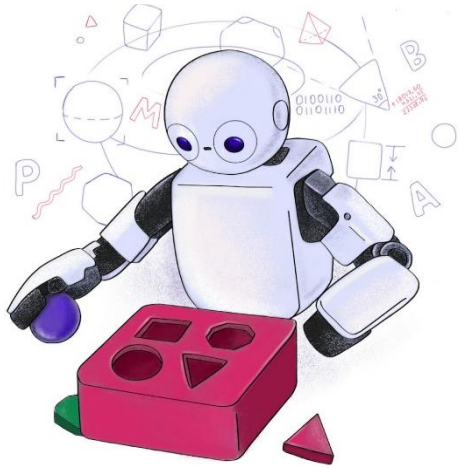
T319 - Introdução ao Aprendizado de Máquina:

Regressão Linear (Parte II)



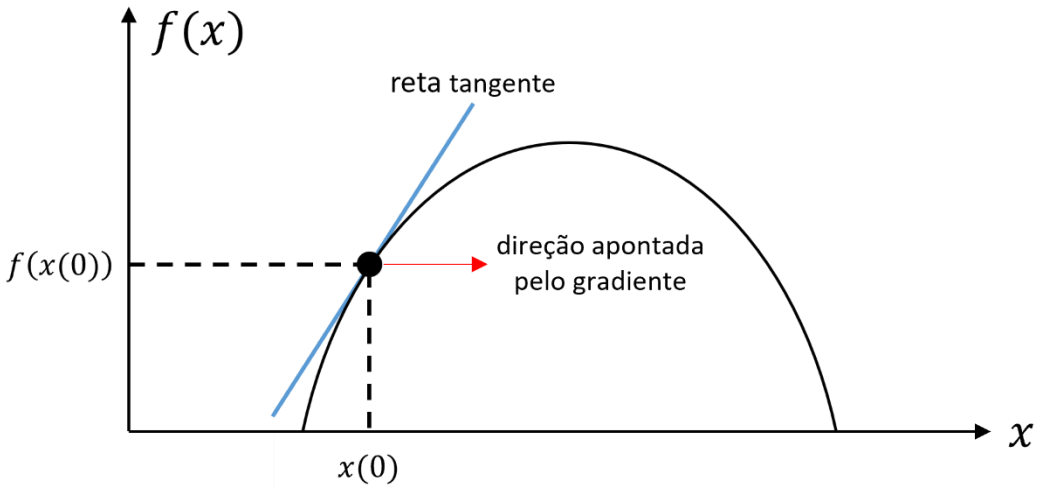
Inatel

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

Recapitulando

- Vimos a ***motivação*** por trás da ***regressão linear***: encontrar funções que ***aproximem o comportamento geral*** de um conjunto de amostras (normalmente 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 a ***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 começaremos a discutir a seguir.

Vetor gradiente

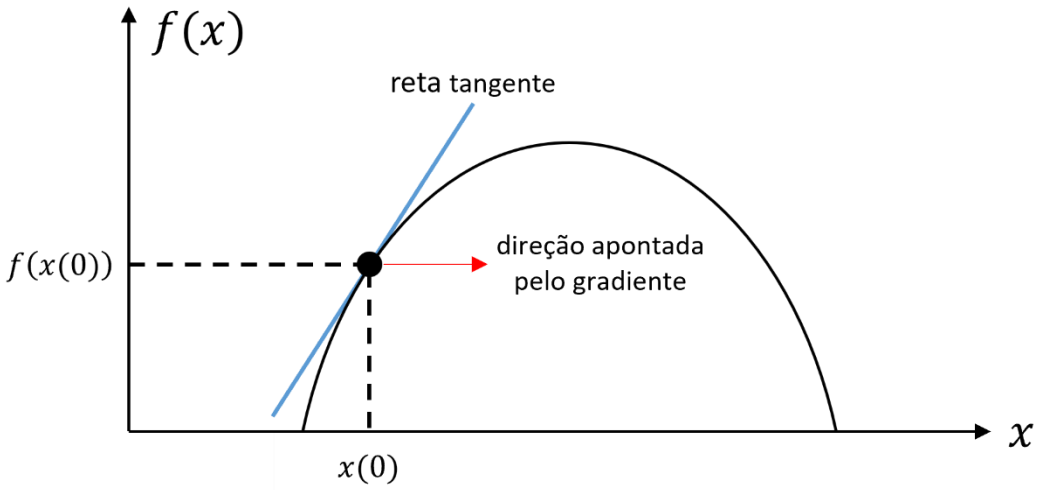


- Vocês se lembram das aulas de cálculo vetorial, onde vocês aprenderam sobre o ***vetor gradiente***?

$$\nabla f(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

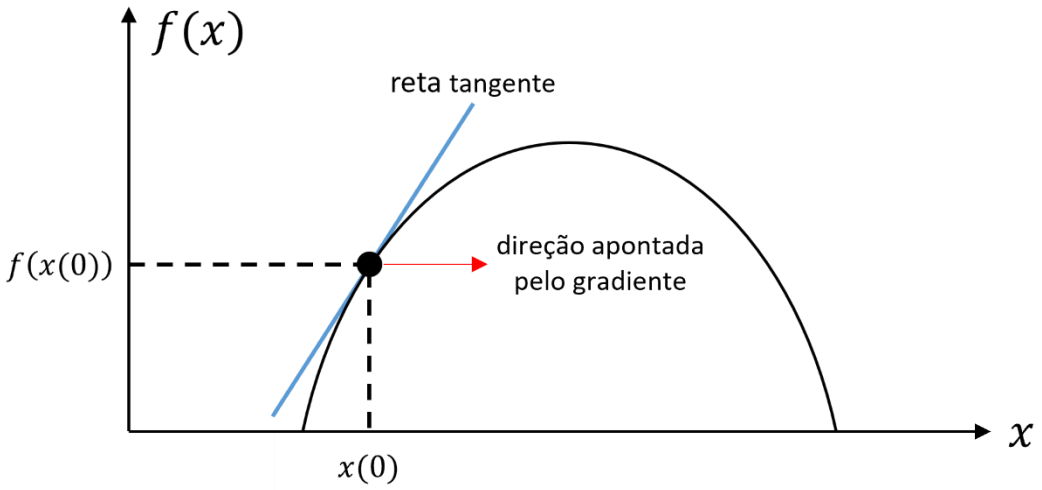
- Qual ***informação*** ele nos dá sobre a função, $f(\boldsymbol{x})$?

Vetor gradiente



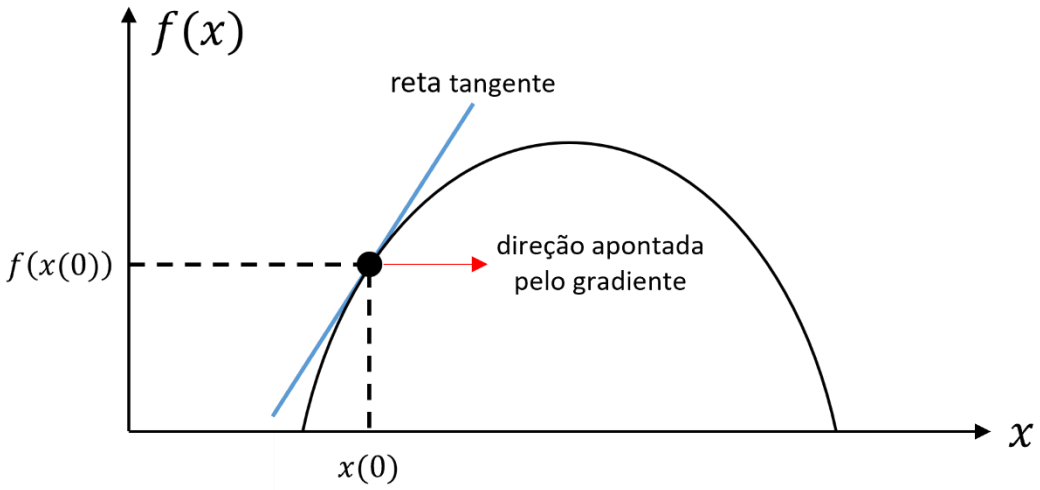
- O vetor gradiente ***aponta*** na ***direção*** em que a função $f(\boldsymbol{x})$ ***cresce mais rapidamente a partir do ponto em que é avaliado***.

