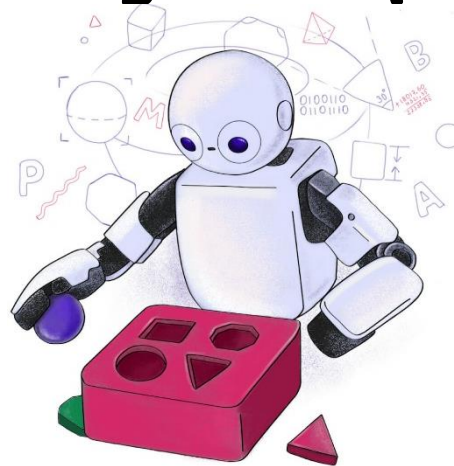


T320 - Introdução ao Aprendizado de Máquina II: *Classificação (Parte III)*



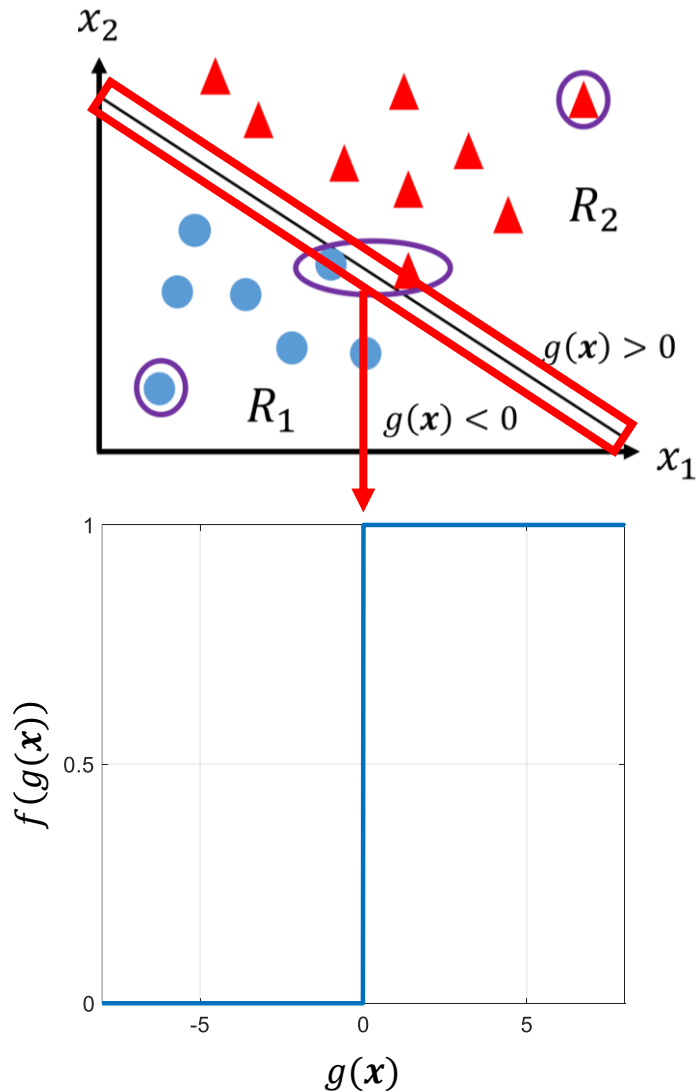
Recapitulando

- Anteriormente, nós aprendemos que a ***classificação*** pode ser feita usando-se uma ***função discriminante***, que nada mais é do que um ***polinômio***, que tem sua saída passada através de outra função, chamada de ***função de limiar***.
- Assim como na ***regressão linear***, o problema da classificação está em encontrar uma ***função discriminante*** (i.e., equação apropriada e seus respectivos pesos) que separa as classes da melhor forma possível.

Recapitulando

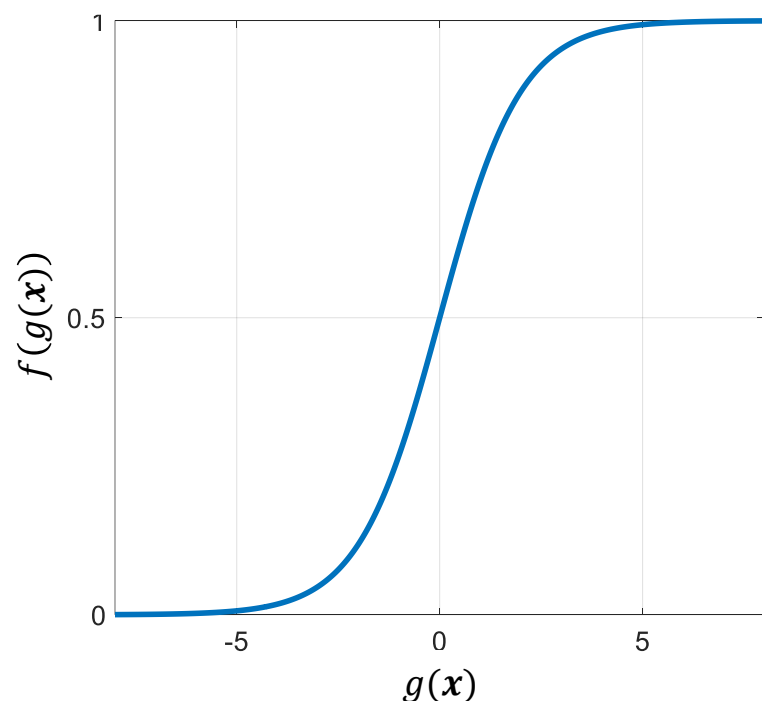
- Vimos que a **função de limiar mais simples** é a de **limiar rígido**, porém, ela apresenta alguns **problemas** como não poder ser utilizada para encontrar uma **solução em forma fechada** ou com o **gradiente descendente** e não nos dar a **confiança das predições**.
- Aprendemos também, uma forma intuitiva e iterativa de encontrar os pesos da **função discriminante** quando usamos o **limiar rígido**.
- Na sequência, introduziremos **outra função de limiar**, chamada de **função logística**, com a qual é possível encontrar uma solução eficiente com o **gradiente descendente** e termos o **grau de confiança** de uma predição feita pelo modelo.

Classificação com função de limiar logístico



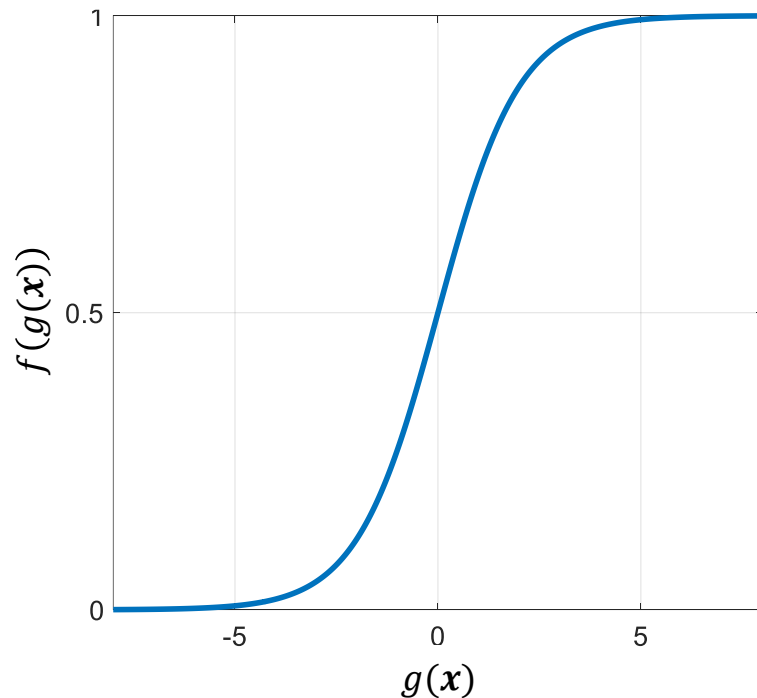
- Como discutimos anteriormente, a **função hipótese**, $h_a(x) = f(g(x))$, com **limiar de decisão rígido** é **descontínua** em $g(x) = 0$ e tem **derivada igual a zero** para todos os outros valores de $g(x)$.
- Além disso, o **classificador com limiar de decisão rígido** sempre faz **predições completamente confiantes** das classes (i.e., 0 ou 1), mesmo para exemplos muito próximos da **fronteira de decisão**.

