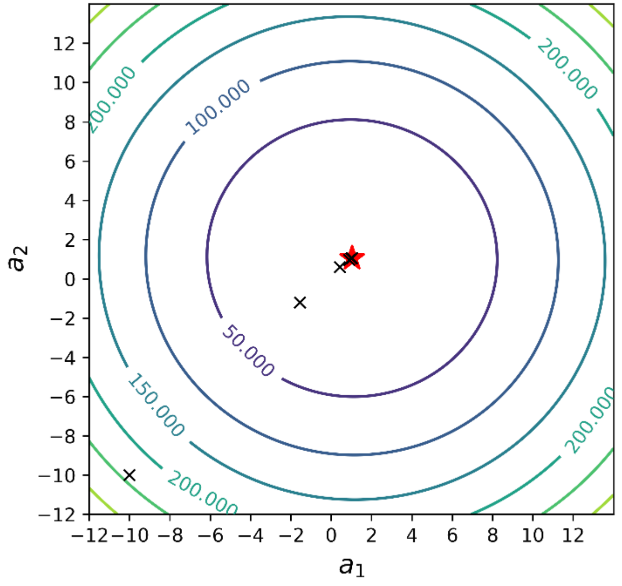
- **Batelada** (do inglês ***batch***): a cada época do algoritmo, ***todos*** os exemplos de treinamento são considerados no processo de treinamento do modelo. Esta versão foi a utilizada no exemplo anterior.

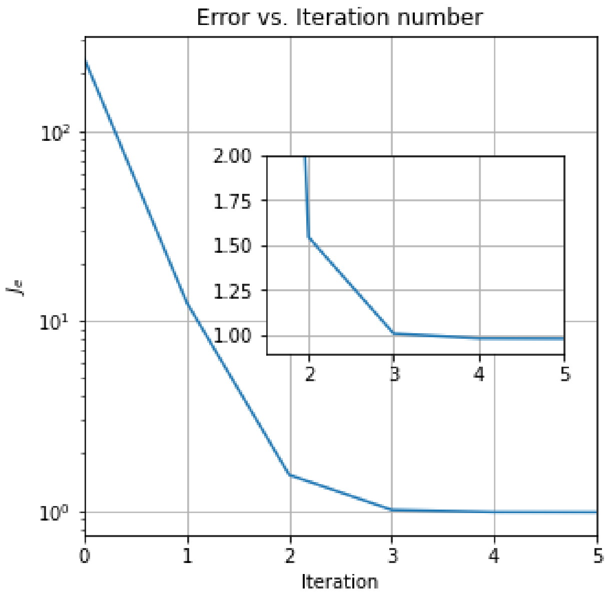
$$a_k = a_k + \alpha \boxed{\sum_{n=0}^{N-1}} \left[y(n) - h\big(\boldsymbol{x}(n)\big)\right] x_k(n),\ \ k = 1, \dots, K$$

- **Características**:
  - Utilizado quando se possui previamente todos os atributos, $\boldsymbol{x}$, e rótulos, y, 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.
  - Se o conjunto de treinamento for muito grande, pode ser impossível treinar o modelo, pois ele consome muitos recursos computacionais (CPU e memória).

Características do GD em Batelada







- Segue diretamente, sem alterar a direção, para o mínimo global.
- Atinge o mínimo global em aproximadamente 3 épocas.
- Nesse caso específico, segue uma 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, pois o vetor gradiente neste ponto é praticamente nulo.