Vetor gradiente



- As ***magnitudes*** dos elementos do vetor gradiente indicam a ***taxa de crescimento da função*** na direção apontada por ele.
  - Quanto maior a magnitude, maior a taxa de crescimento naquela direção.
- Os ***sinais*** dos elementos do vetor dizem para qual "***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, f, ter apenas um argumento, 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 hiperplano tangente à função no ponto onde ele é calculado***.
  - Quanto ***maior o valor absoluto*** do gradiente, ***mais inclinado é o hiperplano 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

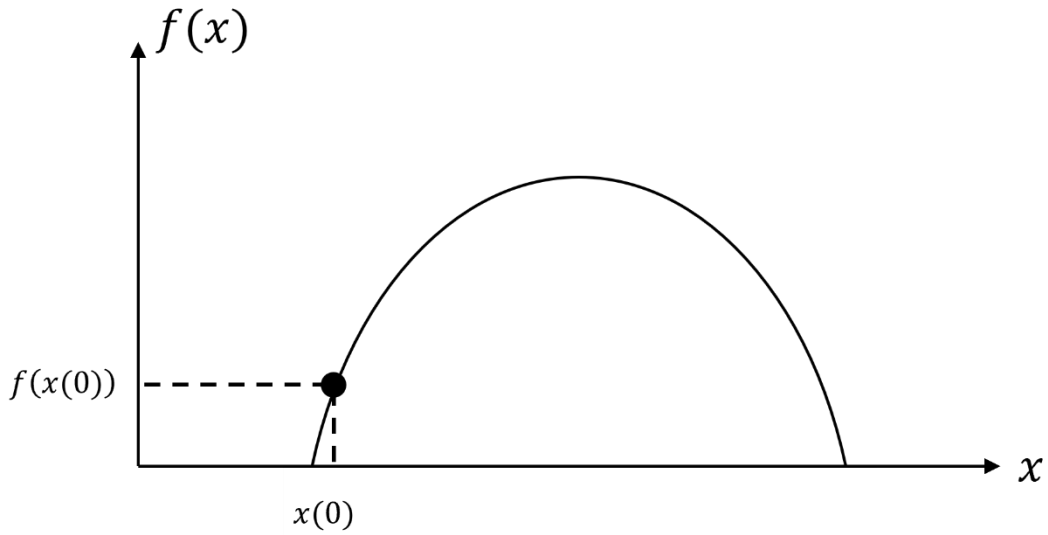
- O ***vetor gradiente*** de uma função com K argumentos, $f(x_1, x_2, \dots, x_K)$, é definido pela ***derivada parcial em relação a cada um de seus argumentos*** $x_k, k = 1, \dots, K$:

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

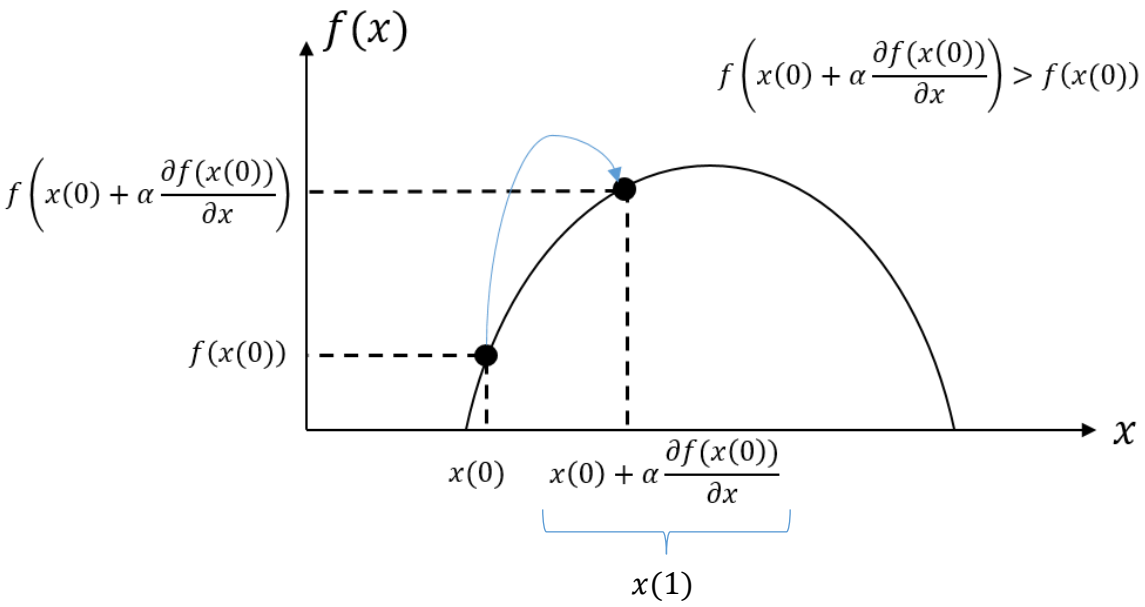
- Notem que o vetor gradiente é representado pelo símbolo *Nabla*, ∇, e é definido como um ***vetor coluna***, com ***número de elementos igual ao número de argumentos da função***.
- **OBS**.: Na sequência, por questões didáticas, mas sem perda de generalidade, nós vamos assumir uma função com apenas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

O vetor gradiente indica o caminho para o ponto de máximo da função

- Imaginem o ponto inicial, $x(0)$, 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 ponto de máximo da função

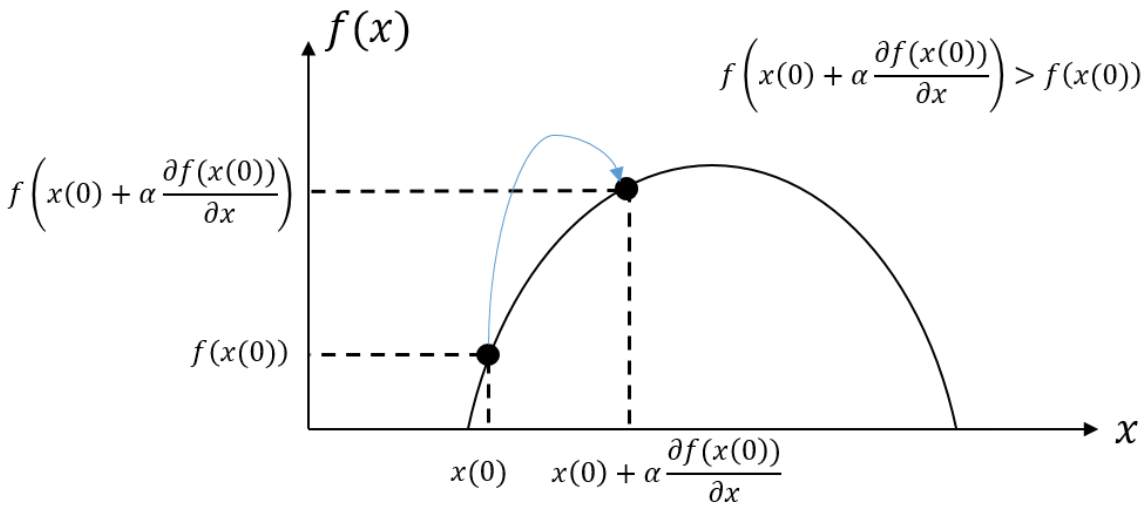


- O vetor gradiente calculado no ponto $x(0)$, $\nabla f(x(0))$, diz ***em qual direção*** devemos caminhar para ***aumentar o valor da função*** $f(x)$ mais rapidamente.
- Se ***adicionarmos*** uma ***porcentagem****, α, do gradiente ao valor de $x(0)$, teremos que o ***novo ponto***, $x(1)$, terá um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

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

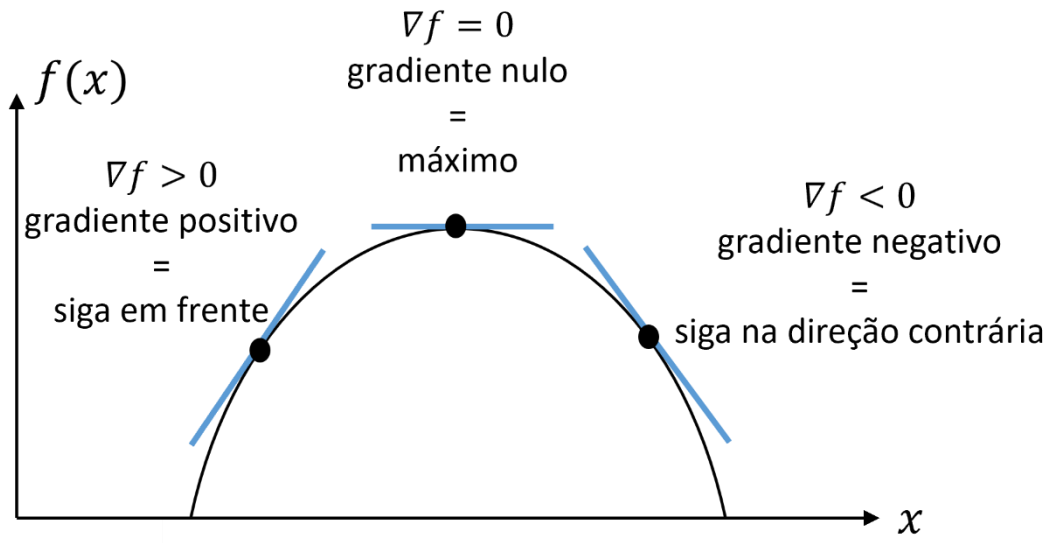
** O vetor gradiente dá a direção e não a distância até o ponto de máximo.*

O vetor gradiente indica o caminho para o ponto de máximo da função



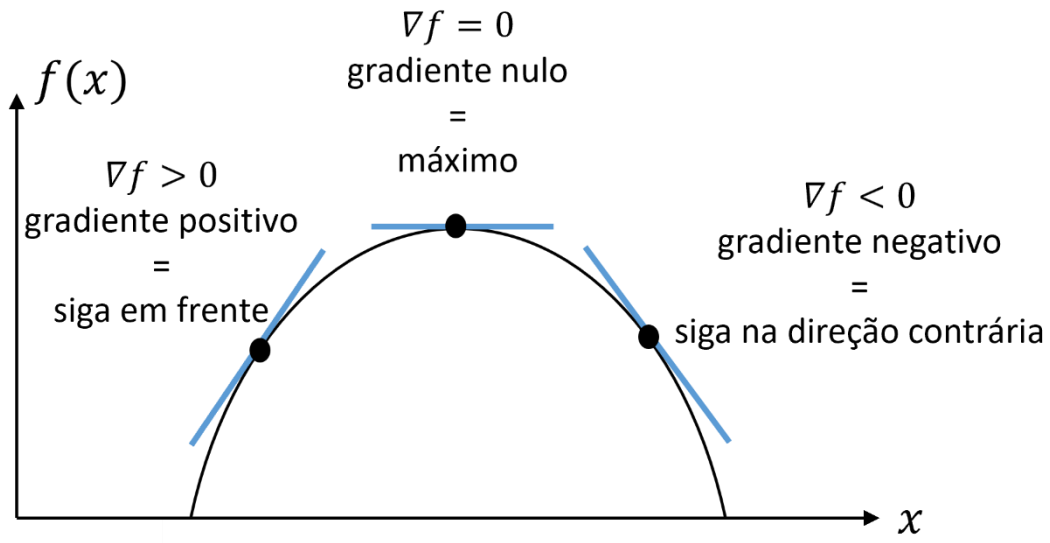
- Se, a cada ***ponto atual***, nós calcularmos o vetor gradiente e adicionarmos uma ***porcentagem*** dele àquele ponto, teremos um ***novo ponto*** que leva a um ***valor da função*** $f(x)$ ***maior do que o valor anterior***.
- Portanto, podemos criar um ***procedimento*** que vá ***iterativamente caminhando*** em ***direção ao ponto de máximo da função***.
- Vamos entender como isso pode ser feito.