Classificação com função de limiar logístico



- Em muitas situações, nós precisamos de **valores mais graduados**, que **indiquem incertezas** quanto à predição.
- Todos esses problemas são resolvidos com a **suavização** da **função de limiar rígido** através de sua aproximação por uma função que seja **contínua**, **diferenciável** e **assuma valores reais dentro do intervalo de 0 a 1**.
- Uma função que apresenta essas características é a **função logística**, ou **sigmoide**.

Classificação com função de limiar logístico



- A **função logística** é definida como

$$\text{Logistic}(z) = \frac{1}{1+e^{-z}} \in [0, 1].$$

- A função realiza um mapeamento

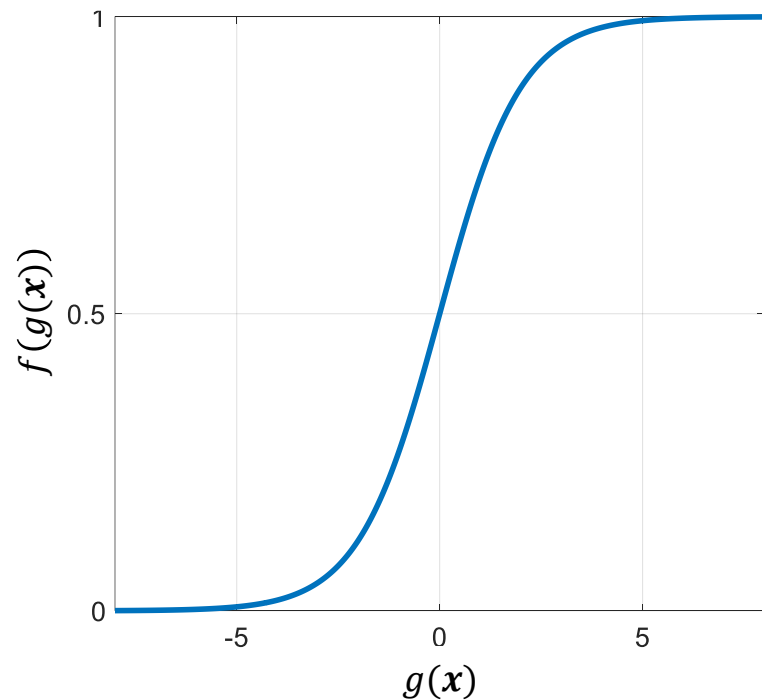
$$\mathbb{R} \rightarrow [0, 1].$$

- Utilizando a **função logística** como **função de limiar de decisão**, temos a seguinte **função hipótese**

$$h_a(x) = f(g(x)) = \frac{1}{1+e^{-g(x)}} \in [0, 1].$$

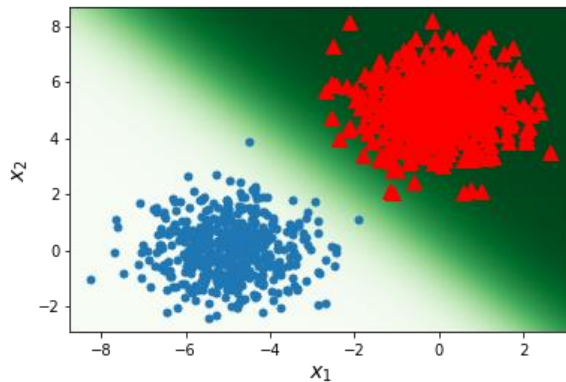
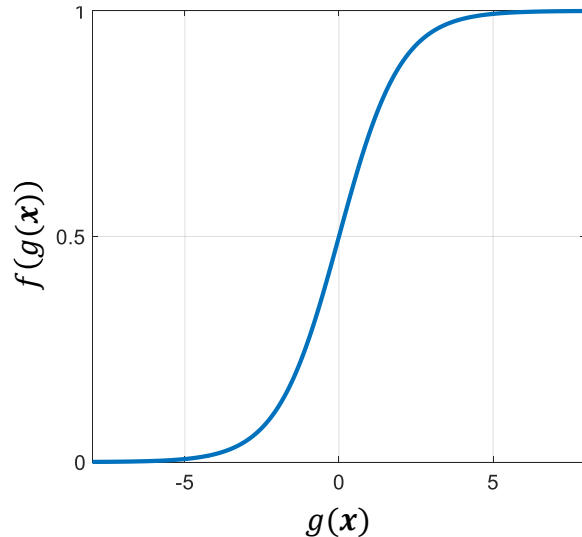
- Lembrando apenas que $g(x)$ pode assumir o formato de um **hiperplano**, de um **polinômio**, etc.

Classificação com função de limiar logístico



- A saída de $h_a(\mathbf{x})$ será um número real entre 0 e 1.
- Esse valor pode ser interpretado como a **probabilidade** de um dado exemplo de entrada **pertencer à classe positiva** (C_2).
- A **probabilidade da classe negativa** (C_1) é obtida através do complemento de $h_a(\mathbf{x})$, ou seja, $1 - h_a(\mathbf{x})$.

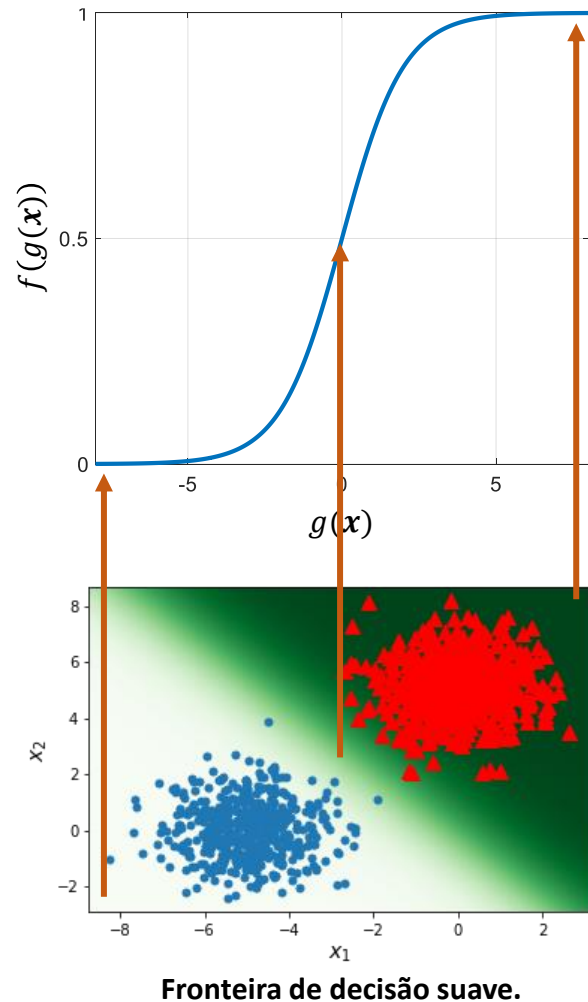
Classificação com função de limiar logístico



Fronteira de decisão suave.

- Por $f(\cdot)$ ter uma **transição suave**, a nova **função hipótese**, $h_a(\mathbf{x})$, cria uma **transição suave entre as classes**.
 - Esse comportamento é diferente do obtido com o **limiar rígido**, onde a **mudança entre classes era abrupta**.
- $h_a(\mathbf{x})$ prediz uma probabilidade de 0.5 para exemplos posicionados exatamente em cima da **fronteira de decisão**.
 - A **fronteira de decisão** é definida pela **função discriminante** e passa pelos pontos onde $g(\mathbf{x}) = 0$.
- $h_a(\mathbf{x})$ se aproxima de 0 ou 1 conforme o exemplo se distancia da **fronteira de decisão**.

Classificação com função de limiar logístico



- Lembrem-se que quanto **mais longe** da **fronteira de decisão**, **maior** será o **valor absoluto de $g(x)$** .
- Assim, quanto **mais longe** um exemplo **estiver da fronteira**, **mais próximo** o valor de $h_a(x)$ **estará de 0 ou de 1** e, portanto, mais **certeza teremos sobre uma predição**.
- Da mesma forma, quanto **mais próximo** um exemplo estiver da **fronteira de decisão**, **mais próximo** o valor de $h_a(x)$ estará de **0.5**.
 - Isso indica uma **indecisão** sobre a classe do exemplo.