Exemplo: batch_gradient_descent_with_figures.ipynb

Versões do Gradiente Descendente

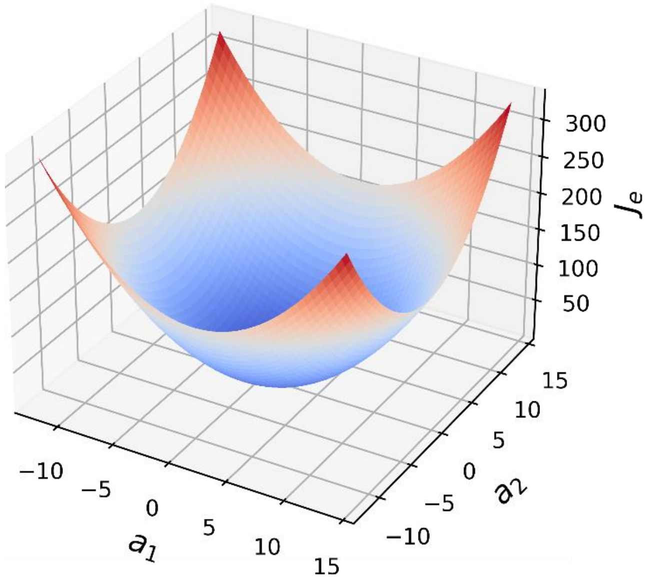
- **Gradiente Descendente Estocástico (GDE)**: também conhecido como ***online*** ou ***incremental*** (exemplo-a-exemplo). Com esta versão, os ***pesos do modelo são atualizados a cada novo exemplo de treinamento***.

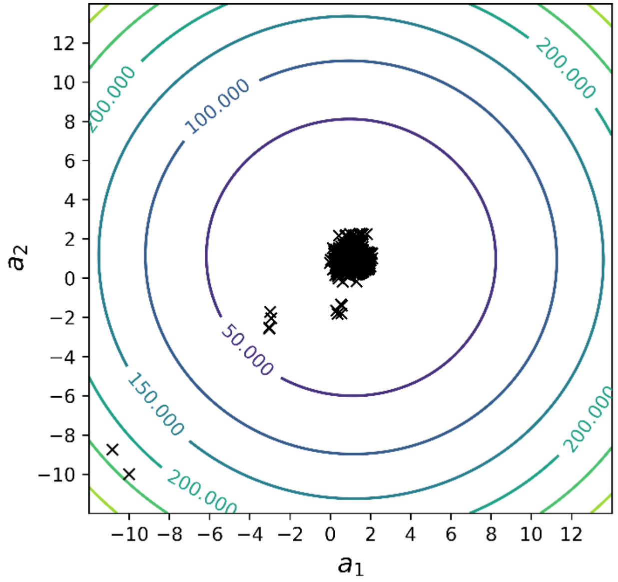
$$a_k = a_k + \alpha\left[y(n) - h\big(\boldsymbol{x}(n)\big)\right]x_k(n),\ \ k = 1, \dots, K$$

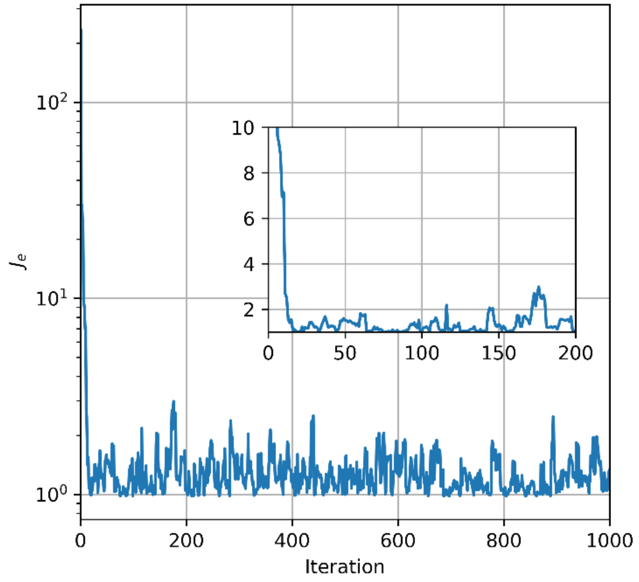
As amostras da função observável podem conter ruído.