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 com sinal:
  - + (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, um valor ***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

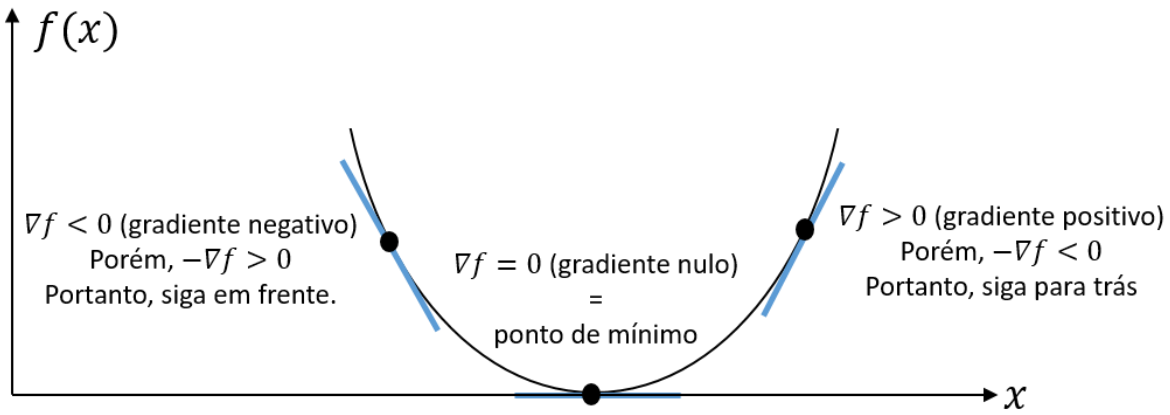
$$f(x(i+1)) > f(x(i)), i \geq 0.$$

- 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*** (i.e., o algoritmo convergiu).
  - Chamamos de ***convergência*** quando a variação da função $f(x)$ entre iterações consecutivas é muito pequena, ou seja, o valor é praticamente constante.

Porém, se vocês se lembram, no problema da regressão linear nós queremos ***encontrar o ponto de mínimo da função de erro*** ao invés do seu máximo.

O que devemos fazer?

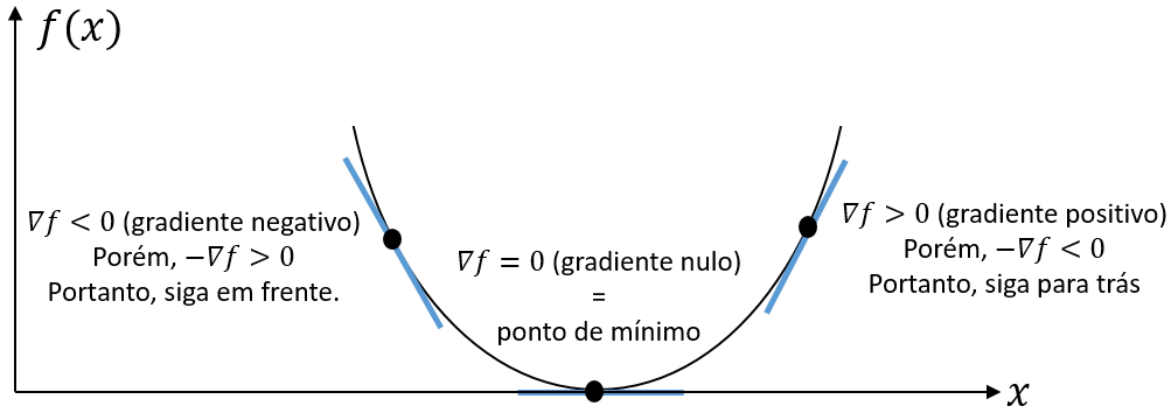
Algoritmo do gradiente descendente



- Se para encontrarmos o ***ponto de máximo*** de uma função basta seguirmos a direção apontada pelo vetor gradiente.
- Portanto, para encontrarmos o ***ponto de mínimo***, basta seguirmos na ***direção oposta a apontada pelo vetor gradiente*** em um determinado ponto, $x(i)$

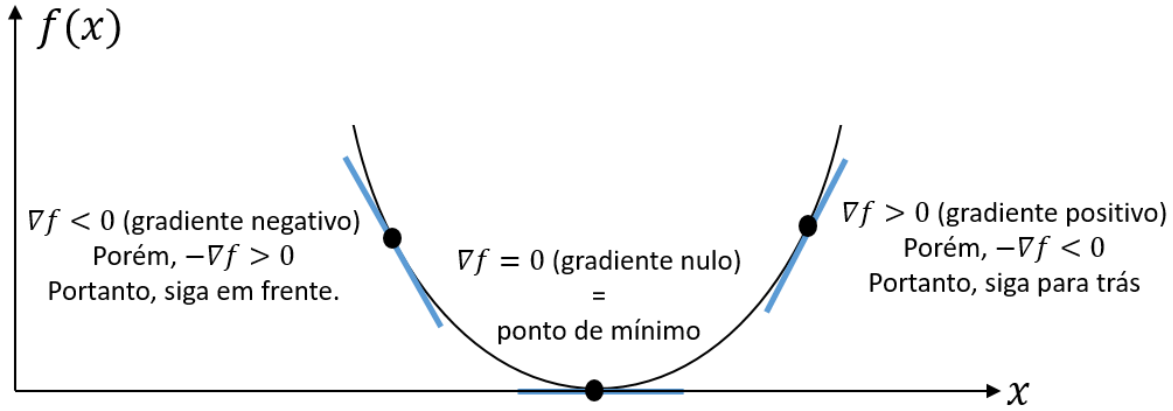
$$-\nabla f\big(x(i)\big).$$

Algoritmo do gradiente descendente



- Quando seguimos na ***direção contrária a da máxima taxa de crescimento***, dada pelo ***vetor gradiente***, estamos indo na ***direção de decrescimento mais rápido da função.***
- Portanto, um elemento do vetor gradiente com sinal:
  - - (reta com inclinação negativa) indica que o ***ponto de mínimo está à frente***.
  - + (reta com inclinação positiva) indica que o ***ponto de mínimo está atrás***.
  - 0 (reta com inclinação nula) indica que ***ponto de mínimo foi encontrado***.

Algoritmo do gradiente descendente



- Assim, um ***algoritmo iterativo de otimização*** que siga na ***direção contrária*** a indicada pelo ***vetor gradiente*** para encontrar o ***ponto de mínimo*** de uma função é chamado de ***gradiente descendente (GD)***.

Algoritmo do gradiente descendente

- Inicializa-se o argumento $x(0)$ com um valor arbitrário, em geral, um valor ***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

Única diferença para o gradiente ascendente.

$$x(i+1) = x(i) - \alpha \nabla f(x(i)), i \geq 0.$$

- 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 mínimo*** seja atingido, ou seja, $\nabla f(x(i)) = 0$ e, consequentemente, o argumento x ***não sofra mais atualizações*** (i.e., o algoritmo convergiu).
  - ***Convergência***: é quando o valor de x e, consequentemente, de $f(x)$ entre iterações consecutivas é praticamente constante.

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, $f(x_0, x_1, \dots, x_K)$.
- Esse será o caso das funções que iremos utilizar em breve.
- No nosso caso, a função será a do ***erro quadrático médio (EQM)***, que terá como ***argumentos*** os ***pesos da função hipótese***.

Como neste curso queremos ***minimizar a função de erro***, iremos focar no ***gradiente descendente***.