Regressão logística

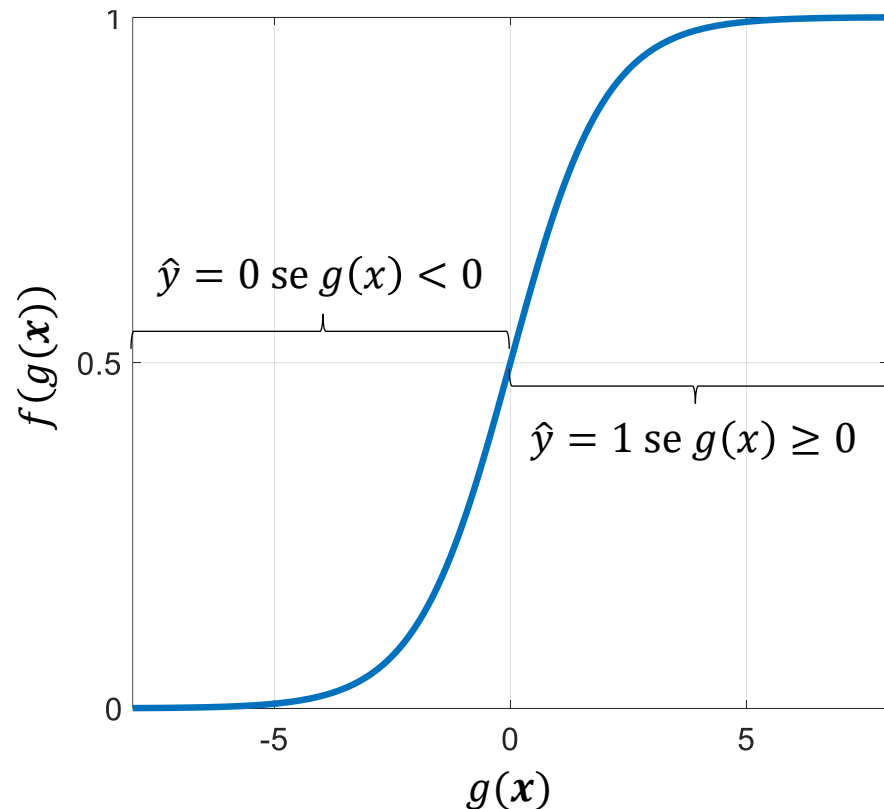
- Esse modelo que estima a probabilidade de um dado exemplo de entrada pertencer à classe positiva, *não é um classificador no sentido estrito*.
- Ele é na verdade um *regressor* e é chamado de *regressor logístico*.
- É um regressor pois sua saída pode *assumir infinitos valores* no intervalo entre 0 e 1.
 - Por exemplo, podemos treiná-lo para estimar a probabilidade de um dado email ser um spam.
- O *regressor logístico* é normalmente usado para *classificação binária* (i.e., classificação entre duas classes, C_1 e C_2), mas para isso, precisamos *quantizar sua saída*.

Regressão logística

- Se quantiza a saída da **função hipótese**, $h_a(\mathbf{x})$, nos valores 0 ou 1 estabelecendo-se um **limiar de decisão**.
- Em geral, o **limiar de decisão** é feito igual a 0.5 (i.e., 50% de probabilidade).
- Se a **probabilidade** estimada para um exemplo for igual ou maior que 50%, o classificador **prediz** que o exemplo pertence à **classe positiva**, caso contrário **prediz** que pertence à **classe negativa**.
- Ou seja, a saída **quantizada** do **regressor logístico** é dada por

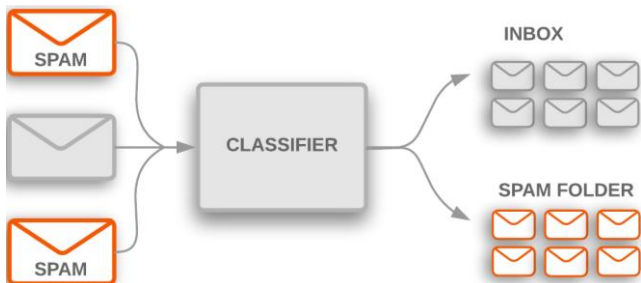
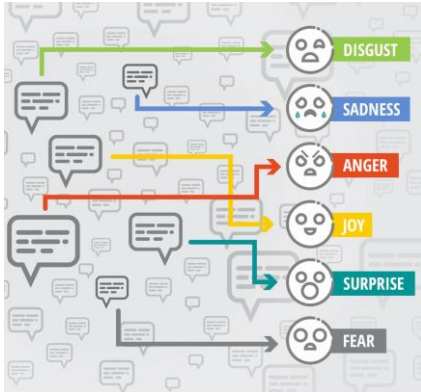
$$Classe = \hat{y} = \begin{cases} 0 \text{ (classe } C_1 \text{ – Negativa), se } h_a(\mathbf{x}) < 0.5 \\ 1 \text{ (classe } C_2 \text{ – Positiva), se } h_a(\mathbf{x}) \geq 0.5 \end{cases}$$

Regressão logística



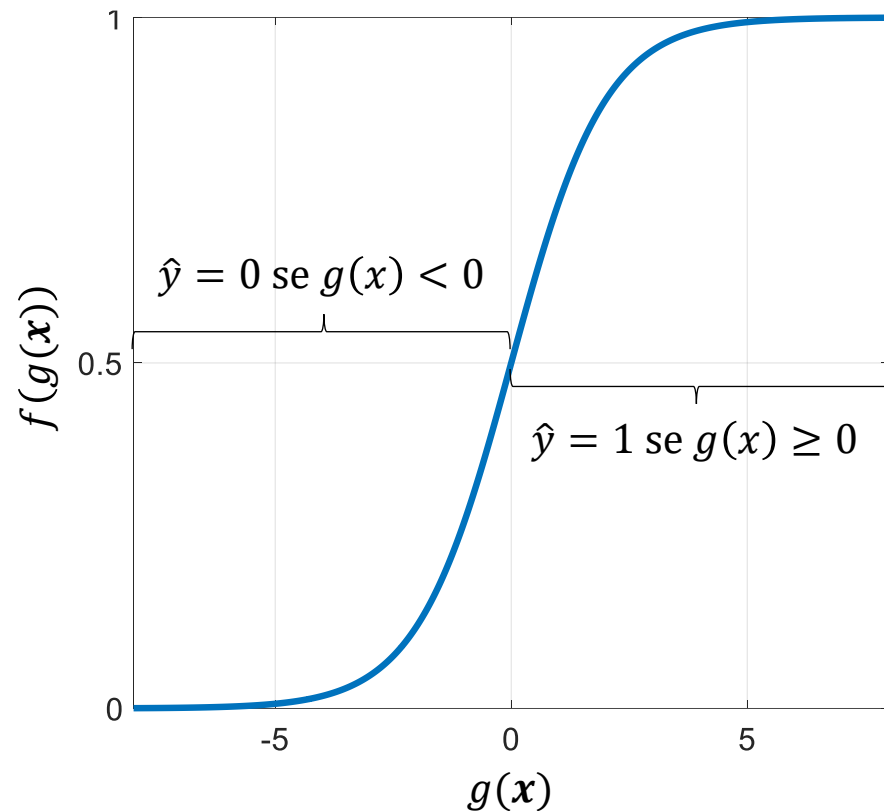
- Notem que $f(g(\mathbf{x})) < 0.5$ quando $g(\mathbf{x}) < 0$ e $f(g(\mathbf{x})) \geq 0.5$ quando $g(\mathbf{x}) \geq 0$.
- Portanto, o **regressor logístico** prediz a classe positiva, C_2 (i.e., $\hat{y} = 1$), se $g(\mathbf{x}) \geq 0$ e a classe negativa, C_1 (i.e., $\hat{y} = 0$), se $g(\mathbf{x}) < 0$.
- Nosso objetivo será encontrar uma **função discriminante apropriada e seus respectivos pesos** de forma que o **erro de classificação seja minimizado**.
- Assim, em breve, **definiremos uma função de erro** que nos ajudará a treinar o modelo.

Regressão logística



- Mesmo sendo uma técnica bastante simples, a **regressão logística** é muito utilizada em várias aplicações do mundo real em áreas como medicina, marketing, análise de crédito, etc.
 - **Exemplos:** classificar críticas de filmes, probabilidade de um paciente desenvolver uma doença, detecção de spam, classificar transações como fraudulentas ou não, etc.
- Além disto, toda a teoria por trás da **regressão logística** foi a base para a criação das **primeiras redes neurais**.