- **Características:**
  - ***Aproxima o vetor gradiente*** através de uma ***estimativa estocástica***, ou seja, gradiente é calculado com um exemplo ***tomado aleatoriamente do conjunto de treinamento***.
  - Essa ***aproximação estocástica*** faz com que as ***atualizações dos pesos não sigam a direção de máxima declividade***, ***podendo ter direções divergentes a cada iteração.***
  - Utilizado quando os ***atributos e rótulos*** são ***obtidos sequencialmente*** (e.g., sensores).
  - Ou quando o ***conjunto de treinamento é muito grande*** (toma-se amostras aleatoriamente).
  - ***Computacionalmente mais rápido*** e ***menos custoso em termos de CPU e memória*** que o GD em batelada.
  - ***Se as amostras estiveram contaminadas com ruído, a convergência não é garantida*** com um passo de aprendizagem fixo.
    - ✓ O algoritmo pode ***oscilar*** em torno do mínimo ***sem nunca convergir*** para o valor ótimo.
  - Esquemas de variação do passo de aprendizagem ajudam a garantir a convergência.

Características do GD Estocástico







- Por aproximar o vetor gradiente com apenas um ***exemplo tomado de forma aleatória***, ***não apresenta um caminho regular para o mínimo***, mudando de direção várias vezes.
- Se as ***amostras contiverem ruído***, o algoritmo ***não converge para o mínimo***: "*oscila*" em torno dele.
- Nesse caso, quando o treinamento termina, ***os valores finais dos pesos podem não ser ótimos***.
- Além disso, a ***convergência ocorre apenas na média***.
- Entretanto, ***consome menos recursos computacionais*** e o ***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, α, para ficar mais "*comportado*".

[Exemplo: stocastic_gradient_descent_with_figures.ipynb]

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.

$$a_k = a_k + \alpha \boxed{\sum_{n=0}^{MB-1}} [y(n) - h(\boldsymbol{x}(n))]\, x_k(n),\ k = 1, \ldots, K$$

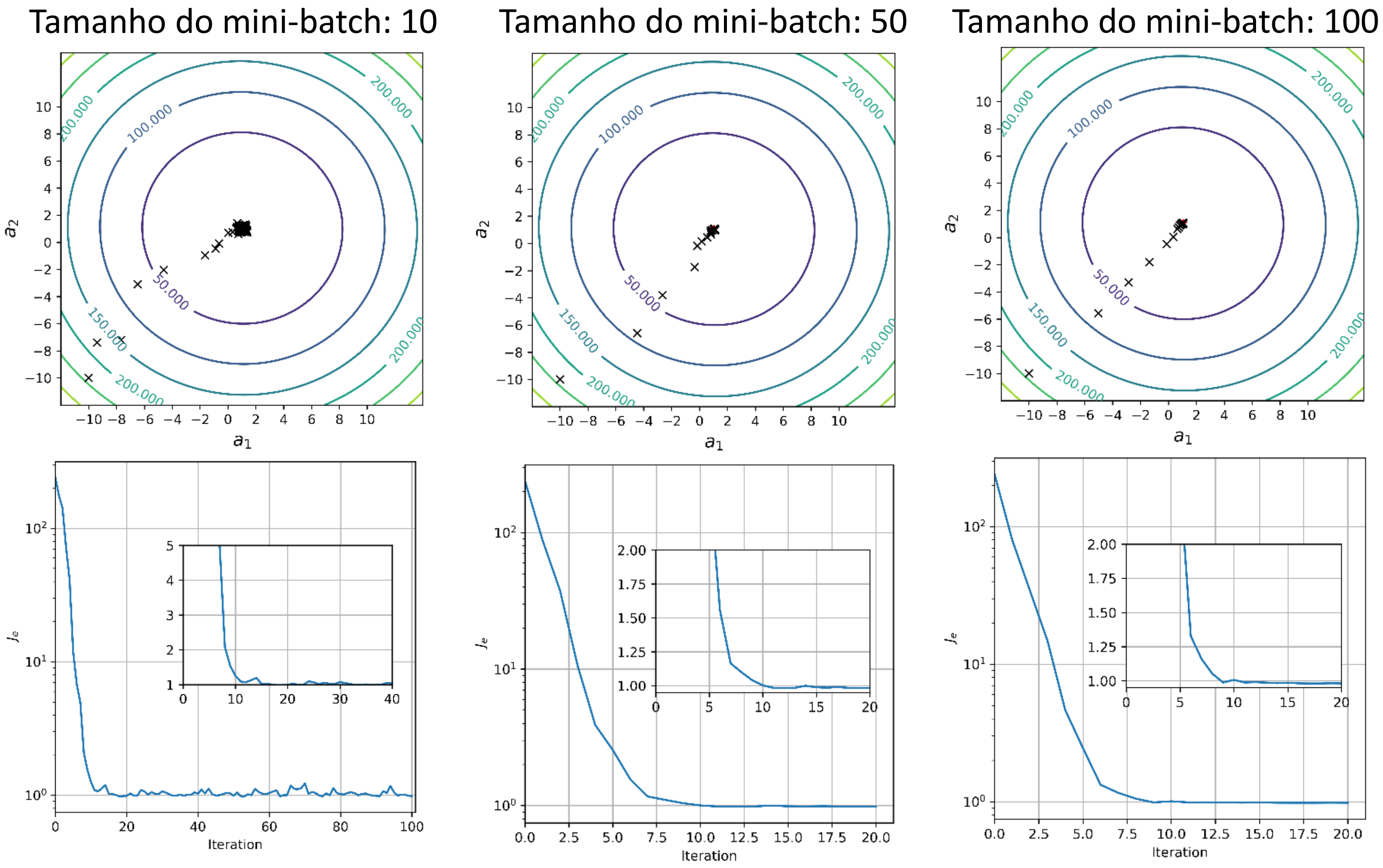
onde MB é o tamanho do mini-batch.