Características do gradiente descendente

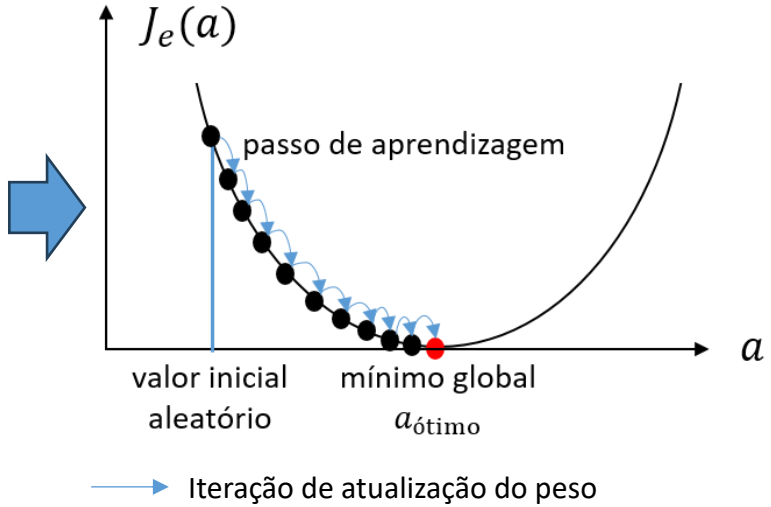
- Algoritmo ***genérico de otimização***: pode ser aplicado não apenas a problemas de aprendizado de máquina, mas a ***qualquer problema que envolva encontrar os parâmetros que minimizem uma função***.
- O único requisito é que essa função seja ***diferenciável***.
  - No caso da regressão linear, a função de erro deve ser diferenciável.
- Escalona melhor do que o método da ***equação normal*** para grandes conjuntos de dados.
- É de fácil implementação.
- Não precisamos nos preocupar com matrizes ***mal condicionadas***, ou seja, matrizes com ***determinante próximo de 0*** (i.e., quase ***singulares***).
- Pode ser usado com modelos não-lineares.

*Como aplicamos o algoritmo do **gradiente descendente** ao problema da **regressão linear**?*

*Ou seja, como encontramos os **pesos ótimos da função hipótese** usando o gradiente descendente?*

O algoritmo do gradiente do descendente

$\boldsymbol{a}$ ← inicializa o vetor de pesos em um ponto aleatório do espaço de pesos
loop até convergir **ou** atingir o número máximo de iterações **do**
$\nabla J_e(\boldsymbol{a})$ ← cálculo do vetor gradiente, $\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}}$
$\boldsymbol{a} \leftarrow \boldsymbol{a} - \alpha \times \nabla J_e(\boldsymbol{a})$ (***regra de atualização dos pesos***)



- **OBS**.: O ***passo de aprendizagem***, α, dita o tamanho do deslocamento dado na direção oposta a apontada pelo ***vetor gradiente***.
  - Ele é sempre um valor maior do que zero.
- Na sequência, encontraremos o ***vetor gradiente*** da ***função de erro***.

Calculando o vetor gradiente

- Para calcularmos o vetor gradiente, vamos considerar o ***EQM*** como ***função de erro*** e a ***função do hiperplano*** como ***função hipótese***:

$$\hat{y}(n) = h(n) = a_0 + a_1 x_1(n) + \cdots + a_K x_K(n) = \sum_{i=0}^{K} a_i x_i(n) = \boldsymbol{a}^T \boldsymbol{x}(n),$$

onde n é o ***número do exemplo*** (ou amostra), K é o ***número atributos***, $a_i, \forall i$ e $x_i, \forall i$ são os ***pesos e atributos da função hipótese***, respectivamente, $x_0 = 1$ (i.e., atributo de *bias*) e $\boldsymbol{a}$ e $\boldsymbol{x}(n)$ são vetores coluna com todos os pesos e atributos da n-ésima amostra, respectivamente.

- Agora podemos encontrar o vetor gradiente.

Calculando o vetor gradiente

- O ***vetor gradiente*** da ***função de erro em relação aos pesos*** é dado por

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

$$= -\frac{2}{N}\sum_{n=0}^{N-1}[y(n) - \hat{y}(n)]\boldsymbol{x}(n) = \boxed{-\frac{2}{N}\boldsymbol{X}^T(\boldsymbol{y} - \hat{\boldsymbol{y}}),}$$

Versão matricial do vetor gradiente

onde $\boldsymbol{X}$ é a matriz de atributos com dimensão $(N \times K + 1)$ e $\boldsymbol{y}$ e $\hat{\boldsymbol{y}}$ são vetores coluna $(N \times 1)$ com todos os valores esperados e de saída da função hipótese para os N exemplos coletados, respectivamente.

- **OBS**.: Esse cálculo pode ser diretamente estendido para polinômios.

Atualizando os pesos

- Substituindo o ***vetor gradiente*** na ***equação de atualização dos pesos***, temos

$$\boldsymbol{a} = \boldsymbol{a} - \alpha \frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = \boldsymbol{a} + \alpha \frac{2}{N} \sum_{n=0}^{N-1} [y(n) - \hat{y}(n)] \boldsymbol{x}(n) = \boldsymbol{a} + \frac{2\alpha}{N} \boldsymbol{X}^T (\boldsymbol{y} - \hat{\boldsymbol{y}}).$$

- Percebam que o vetor gradiente é a ***média da diferença*** entre o rótulo e a saída da função hipótese vezes os atributos ***tomada ao longo de todos os N exemplos do conjunto de treinamento.***
- **Observações**:
  - O sinal + é devido ao vetor gradiente encontrado ter sinal negativo.
  - Por ser constante, o termo $2/N$ pode ser absorvido por α.

Atualizando os pesos

$\boldsymbol{a} \leftarrow$ inicializa o vetor de pesos em um ponto aleatório do espaço de pesos
loop até convergir **ou** atingir o número máximo de iterações **do**
$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \alpha \frac{2}{N} \sum_{n=0}^{N-1} [y(n) - \hat{y}(n)] \boldsymbol{x}(n)$$

- Lembrem-se que ***a cada iteração*** (i.e., *loop*) do gradiente descendente, precisamos ***calcular o vetor gradiente***.
- Isso envolve ***calcular o somatório acima para cada iteração.***
- Assim, ***se o conjunto de treinamento e o modelo forem muito grandes***, o ***treinamento pode ser muito longo e consumir muita CPU e memória***.

Versões do gradiente descendente

- Portanto, para lidar com essa situação, podemos ter ***3 versões diferentes, dependendo da quantidade de exemplos*** considerados no somatório do vetor gradiente:
  - ***Gradiente descendente em batelada*** **(GDB)**
  - ***Gradiente descendente estocástico*** **(GDE)**
  - ***Gradiente descendente em mini-lotes*** **(GDML)**

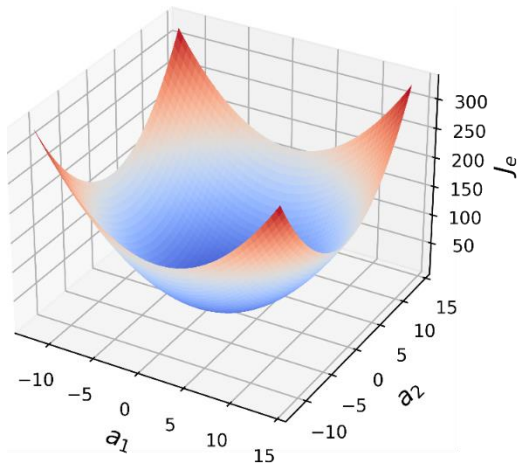
Gradiente descendente em batelada

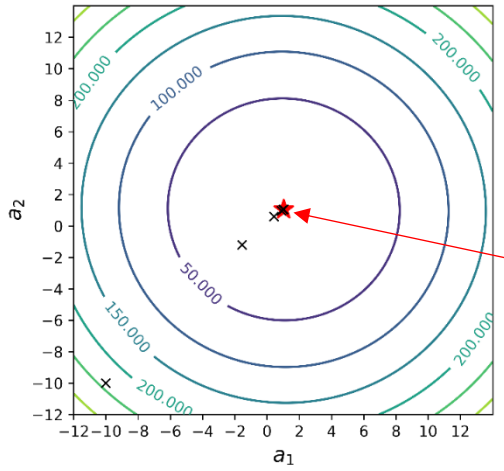
$$\boldsymbol{a} = \boldsymbol{a} + \alpha \frac{2}{N} \sum_{n=0}^{N-1} [y(n) - \hat{y}(n)] \boldsymbol{x}(n) .$$

- Considera ***todos os exemplos*** do conjunto de treinamento para calcular o vetor gradiente.
- Pode ser ***computacionalmente custoso*** dependendo do tamanho do modelo e do conjunto de dados.
  - Por processar todos os exemplos a cada iteração, pode ser lento e consumir muita CPU e memória com conjuntos muito grandes.
- ***Convergência*** para o mínimo global é ***garantida*** quando a função de erro é ***convexa*** e o passo de aprendizagem, α, não for muito grande.
- É a versão que ***obtém os melhores resultados***.

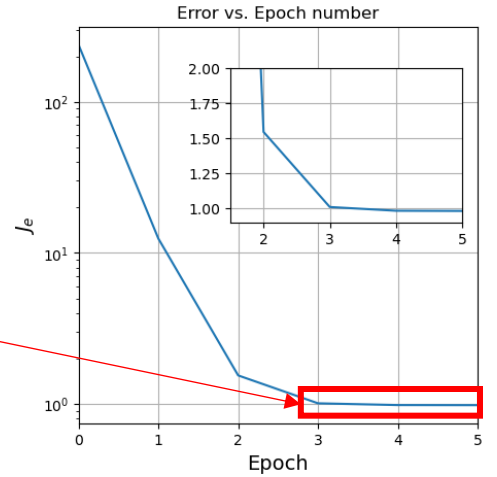
Exemplo: gradiente descendente em batelada

Características do GD em batelada





convergência



- Por usar todos os exemplos, ***segue diretamente***, sem alterar a direção, para o ***ponto de mínimo***.
  - Nesse exemplo, 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).
- ***Convergência é garantida*** dado que o ***passo de aprendizagem não seja muito grande*** e se ***espere tempo suficiente***.
  - Não fica "*oscilando*" em torno do ponto de mínimo após alcançá-lo, pois o vetor gradiente neste ponto é praticamente nulo.