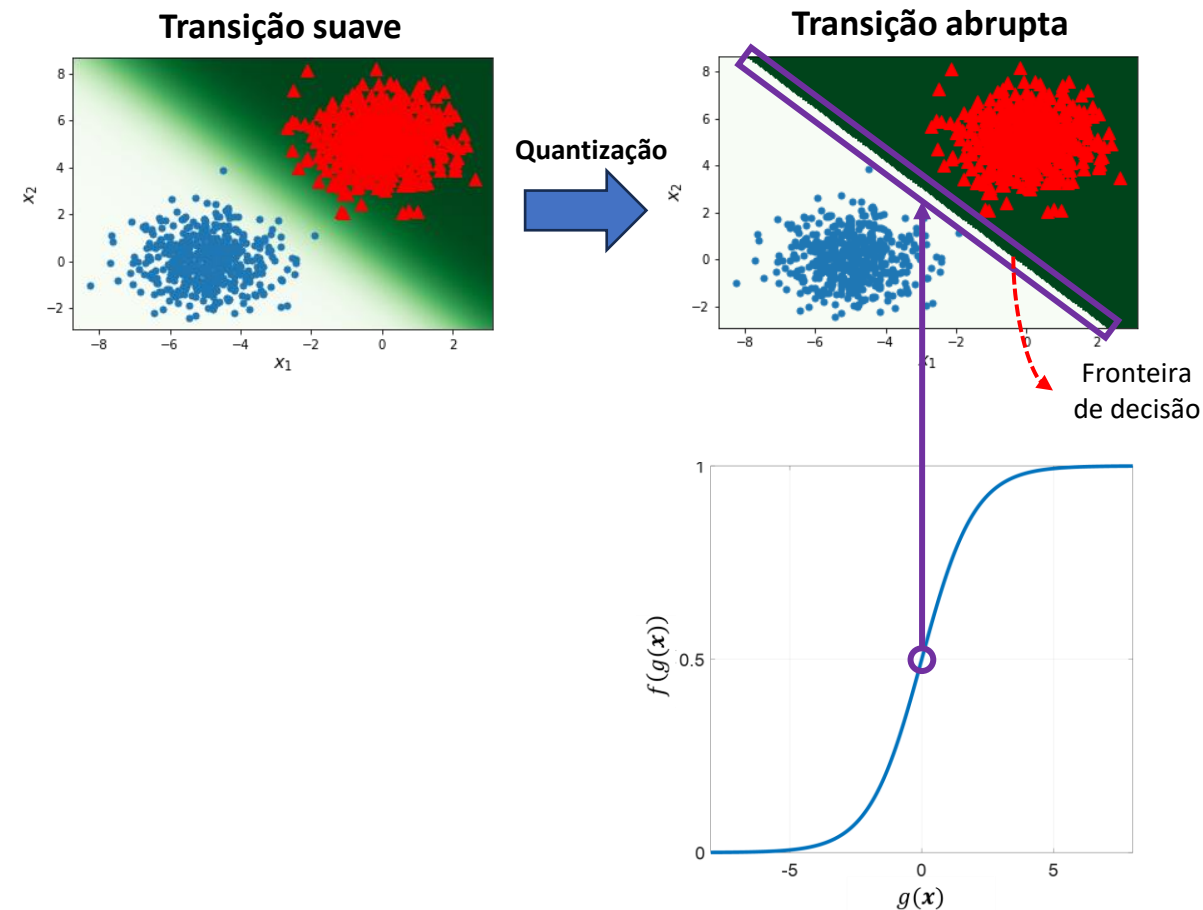
Propriedades da regressão logística



- Os valores de saída da **função hipótese**, $h_a(\mathbf{x})$, ficam restritos ao intervalo $0 \leq h_a(\mathbf{x}) \leq 1$.
- A saída de $h_a(\mathbf{x})$ representa a **probabilidade** da classe positiva (C_2) para um dado vetor de atributos \mathbf{x} e um dado vetor de pesos, \mathbf{a} .
- Ou seja, $h_a(\mathbf{x})$ dá a **probabilidade condicional da classe positiva**, C_2
$$h_a(\mathbf{x}) = P(C_2 \mid \mathbf{x}; \mathbf{a}).$$
- Consequentemente, o complemento de $h_a(\mathbf{x})$
$$(1 - h_a(\mathbf{x})) = P(C_1 \mid \mathbf{x}; \mathbf{a}),$$

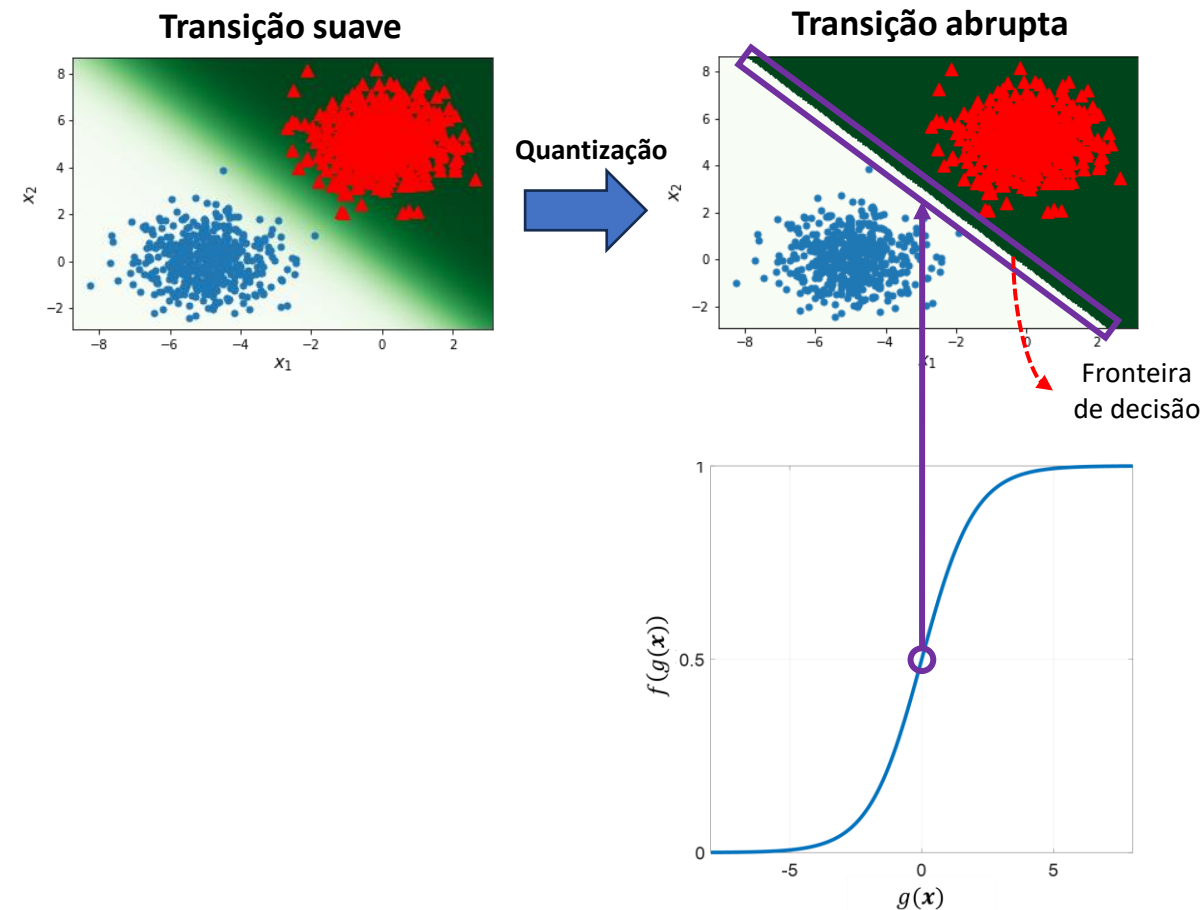
dá a **probabilidade condicional da classe negativa**, C_1 .