Características:

- Pode ser visto como uma ***generalização*** das 2 versões anteriores:
  - Caso $MB = \text{N}$, então ele se torna o GD em batelada.
  - Caso $MB = 1$, então ele se torna o GD estocástico.
- Computacionalmente mais rápido do que o GD em batelada, mas mais lento do que o GD estocástico.
- Em caso de amostras ruidosas, a convergência depende do tamanho do mini-batch.
- Pode usar esquemas de variação do passo de aprendizagem para melhorar a convergência caso o mini-batch seja muito pequeno.

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***.
- Esquema de redução de α pode balancear ***rapidez*** e ***convergência***.



Exemplo: mini_batch_gradient_descent_with_figures.ipynb

Tarefas

- **Quiz**: “*T319 - Quiz - Regressão: Parte II*” que se encontra no MS Teams.
- **Exercício Prático**: **Laboratório #3**.
  - Pode ser acessado através do link acima (Google Colab) ou no GitHub.
  - Vídeo explicando o laboratório: Arquivos -> Material de Aula -> Laboratório #3
  - 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!

Encontrando o vetor gradiente

Função hipótese com 2 pesos, a_1 e a_2

$$\hat{y}(n) = h(\boldsymbol{x}(n)) = a_1x_1(n) + a_2x_2(n).$$

A função de erro é dada por

$$J_e(\boldsymbol{a}) = \frac{1}{N}\sum_{n=0}^{N-1}\left[y(n) - \left(a_1x_1(n) + a_2x_2(n)\right)\right]^2.$$