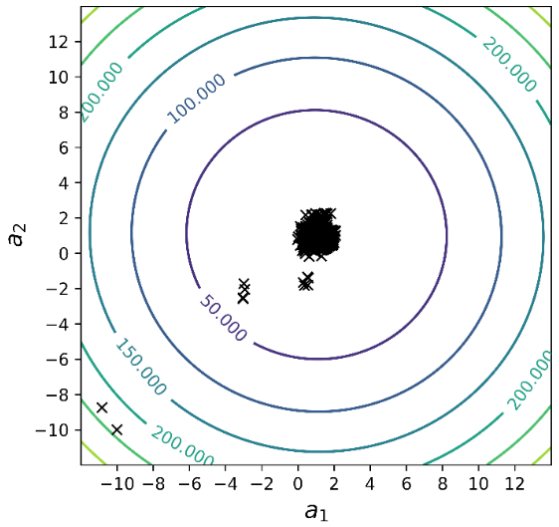
Gradiente descendente estocástico

$$\boldsymbol{a} = \boldsymbol{a} + \alpha 2[y(n_{\text{random}}) - \hat{y}(n_{\text{random}})]\boldsymbol{x}(n_{\text{random}}).$$

- Utiliza a cada ***iteração de atualização dos pesos*** ***apenas um exemplo*** do conjunto de treinamento para calcular uma ***estimativa estocástica do vetor gradiente***.
  - É estocástica pois a cada iteração toma-se ***uma amostra aleatória*** do conjunto de treinamento para calcular a estimativa do vetor gradiente.
- Por usar uma estimativa, ***não segue diretamente a direção de máxima declividade da função de erro***, ***mudando de direção várias vezes***.

Gradiente descendente estocástico

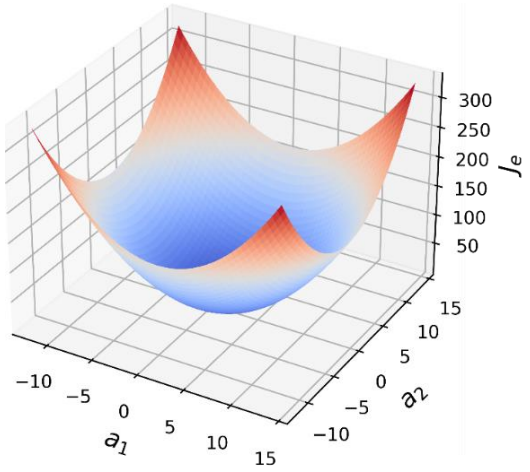
$$\boldsymbol{a} = \boldsymbol{a} + \alpha 2[y(n_{\text{random}}) - \hat{y}(n_{\text{random}})]\boldsymbol{x}(n_{\text{random}}).$$

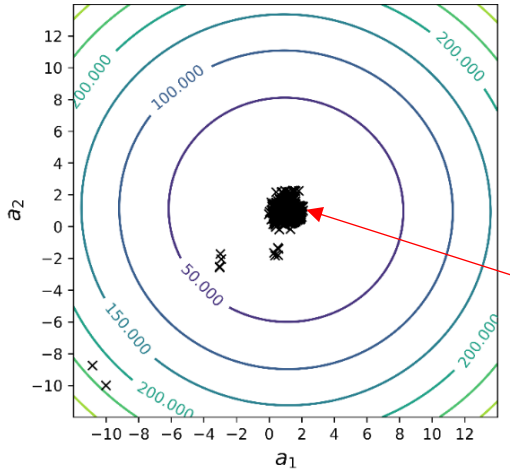


- Quando os ***dados de treinamento estão contaminados com ruído***, a ***estimativa*** do gradiente é ***ruidosa***, fazendo com que a ***convergência não ocorra*** ou ***não seja garantida***.
  - O algoritmo ***oscila*** em torno do ponto de mínimo ***sem nunca convergir***.
- Entretanto, é ***mais rápido e menos complexo computacionalmente***, usando menos CPU e memória do que o GDB.

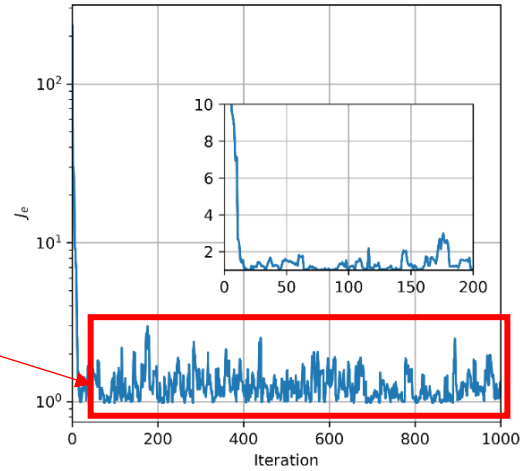
Exemplo: gradiente descendente estocástico

Características do GD estocástico





Não converge



- ***Apresenta um caminho irregular para o mínimo***, mudando de direção várias vezes.
- Quando as ***amostras contém ruído (caso acima)***, ***não converge para o ponto de mínimo***, "*oscilando*" em torno dele.
  - Essa oscilação também pode ser vista na curva de erro.
- Algumas técnicas podem ser usadas para torná-lo mais comportado e talvez convergir: redução do passo, *early-stop*, momentum, etc.

Gradiente descendente em mini-lotes

$$\boldsymbol{a} = \boldsymbol{a} + \alpha \frac{2}{MB} \sum_{n=0}^{MB-1} [y(n) - \hat{y}(n)]\boldsymbol{x}(n)\,.$$

- Utiliza a cada iteração um ***subconjunto aleatório de exemplos***, de tamanho MB, do conjunto de treinamento para o cálculo do gradiente.
- Em geral, $1 < MB < N$, portanto é mais rápido que o GDB e mais preciso e estável do que o GDE.

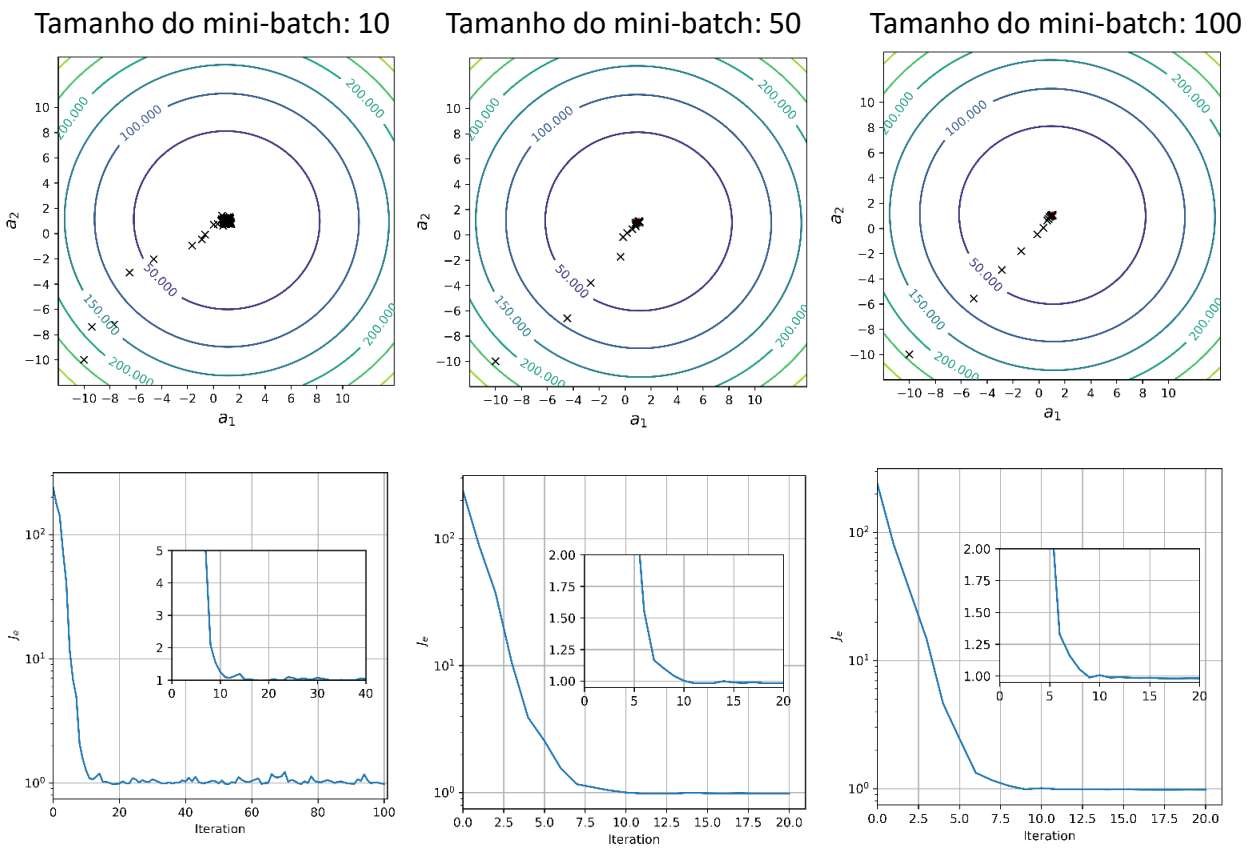
Gradiente descendente em mini-lotes

$$\boldsymbol{a} = \boldsymbol{a} + \alpha \frac{2}{MB} \sum_{n=0}^{MB-1} [y(n) - \hat{y}(n)]\boldsymbol{x}(n)\,.$$

- Porém, por MB ser variável, essa versão é vista como uma ***generalização*** das duas versões anteriores, pois MB pode ser feito igual a 1 ou N.
- Em caso de ***amostras ruidosas***, a convergência depende do tamanho de MB, ***quanto maior, melhor é a estimativa*** do vetor gradiente e, consequentemente, ***maior a chance de convergência***.