Propriedades da regressão logística



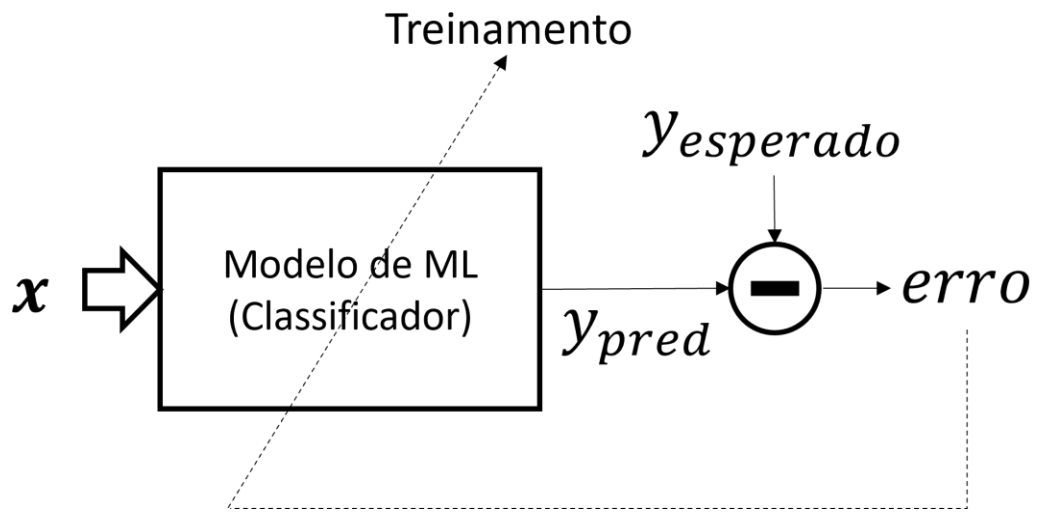
- A **transição entre classes** com o regressor logístico **é suave**, mas **após a quantização** de sua saída, ela **se torna abrupta**.
- A **fronteira de decisão** (pois após a quantização tem-se um **classificador**) é determinada quando há uma **indecisão** entre as classes.
- Ou seja, quando
$$P(C_1 | \mathbf{x}; \mathbf{a}) = P(C_2 | \mathbf{x}; \mathbf{a}),$$
que ocorre quando
$$h_{\mathbf{a}}(\mathbf{x}) = P(C_2 | \mathbf{x}; \mathbf{a}) = 0.5.$$

Propriedades da regressão logística



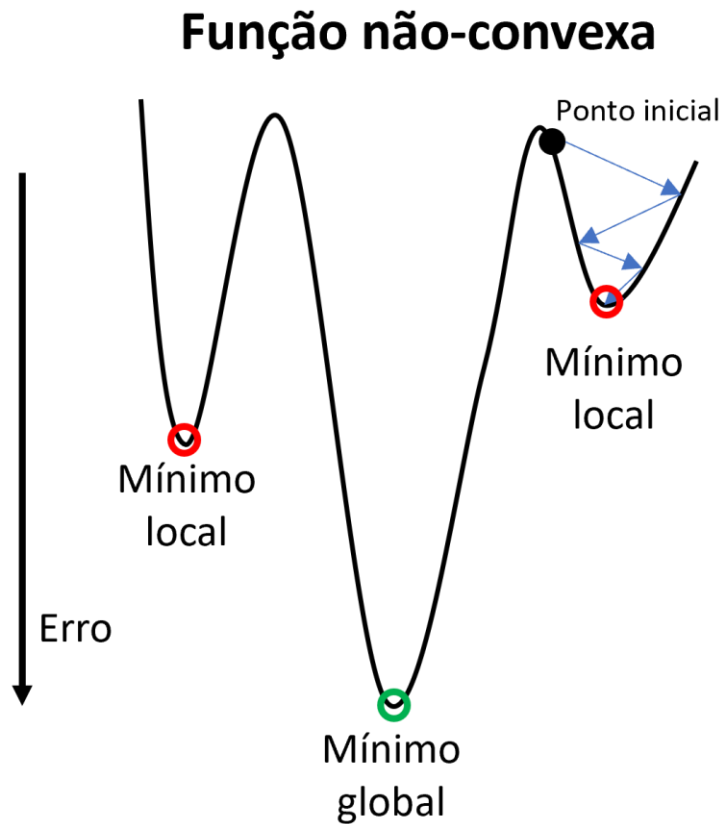
- Observando a figura da **função logística**, nós percebemos que $f(g(x)) = 0.5$ quando $g(x) = 0$.
- Ou seja, quando o vetor de atributos, x , estiver exatamente em cima da **função discriminante**, a probabilidade das classes C_1 e C_2 dado x e a é de 50%.
- Isso indica que o **classificador está indeciso**.

Função de erro



- Como discutimos antes, para **treinarmos um regressor logístico** e encontrarmos os **pesos** da **função discriminante**, nós **precisamos**, assim como fizemos com a **regressão linear**, **definir uma função de erro**.
- Porém, adotar a função do **erro quadrático médio** como **função de erro não é uma boa escolha** para a **atualização dos pesos** no caso da **regressão logística** e **classificadores em geral** como veremos a seguir.

Função de erro

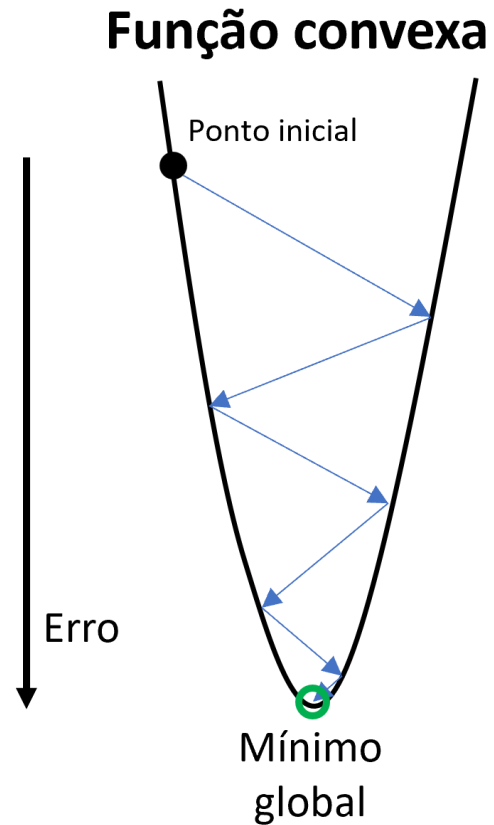


- A **função de erro**, $J_e(\mathbf{a})$, utilizando o **erro quadrático médio** é dada por

$$J_e(\mathbf{a}) = \frac{1}{N} \sum_{i=1}^N (y(i) - h_{\mathbf{a}}(\mathbf{x}))^2$$
$$= \frac{1}{N} \sum_{i=1}^N (y(i) - f(g(\mathbf{x})))^2.$$

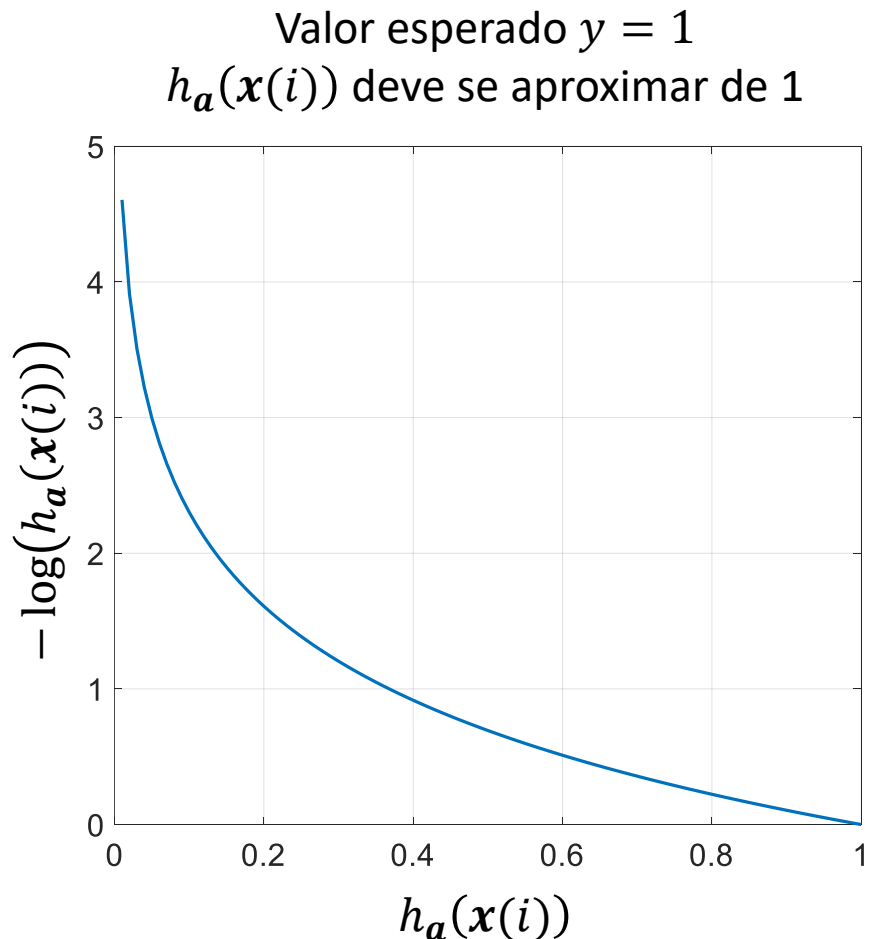
- Como $f(\cdot)$ é uma **função não-linear**, $J_e(\mathbf{a})$ **não será**, consequentemente, **uma função convexa**, de forma que a **superfície de erro** poderá apresentar **vários mínimos locais** que vão dificultar o aprendizado do modelo (e.g., o algoritmo do GD pode ficar preso em um mínimo local).