Cada elemento do vetor gradiente é dado por

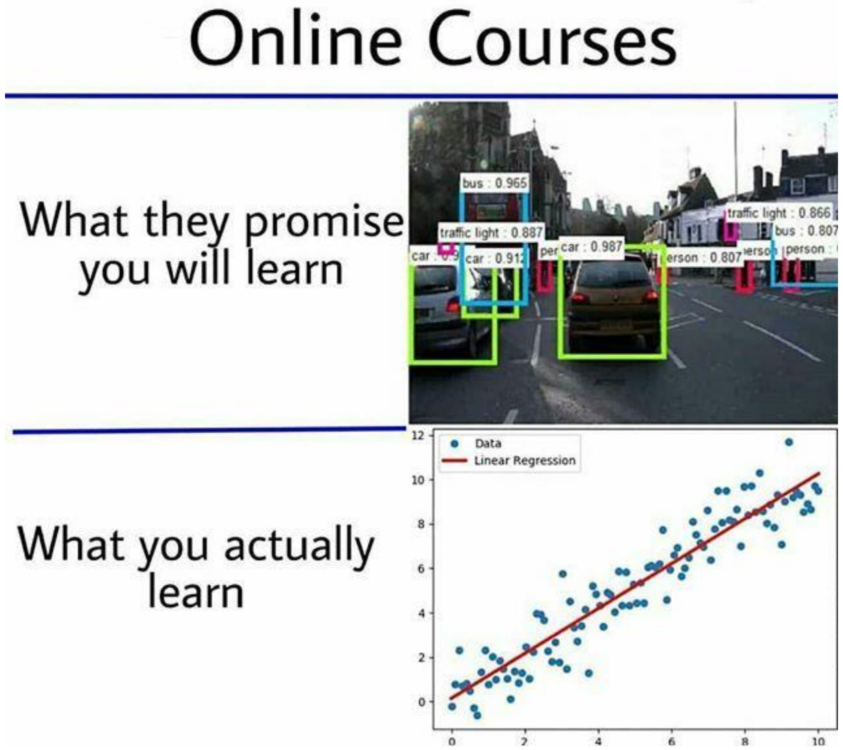
$$\frac{\partial J_e(\boldsymbol{a})}{\partial a_k} = \frac{\partial \frac{1}{N}\sum_{n=0}^{N-1}\left[y(n) - \left(a_1x_1(n) + a_2x_2(n)\right)\right]^2}{\partial a_k}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1}\frac{\partial\left[y(n) - \left(a_1x_1(n) + a_2x_2(n)\right)\right]^2}{\partial a_k}$$

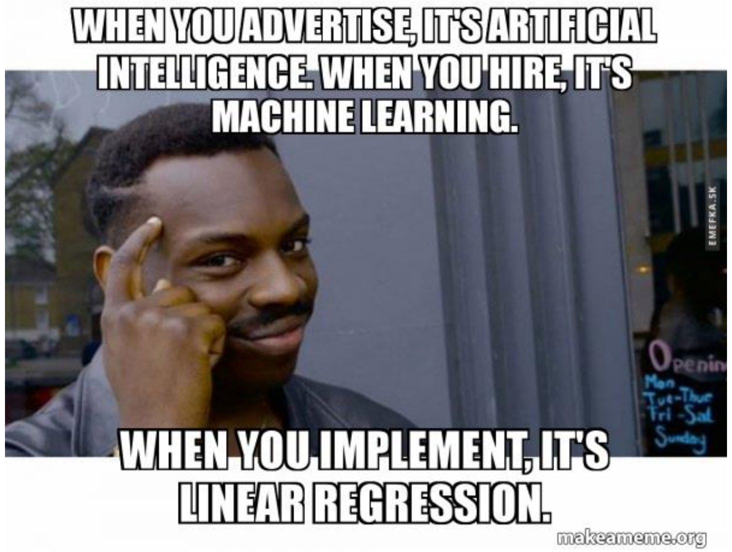
Operação da derivada parcial é distributiva.

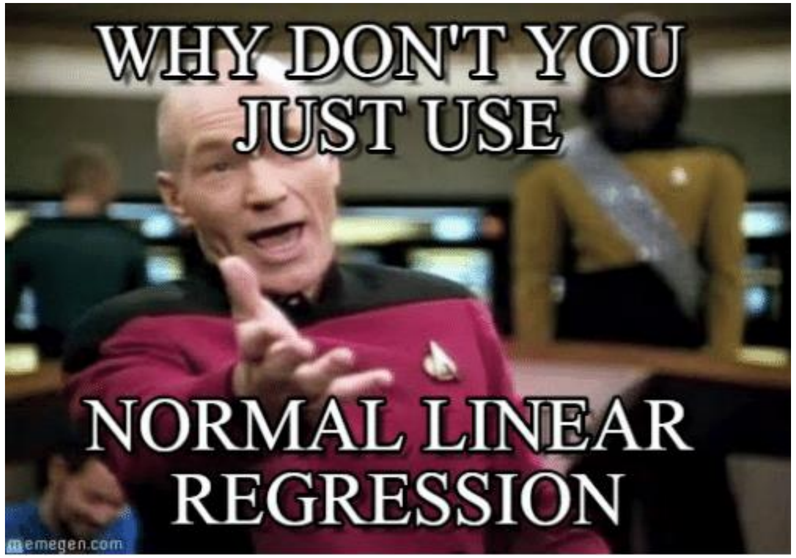
$$= -\frac{2}{N}\sum_{n=0}^{N-1}\left[y(n) - \left(a_1x_1(n) + a_2x_2(n)\right)\right]x_k(n)\,, k = 1{,}2$$