Exemplo: gradiente descendente em mini-lotes

Características do GD em mini-lotes



- Vejam que conforme MB aumenta,
  - o progresso se torna menos irregular do que o do GDE,
  - e a oscilação ao redor do ponto de mínimo diminui.
- Tem comportamento mais próximo do GD em batelada para mini-lotes maiores.
- Para MB pequenos, pode se beneficiar de técnicas para torná-lo mais comportado e talvez convergir.
  - Essas técnicas podem ajudar a ***balancear rapidez e convergência***.

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 -> Recordings -> Laboratório #3
  - Se atentem aos prazos de entrega.
  - Instruções para resolução e entrega dos laboratórios.

Referências

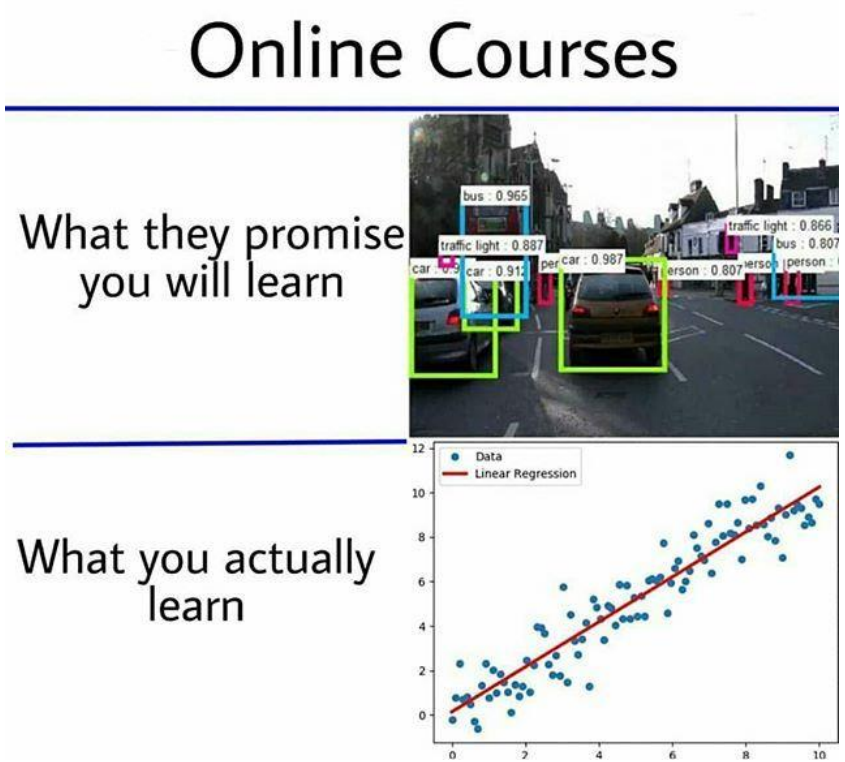
[1] Marcos Eduardo Valle, “Derivadas Direcionais e o Vetor Gradiente”, https://www.ime.unicamp.br/~valle/Teaching/MA211/Aula6.pdf

[2] Marcos Eduardo Valle, “Fatoração de Cholesky e Condicionamento de uma Matriz.” https://www.ime.unicamp.br/~valle/Teaching/MS211/Aula06.pdf

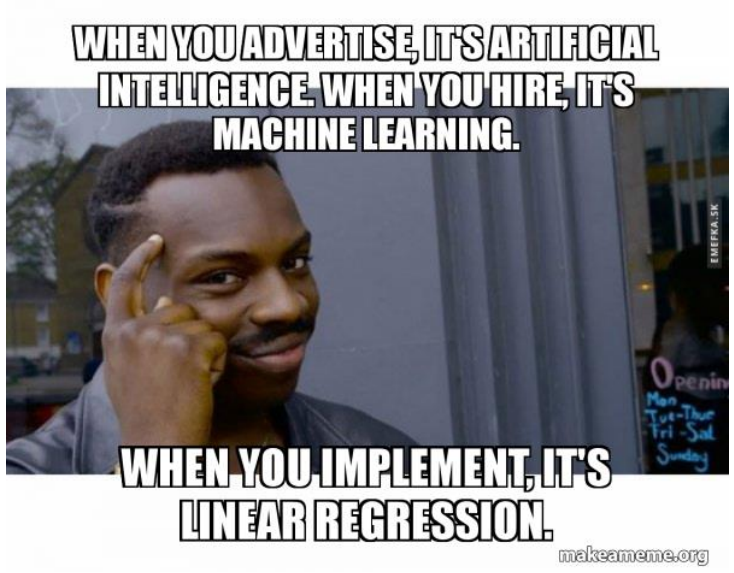
[3] IFSul, “Matrizes, Determinantes e Sistemas Lineares”, http://tics.ifsul.edu.br/matriz/conteudo/disciplinas/algl/ue/1/4.html

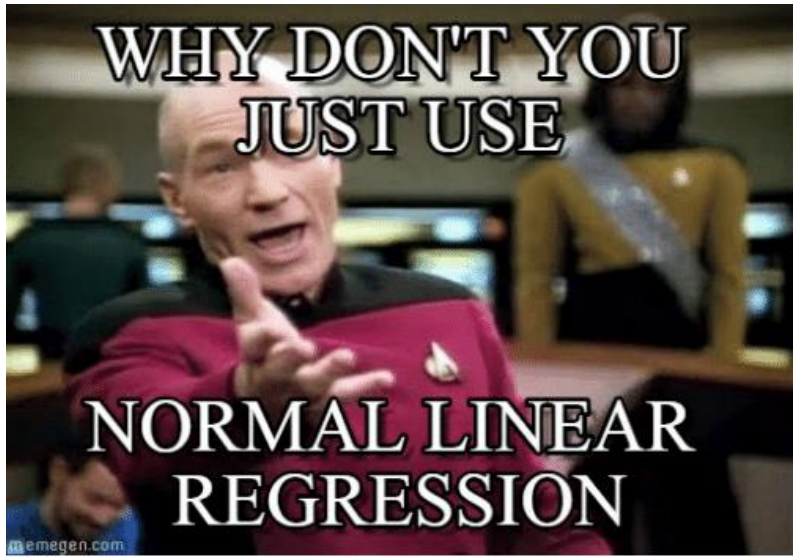
Obrigado!


Deep Learning
linear regression


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


Linear Regression
is NOT Machine Learning
CHANGE MY MIND


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

Anexo I:
Cálculo do vetor gradiente

Cálculo do vetor gradiente

Vamos considerar o hiperplano como a função hipótese

$$\hat{y}(n) = \boldsymbol{a}^T \boldsymbol{x}(n).$$

O vetor gradiente é calculado como

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = \left[\frac{\partial J_e(\boldsymbol{a})}{\partial a_0} \quad \cdots \quad \frac{\partial J_e(\boldsymbol{a})}{\partial a_K}\right]^T.$$

Assim, o vetor gradiente da função de erro em relação aos pesos é dado por

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

Cálculo do vetor gradiente

Como a operação de derivada é distributiva, podemos reescrever a equação acima como

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = \frac{1}{N}\sum_{n=0}^{N-1}\frac{\partial\left(y(n)-\hat{y}(n)\right)^2}{\partial \boldsymbol{a}}.$$

Substituindo a função hipótese na equação acima, temos

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = \frac{1}{N}\sum_{n=0}^{N-1}\frac{\partial\left(y(n)-\boldsymbol{a}^T\boldsymbol{x}(n)\right)^2}{\partial \boldsymbol{a}}.$$

Cálculo do vetor gradiente

Aplicando a regra da cadeia, reescrevemos a equação anterior como

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = -\frac{2}{N}\sum_{n=0}^{N-1}\big(y(n) - \boldsymbol{a}^T\boldsymbol{x}(n)\big)\frac{\partial \boldsymbol{a}^T\boldsymbol{x}(n)}{\partial \boldsymbol{a}}.$$

Sabendo que a derivada de $\frac{\partial \boldsymbol{a}^T\boldsymbol{x}(n)}{\partial \boldsymbol{a}}$ é igual a $\boldsymbol{x}(n)$, reescrevemos a equação anterior como

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = -\frac{2}{N}\sum_{n=0}^{N-1}[y(n) - \hat{y}(n)]\boldsymbol{x}(n)\,.$$

Cálculo do vetor gradiente

Fazendo $y(n) - \hat{y}(n) = d(n)$

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = -\frac{2}{N}\sum_{n=0}^{N-1}[y(n) - \hat{y}(n)]\boldsymbol{x}(n)$$

$$= -\frac{2}{N}\left\{ d(0)\begin{bmatrix} x_0(0) \\ \vdots \\ x_K(0) \end{bmatrix} + d(1)\begin{bmatrix} x_0(1) \\ \vdots \\ x_K(1) \end{bmatrix} + \cdots + d(N-1)\begin{bmatrix} x_0(N-1) \\ \vdots \\ x_K(N-1) \end{bmatrix} \right\}$$

$$= -\frac{2}{N}\left\{ \begin{bmatrix} d(0)x_0(0) + d(1)x_0(1) + \cdots + d(N-1)x_0(N-1) \\ \vdots \\ d(0)x_K(0) + d(1)x_K(1) + \cdots + d(N-1)x_K(N-1) \end{bmatrix} \right\}.$$

Notem que a equação acima é um ***vetor coluna*** com dimensão $K + 1 \times 1$.