Função de erro



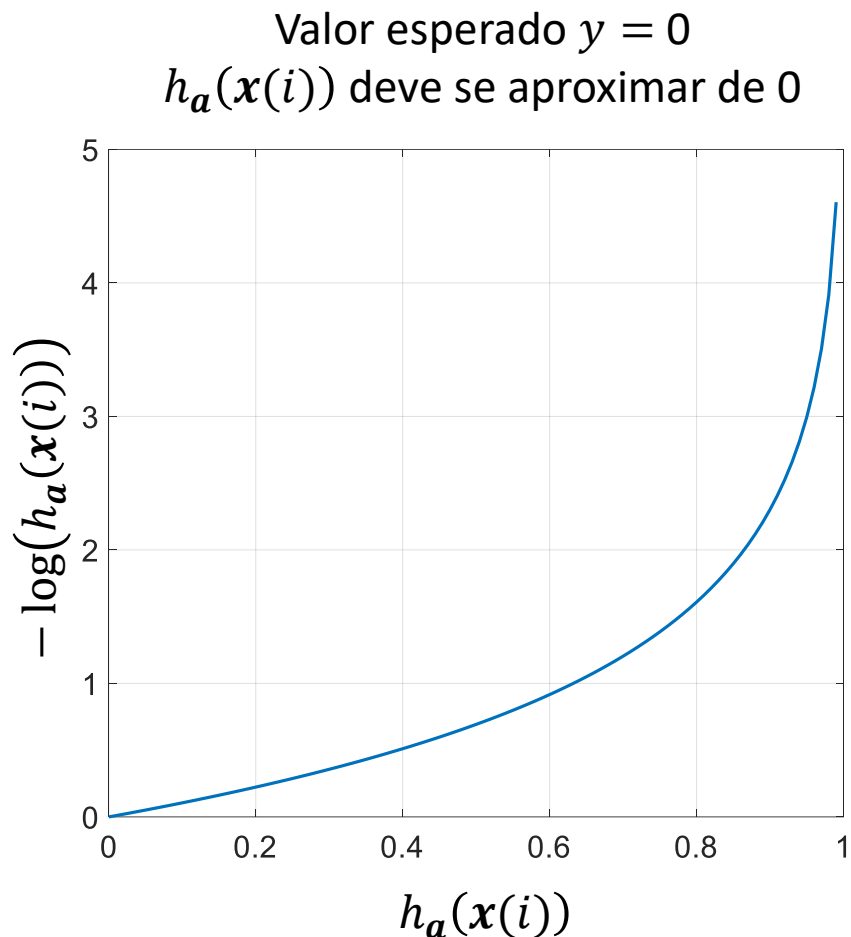
- **Ideia:** encontrar uma **função de erro** que tenha **superfície de erro** resultante **convexa**.
- Uma proposta **intuitiva** para a **função de erro para cada exemplo** de entrada é dada por
$$\begin{aligned} & \text{Erro}(h_a(\mathbf{x}(i)); y(i)) \\ &= \begin{cases} -\log(h_a(\mathbf{x}(i))), & \text{se } y(i) = 1 \\ -\log(1 - h_a(\mathbf{x}(i))), & \text{se } y(i) = 0 \end{cases} \end{aligned}$$
onde $y(i)$ é o i -ésimo valor esperado (i.e., rótulo).
- Veremos a seguir uma justificativa para esta escolha.

Função de erro



- O valor de $-\log(h_a(\mathbf{x}(i)))$ se torna muito grande quando $h_a(\mathbf{x}(i))$ se aproxima de 0.
- Assim, o erro será grande se o regressor estimar uma probabilidade próxima a 0 para um exemplo da classe positiva, C_2 .
- Por outro lado, $-\log(h_a(\mathbf{x}(i)))$ se torna próximo de 0 quando $h_a(\mathbf{x}(i))$ se aproxima de 1.
- Portanto, o erro será próximo de 0 se a probabilidade estimada for próxima de 1 para um exemplo da classe positiva, C_1 .

Função de erro



- O valor de $-\log(1 - h_a(\mathbf{x}(i)))$ será muito grande se o classificador estimar uma probabilidade próxima de 1 para um exemplo da classe negativa, C_1 .
- Porém, o valor de $-\log(1 - h_a(\mathbf{x}(i)))$ se torna próximo de 0 quando $h_a(\mathbf{x}(i))$ se aproxima de 0.
- Portanto, o erro será próximo de 0 para um exemplo da classe negativa.

Função de erro

- Nós podemos unir a **função de erro para cada exemplo** em uma expressão única **usando a informação dos rótulos**

$$\begin{aligned} & \text{Erro} \left(h_a(\mathbf{x}(i)); y(i) \right) \\ &= \underbrace{-y(i) \log \left(h_a(\mathbf{x}(i)) \right)}_{\text{Só exerce influência no erro se } y(i)=1} \underbrace{-(1 - y(i)) \log \left(1 - h_a(\mathbf{x}(i)) \right)}_{\text{Só exerce influência no erro se } y(i)=0}. \end{aligned}$$

- Assim, podemos definir a seguinte **função de erro médio**

$$J_e(\mathbf{a}) = -\frac{1}{N} \sum_{i=0}^{N-1} y(i) \log \left(h_a(\mathbf{x}(i)) \right) + (1 - y(i)) \log \left(1 - h_a(\mathbf{x}(i)) \right).$$

- Lembrem-se que sempre queremos minimizar o erro ao longo de todo o conjunto de treinamento, por isso tomamos a média.
- Essa função de erro é conhecida na literatura como **entropia cruzada**.

Função de erro

- Uma má notícia com relação a essa função é que **não existe uma equação de forma fechada** para encontrar os **pesos** que minimizem essa **função de erro**.
- Ou seja, não há um equivalente da **equação normal**.
- Entretanto, uma boa notícia é que essa **função de erro** é **convexa** e **derivável**.
- Consequentemente, **conseguimos** calcular o **vetor gradiente da função de erro** com relação aos pesos e **implementar** o algoritmo do **gradiente descendente** (GD).
- Portanto, é garantido que GD encontre o **mínimo global** (dado que a **taxa de aprendizagem** não seja muito grande e se espere tempo suficiente).

Processo de treinamento

- Da mesma forma como fizemos com a **regressão linear**, usamos o algoritmo do **gradiente descendente** para encontrar os **pesos** que **minimizam a função de erro médio**.

- A **atualização iterativa** dos **pesos** é dada por

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

- O **vetor gradiente** da **função de erro médio** é dado por

$$\frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}} = -\frac{1}{N} \sum_{i=0}^{N-1} [y(i) - h_{\mathbf{a}}(\mathbf{x}(i))] \mathbf{x}(i) = -\frac{1}{N} \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}}).$$

Forma matricial

onde \mathbf{X} é a matriz de atributos com dimensão $N \times K + 1$, \mathbf{y} e $\hat{\mathbf{y}}$ são vetores coluna com dimensão $N \times 1$ contendo todos os valores esperados e preditos pelo regressor logístico, respectivamente.

Processo de treinamento

$$\frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}} = -\frac{1}{N} \sum_{i=0}^{N-1} [y(i) - h_{\mathbf{a}}(\mathbf{x}(i))] \mathbf{x}(i) = -\frac{1}{N} \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}})$$

- Para encontrar o vetor gradiente, a função $g(\mathbf{x})$ foi considerada como sendo a equação de um **hiperplano**, mas o resultado pode ser diretamente estendido para polinômios.
- Percebam que o **vetor gradiente** da **função de erro médio** para a **regressão logística** é idêntico (exceto pela constante 2) àquele obtido para a **regressão linear** utilizando a função de **erro quadrático médio**.
- O **vetor gradiente** da **função de erro médio** vai variar dependendo da **função discriminante** adotada.
- Na sequência, veremos alguns exemplos com diferentes $g(\mathbf{x})$.

Vetor gradiente

- O **vetor gradiente** da **função de erro médio** quando $g(x) = a_0 + a_1x_1 + a_2x_2$ (equação de uma **reta**) é dado por

$$\frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}} = -\frac{1}{N} \sum_{i=0}^{N-1} [y(i) - h_a(\mathbf{x}(i))] \mathbf{x}(i) = -\frac{1}{N} \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}}),$$

onde $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2] \in \mathbb{R}^{N \times K+1}$, $\mathbf{x}_0, \mathbf{x}_1$, e $\mathbf{x}_2 \in \mathbb{R}^{N \times 1}$ e \mathbf{y} e $\hat{\mathbf{y}} \in \mathbb{R}^{N \times 1}$.