Deep Learning
linear regression


Online Courses
What they promise you will learn
bus : 0.965
traffic light : 0.887
car : 0.912
car : 0.987
traffic light : 0.866
bus : 0.807
person : 0.807
What you actually learn
Data
Linear Regression

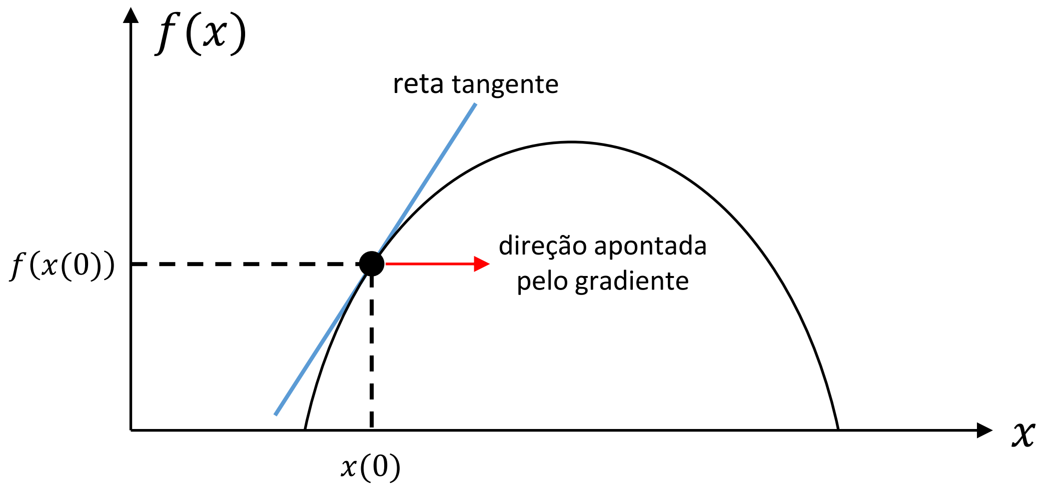

Linear Regression is NOT Machine Learning
CHANGE MY MIND
imgflip.com

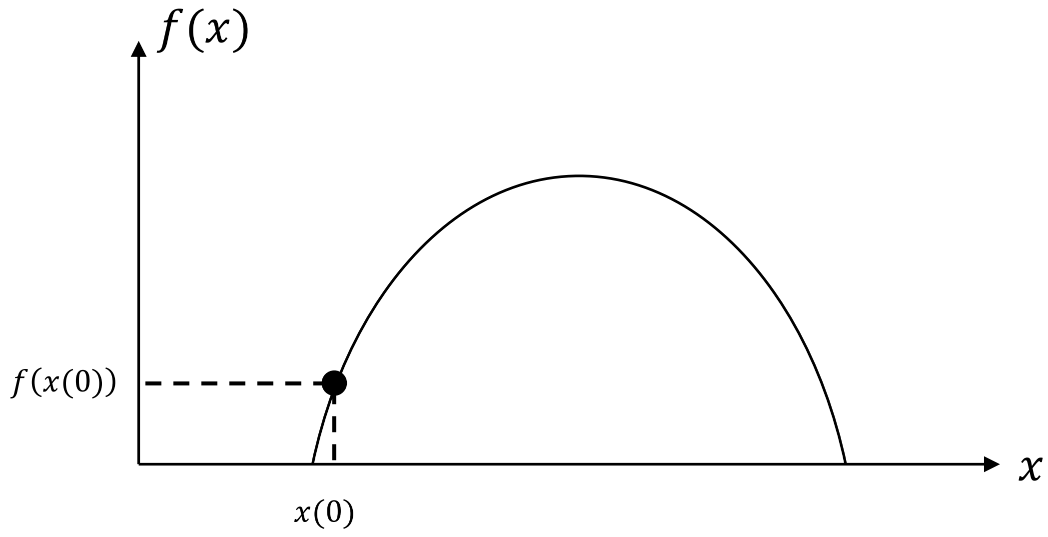

WHEN YOU ADVERTISE, IT'S ARTIFICIAL INTELLIGENCE. WHEN YOU HIRE, IT'S MACHINE LEARNING.
WHEN YOU IMPLEMENT, IT'S LINEAR REGRESSION.
makeameme.org