Cálculo do vetor gradiente

Podemos reescrever a equação (i.e., vetor) anterior como uma multiplicação matricial

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = -\frac{2}{N}\begin{bmatrix} x_0(0) & x_0(1) & \cdots & x_0(N-1) \\ \vdots & \vdots & & \vdots \\ x_K(0) & x_K(1) & \cdots & x_K(N-1) \end{bmatrix}\begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N-1) \end{bmatrix}$$

Percebam que temos a multiplicação de uma matriz com dimensão $K + 1 \times N$ por um vetor coluna de dimensão $N \times 1$.

A matriz contém em cada linha todos os valores de $n = 0$ a $n = N - 1$ de um ***único*** atributo.

O vetor contém em cada linha a diferença $d(n) = y(n) - \hat{y}(n)$ para $n = 0$ até $n = N - 1$.

Cálculo do vetor gradiente

Se definirmos uma matriz que contém todos os N exemplos de todos os $K+1$ atributos e que tem dimensão $N \times K+1$

$$\boldsymbol{X} = \begin{bmatrix} x_0(0) & \cdots & x_K(0) \\ x_0(1) & \cdots & x_K(1) \\ \vdots & & \vdots \\ x_0(N-1) & \cdots & x_K(N-1) \end{bmatrix},$$

e dois vetores coluna com dimensões $N \times 1$ contendo todos os valores esperados (i.e., rótulos) e todos os valores preditos

$$\boldsymbol{y} = \begin{bmatrix} y(0) \\ \vdots \\ y(N-1) \end{bmatrix} \quad \text{e} \quad \hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}(0) \\ \vdots \\ \hat{y}(N-1) \end{bmatrix}$$

Cálculo do vetor gradiente

Usando a matriz e os vetores definidos no slide anterior, podemos reescrever o vetor gradiente como

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = -\frac{2}{N}\begin{bmatrix} x_0(0) & x_0(1) & \cdots & x_0(N-1) \\ \vdots & \vdots & & \vdots \\ x_K(0) & x_K(1) & \cdots & x_K(N-1) \end{bmatrix}\left(\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} - \begin{bmatrix} \hat{y}(0) \\ \hat{y}(1) \\ \vdots \\ \hat{y}(N-1) \end{bmatrix}\right)$$

$$= -\frac{2}{N}\boldsymbol{X}^T(\boldsymbol{y} - \hat{\boldsymbol{y}})$$

O resultado da multiplicação matricial acima continua resultando em um ***vetor coluna*** com dimensão $K+1 \times 1$, ou seja, $(K+1 \times N) \times (N \times 1) = K+1 \times 1$.

Equação de atualização dos pesos

Utilizando o resultado anterior, podemos reescrever a equação de atualização dos pesos como

$$\begin{aligned}\boldsymbol{a} &= \boldsymbol{a} - \alpha \frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} \\ &= \boldsymbol{a} + \alpha \frac{2}{N} \boldsymbol{X}^T (\boldsymbol{y} - \hat{\boldsymbol{y}}).\end{aligned}$$

A soma acima deve resultar em um vetor coluna com dimensão $K + 1 \times 1$, pois esta é a dimensão dos dois vetores sendo somados.

Lembrem-se que $K + 1 \times 1$ é a dimensão do vetor $\boldsymbol{a}$, o qual contém todos os pesos do modelo e que $\frac{2}{N} \boldsymbol{X}^T (\boldsymbol{y} - \hat{\boldsymbol{y}})$ tem dimensão $K + 1 \times 1$ também.

Equação de atualização dos pesos

Podemos reescrever a equação de atualização dos pesos como

$$\boldsymbol{a} = \boldsymbol{a} - \alpha \frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = \begin{bmatrix} a_0 \\ \vdots \\ a_K \end{bmatrix} - \alpha \begin{bmatrix} \dfrac{\partial J_e(\boldsymbol{a})}{\partial a_0} \\ \vdots \\ \dfrac{\partial J_e(\boldsymbol{a})}{\partial a_K} \end{bmatrix}$$

$$= \begin{bmatrix} a_0 \\ \vdots \\ a_K \end{bmatrix} + \alpha \frac{2}{N} \begin{bmatrix} x_0(0) & x_0(1) & \cdots & x_0(N-1) \\ \vdots & \vdots & & \vdots \\ x_K(0) & x_K(1) & \cdots & x_K(N-1) \end{bmatrix} \left(\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} - \begin{bmatrix} \hat{y}(0) \\ \hat{y}(1) \\ \vdots \\ \hat{y}(N-1) \end{bmatrix} \right)$$

$$= \begin{bmatrix} a_0 \\ \vdots \\ a_K \end{bmatrix} + \alpha \frac{2}{N} \left\{ \begin{bmatrix} d(0)x_0(0) + d(1)x_0(1) + \cdots + d(N-1)x_0(N-1) \\ \vdots \\ d(0)x_K(0) + d(1)x_K(1) + \cdots + d(N-1)x_K(N-1) \end{bmatrix} \right\}.$$

Anexo II:
Cálculo do vetor gradiente de uma função hipótese com 2 pesos

Cálculo do vetor gradiente

Função hipótese com 2 pesos, a_1 e a_2

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

A função de erro é dada por

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

Cálculo do vetor gradiente

Cada elemento do vetor gradiente é dado por

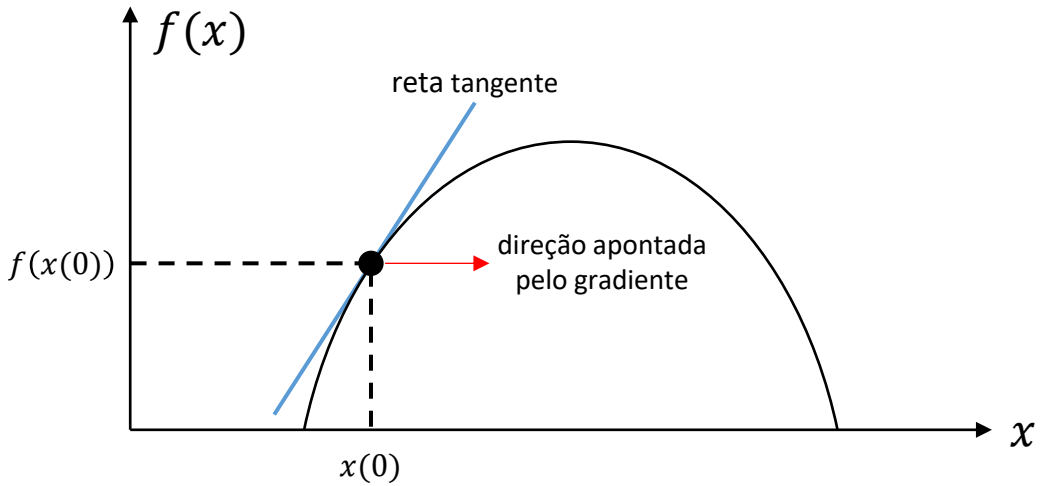
$$\frac{\partial J_e(\boldsymbol{a})}{\partial a_k} = \frac{\partial \frac{1}{N}\sum_{n=0}^{N-1}\left[y(n) - \left(a_1 x_1(n) + a_2 x_2(n)\right)\right]^2}{\partial a_k}$$

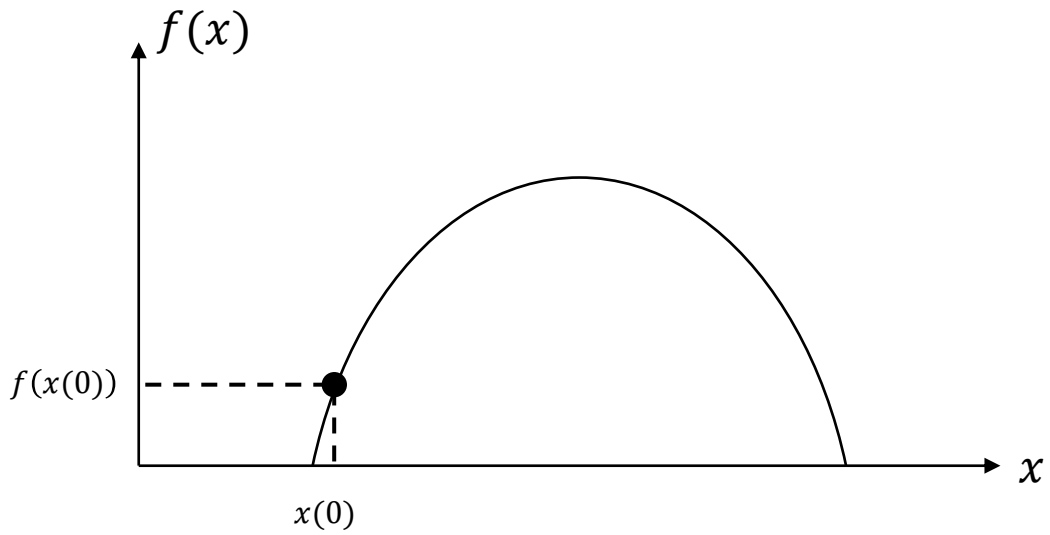
$$= \frac{1}{N}\sum_{n=0}^{N-1} \frac{\partial\left[y(n) - \left(a_1 x_1(n) + a_2 x_2(n)\right)\right]^2}{\partial a_k}$$