- O **vetor gradiente** da **função de erro médio** quando $g(x) = a_0 + x_1^2 + x_2^2$ (equação de um **círculo**) é dado por

$$\frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}} = -\frac{1}{N} \sum_{i=0}^{N-1} [y(i) - h_a(\mathbf{x}(i))] \mathbf{x}(i) = -\frac{1}{N} \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}}),$$

onde $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1^2, \mathbf{x}_2^2] \in \mathbb{R}^{N \times K+1}$, $\mathbf{x}_0, \mathbf{x}_1^2$, e $\mathbf{x}_2^2 \in \mathbb{R}^{N \times 1}$ e \mathbf{y} e $\hat{\mathbf{y}} \in \mathbb{R}^{N \times 1}$.

Vetor gradiente

- O **vetor gradiente** da **função de erro médio** quando $g(\mathbf{x}) = a_0 + a_1 x_1 * x_2$ (equação de uma **hipérbole retangular**) é dado por

$$\frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}} = -\frac{1}{N} \sum_{i=0}^{N-1} [y(i) - h_{\mathbf{a}}(\mathbf{x}(i))] \mathbf{x}(i) = -\frac{1}{N} \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}}),$$

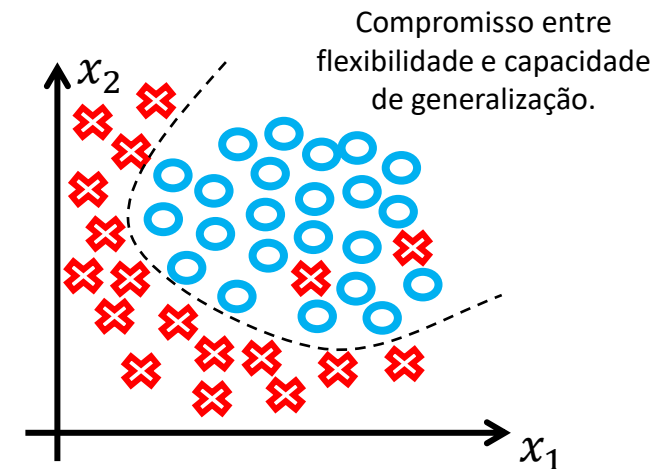
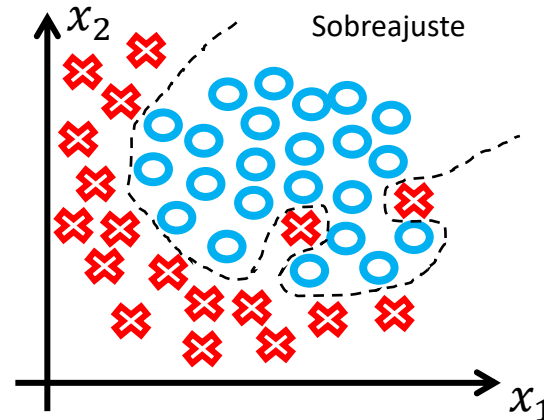
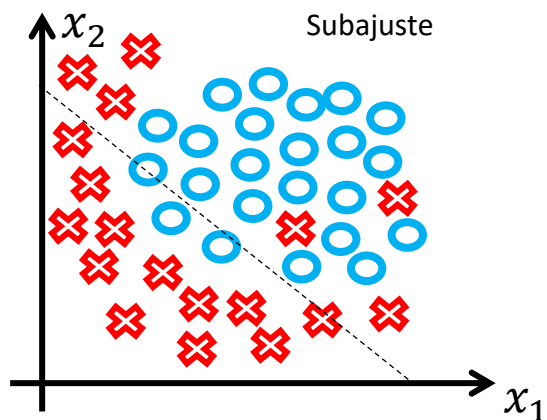
onde $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1 \odot \mathbf{x}_2] \in \mathbb{R}^{N \times K+1}$, $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$, e $\mathbf{x}_1 \odot \mathbf{x}_2 \in \mathbb{R}^{N \times 1}$, \mathbf{y} e $\hat{\mathbf{y}} \in \mathbb{R}^{N \times 1}$ e \odot é a multiplicação elemento-a-elemento.

- Agora, de posse do **vetor gradiente**, podemos usá-lo com o **gradiente descendente** (nas versões em batelada, estocástico ou mini-batch) para atualizar os pesos.

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

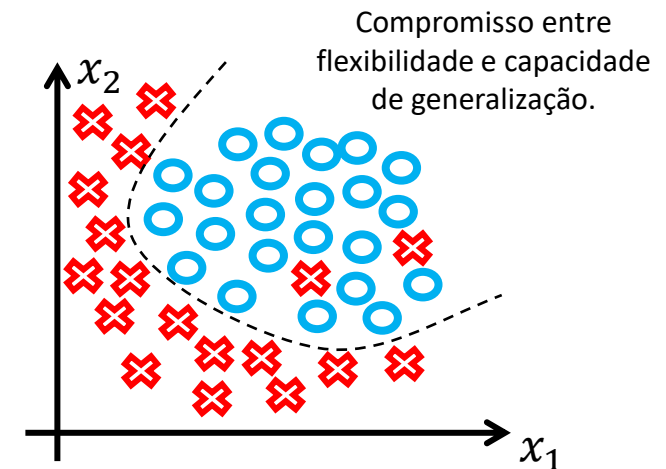
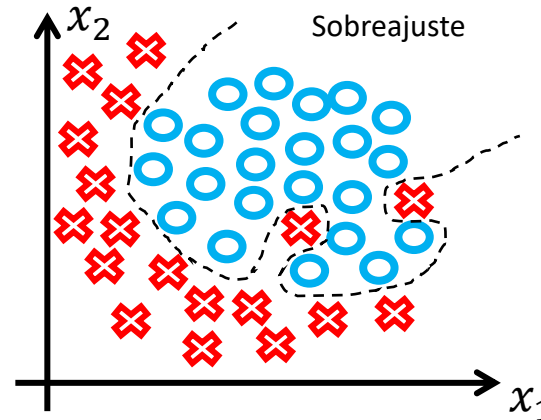
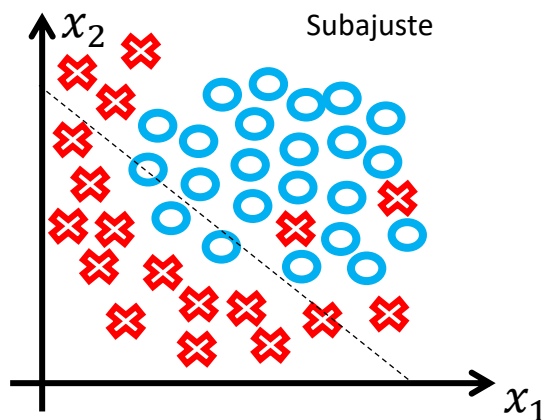
Subajuste e sobreajuste

- Como vimos, a **função discriminante**, $g(\mathbf{x})$, pode assumir também o formato de um **polinômio**.
- Porém, muitas vezes, nós não sabemos qual a **melhor ordem** para este polinômio.
- Assim, como ocorre no caso da **regressão linear**, modelos de **regressão logística** também estão sujeitos à ocorrência de **sobreajuste** e **subajuste**.