WHY DON'T YOU JUST USE
NORMAL LINEAR REGRESSION
memegen.com

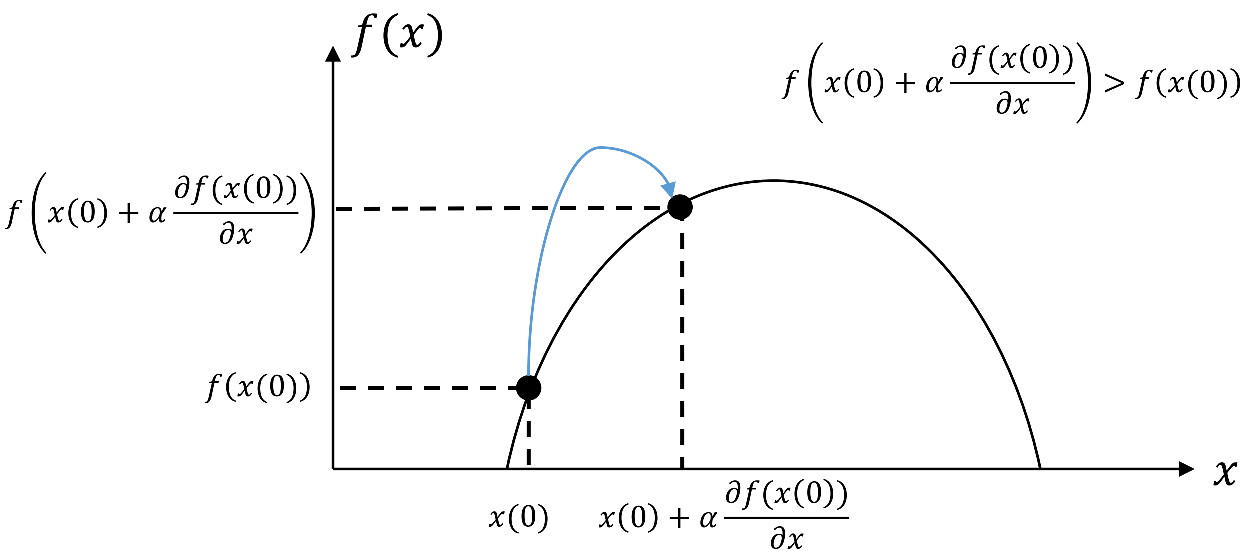

LEARNING ML,DL FROM UNIVERSITY
ONLINE COURSES
FROM YOUTUBE
FROM ARTICLES
FROM MEMES

FIGURAS

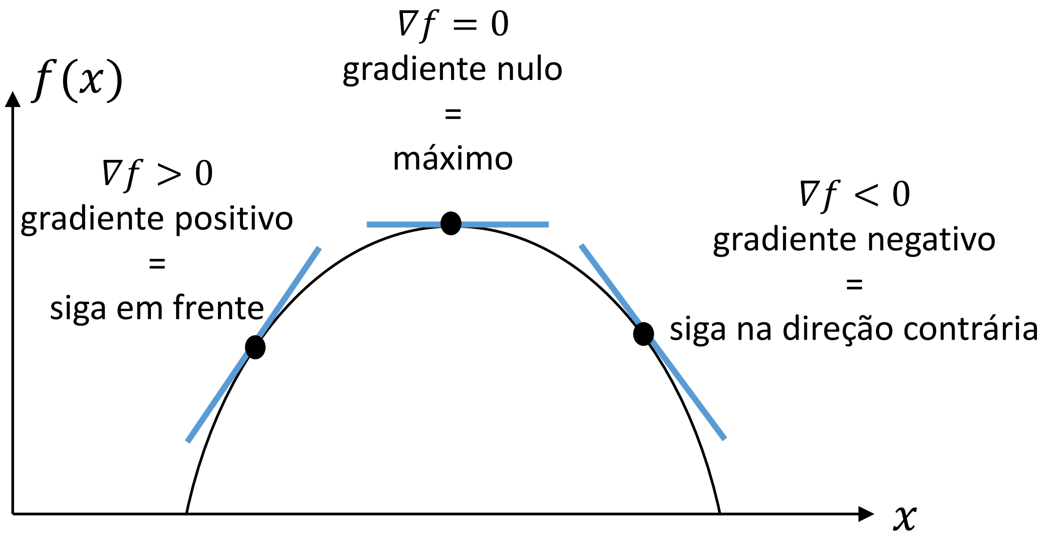

$f(x)$
reta tangente
$f(x(0))$
direção apontada
pelo gradiente
$x(0)$
x

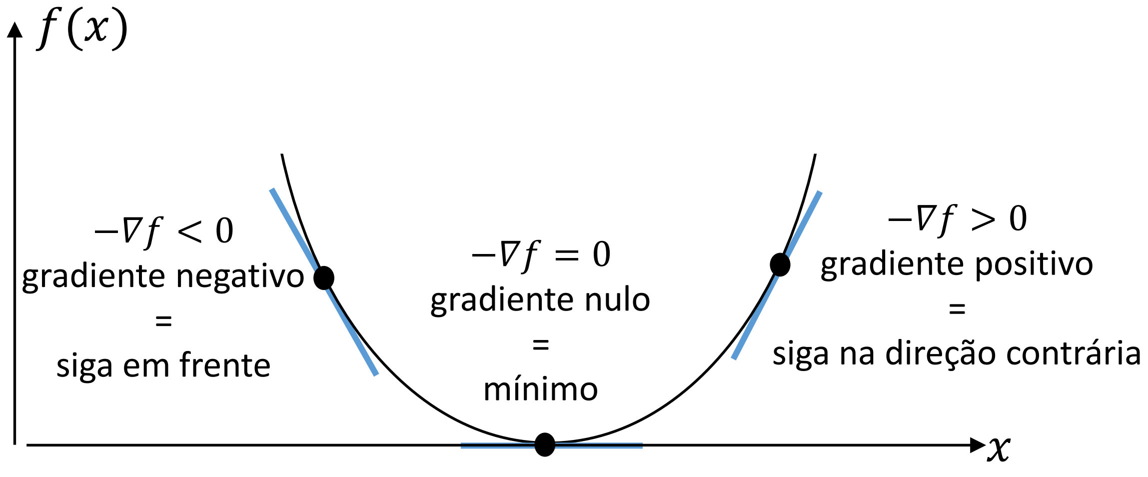


$f(x)$
$f(x(0))$
x
$x(0)$



$f(x)$
$f\left(x(0) + \alpha \frac{\partial f(x(0))}{\partial x}\right) > f(x(0))$
$f\left(x(0) + \alpha \frac{\partial f(x(0))}{\partial x}\right)$
$f(x(0))$
x
$x(0)$
$x(0) + \alpha \frac{\partial f(x(0))}{\partial x}$


$f(x)$
$\nabla f = 0$
gradiente nulo
=
máximo
$\nabla f > 0$
gradiente positivo
=
siga em frente
$\nabla f < 0$
gradiente negativo
=
siga na direção contrária
x


$f(x)$
$-\nabla f < 0$
gradiente negativo
=
siga em frente
$-\nabla f = 0$
gradiente nulo
=
mínimo
$-\nabla f > 0$
gradiente positivo
=
siga na direção contrária
x