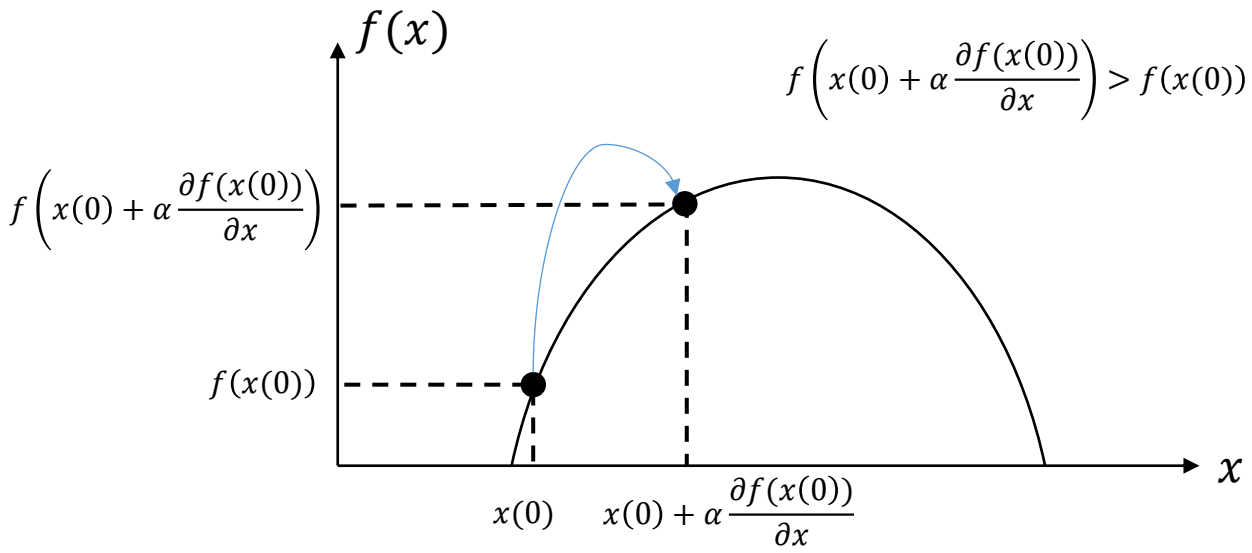
Operação da derivada parcial é distributiva.

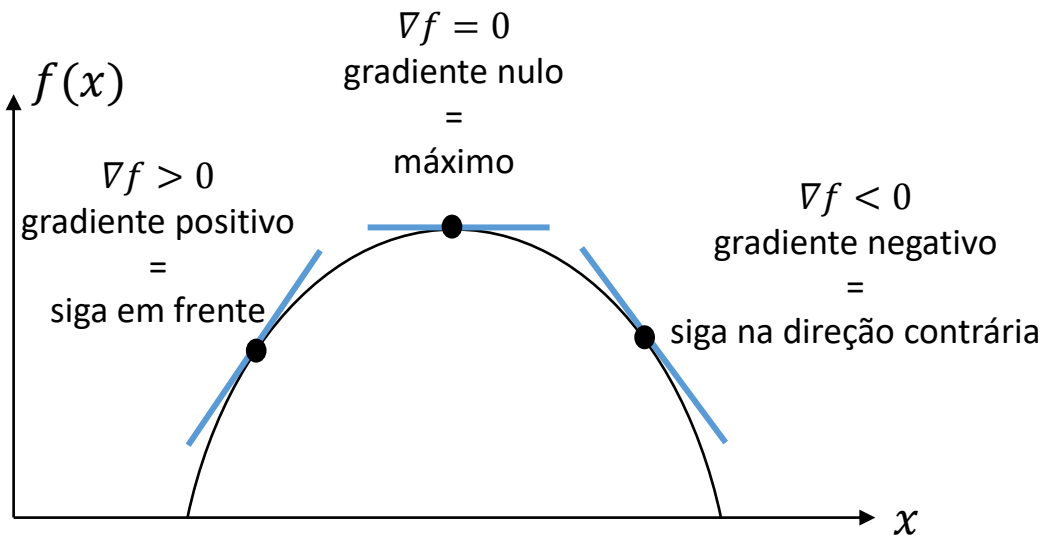
$$= -\frac{2}{N}\sum_{n=0}^{N-1}\left[y(n) - \left(a_1 x_1(n) + a_2 x_2(n)\right)\right] x_k(n)\,, k = 1,2$$

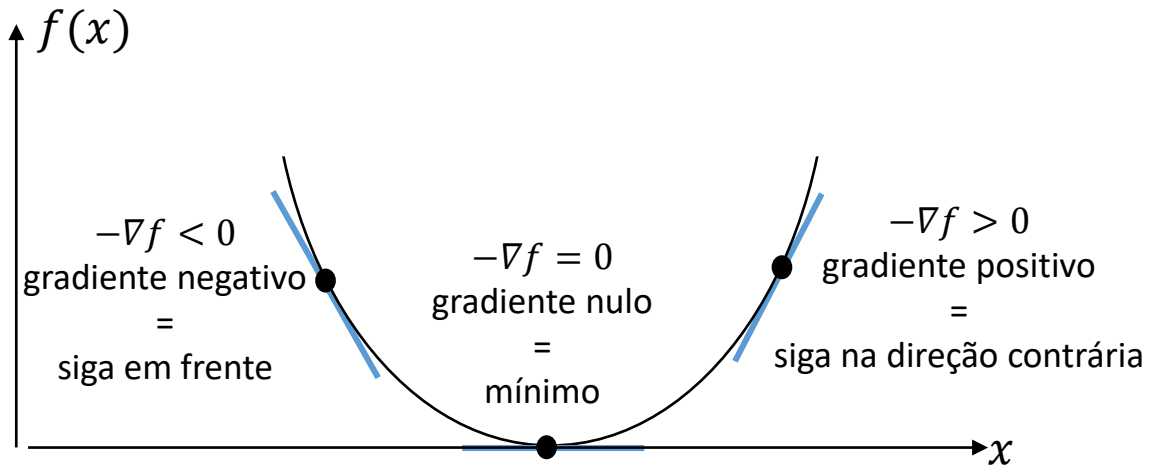
FIGURAS

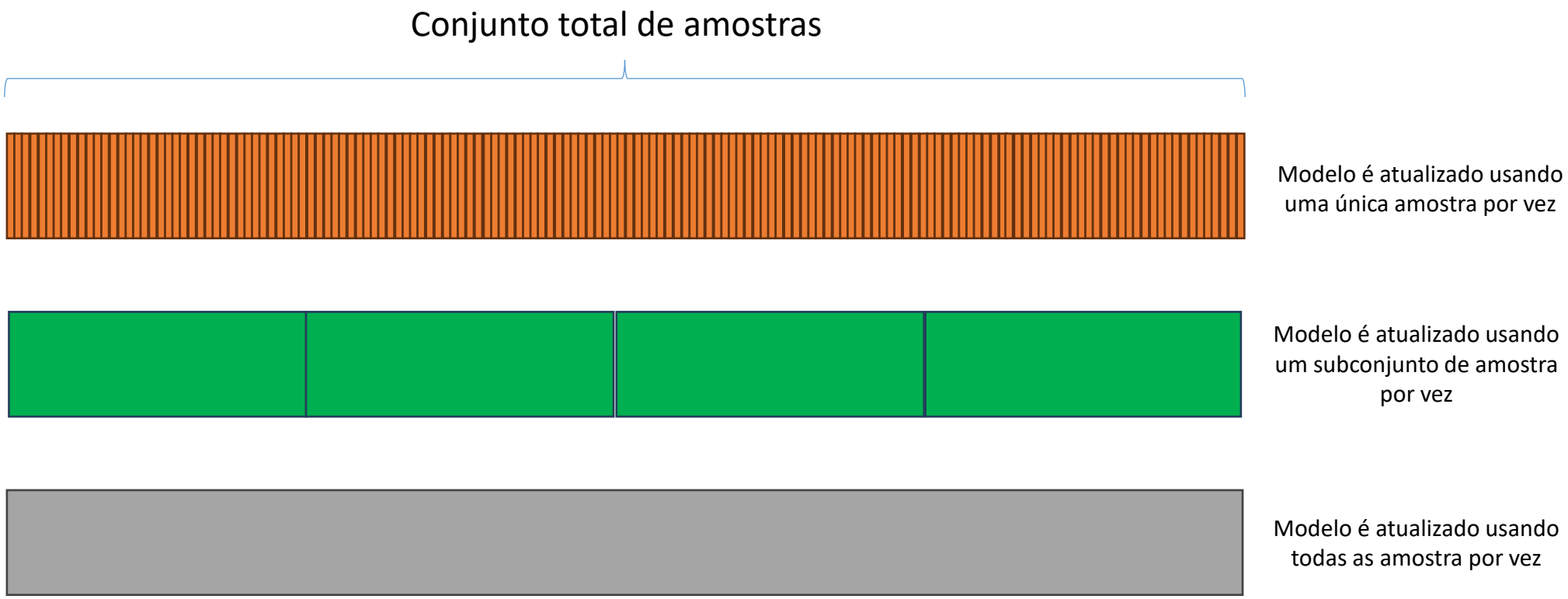

$f(x)$
reta tangente
direção apontada
pelo gradiente
$f(x(0))$
x
$x(0)$


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


Conjunto total de amostras
Modelo é atualizado usando uma única amostra por vez
Modelo é atualizado usando um subconjunto de amostra por vez
Modelo é atualizado usando todas as amostra por vez