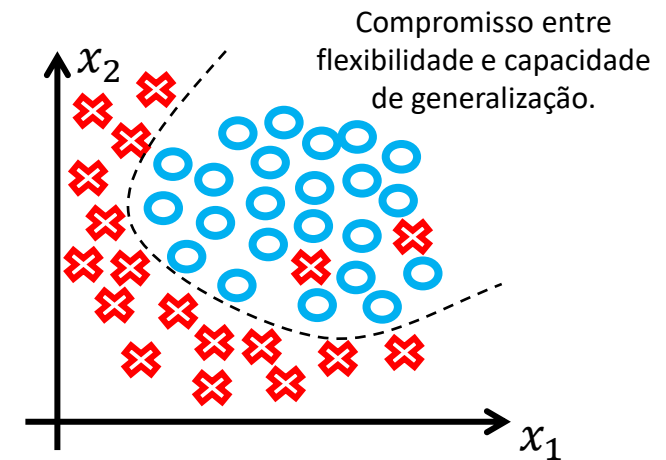
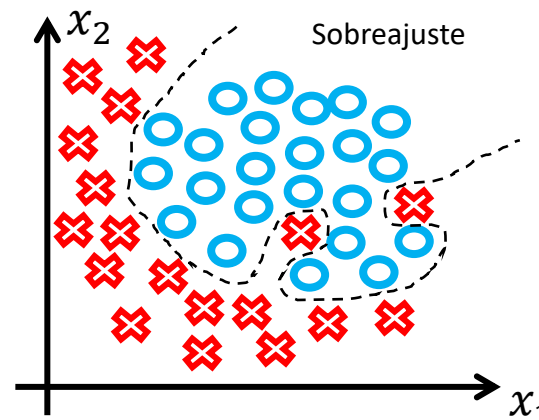
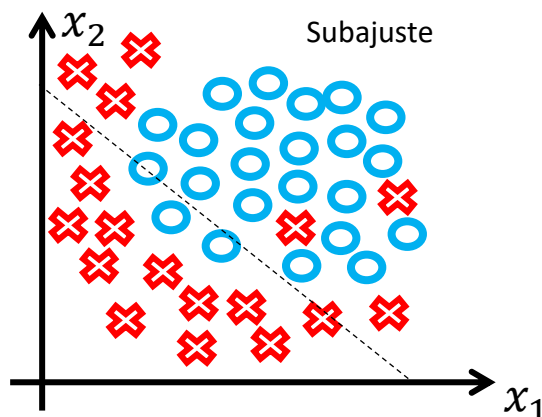
Subajuste e sobreajuste

- Na primeira figura, o polinômio (i.e., reta) tem uma **baixa flexibilidade**.
- Consequentemente, o modelo apresenta uma **baixa capacidade de generalização**.
- Assim, os erros nos conjuntos de treinamento e teste seriam **ambos altos**.
- Ou seja, o classificador não classificaria bem nenhum dos exemplos coletados.



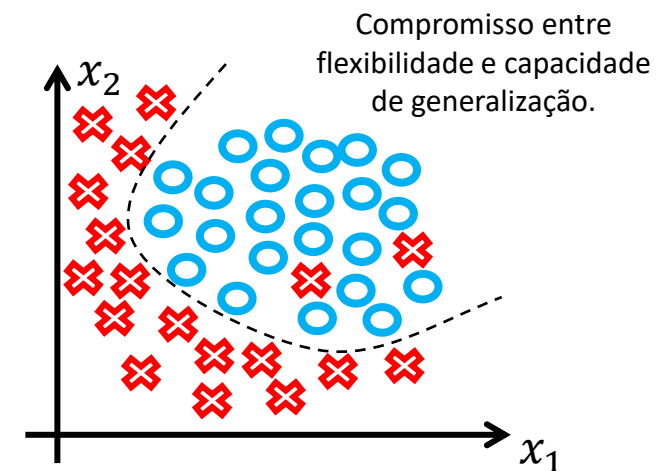
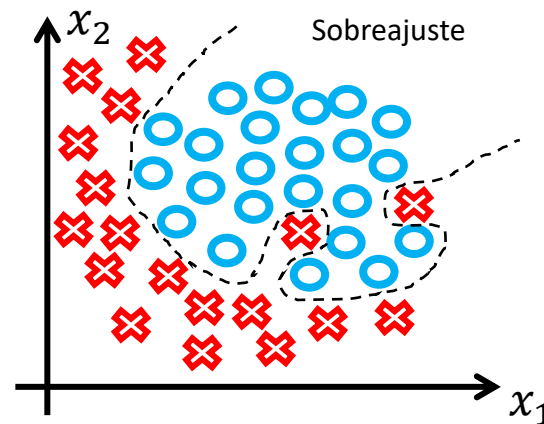
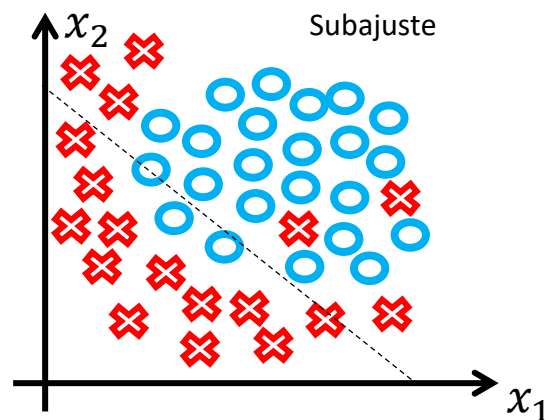
Subajuste e sobreajuste

- Na segunda figura, a **flexibilidade excessiva** do polinômio (i.e., ordem elevada) dá origem a contorções na **fronteira de decisão** na tentativa de **minimizar o erro** de classificação **junto aos dados de treinamento**.
- Porém, o classificador cometerá muitos erros para dados inéditos, ou seja, **não irá generalizar bem**.
- O classificador apresenta erro de **treinamento muito baixo** e de **erro de teste muito alto**.



Subajuste e sobreajuste

- Já a última figura mostra o que seria uma boa **hipótese de classificação**.
- O modelo apresenta uma **boa relação de compromisso** entre **flexibilidade** do polinômio e **capacidade de generalização**.
- Os erros nos conjuntos de treinamento e teste seriam **baixos e próximos**.



Como evitamos o subajuste e sobreajuste?

- Quando não conhecemos a **melhor ordem para o polinômio** da **função discriminante**, $g(x)$, devemos usar **técnicas de validação cruzada** (e.g., *holdout*, *k-fold* ou *leave-p-out*)
- Essas técnicas nos auxiliam a encontrar um polinômio que apresente um **compromisso entre flexibilidade e capacidade de generalização**, evitando assim os dois problemas.
- Uma forma de **evitar apenas problemas de sobreajuste** é usar **técnicas de regularização** (e.g., LASSO, Ridge, Elastic-Net, Early-stop).
- A regularização **reduz a flexibilidade** do modelo por meio de penalizações aplicadas a seus pesos.
 - Modelos que **sobreajustam** têm **pesos com valores absolutos muito altos**.
 - O que as técnicas de regularização fazem é **restringir o aumento** dos pesos.

Tarefas

- **Quiz:** “*T320 - Quiz - Classificação (Parte III)*” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #3](#).
 - Pode ser acessado através do link acima (Google Colab) ou no GitHub.
 - 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

- Antes de encontrarmos o **vetor gradiente** de $J_e(\mathbf{a})$, vamos reescrever a **função de erro** utilizando as seguintes equivalências

$$\log(h_{\mathbf{a}}(\mathbf{x}(i))) = \log\left(\frac{1}{1+e^{-\mathbf{x}(i)^T \mathbf{a}}}\right) = -\log\left(1 + e^{-\mathbf{x}(i)^T \mathbf{a}}\right),$$

$$\log(1 - h_{\mathbf{a}}(\mathbf{x}(i))) = \log\left(1 - \frac{1}{1+e^{-\mathbf{x}(i)^T \mathbf{a}}}\right) = -\mathbf{x}(i)^T \mathbf{a} - \log\left(1 + e^{-\mathbf{x}(i)^T \mathbf{a}}\right).$$

- Assim, a nova expressão para a **função de erro médio** é dada por

$$\begin{aligned} J_e(\mathbf{a}) &= -\frac{1}{N} \sum_{i=0}^{N-1} -y(i) \log\left(1 + e^{-\mathbf{x}(i)^T \mathbf{a}}\right) + (1 \\ &\quad - y(i)) \left[-\mathbf{x}(i)^T \mathbf{a} - \log\left(1 + e^{-\mathbf{x}(i)^T \mathbf{a}}\right) \right] \end{aligned}$$

Encontrando o vetor gradiente

- O termo $-y(i) \log(1 + e^{-x(i)^T \mathbf{a}})$ é cancelado com um dos elementos gerados a partir do produto envolvido no segundo termo, de forma que

$$J_e(\mathbf{a}) = -\frac{1}{N} \sum_{i=0}^{N-1} -\mathbf{x}(i)^T \mathbf{a} + y(i) \mathbf{x}(i)^T \mathbf{a} - \log(1 + e^{-x(i)^T \mathbf{a}}).$$

- Se $-\mathbf{x}(i)^T \mathbf{a} = -\log(e^{x(i)^T \mathbf{a}})$, então

$$-\mathbf{x}(i)^T \mathbf{a} - \log(1 + e^{-x(i)^T \mathbf{a}}) = -\log(1 + e^{x(i)^T \mathbf{a}}).$$

- Desta forma, a **função de erro médio** se torna

$$J_e(\mathbf{a}) = -\frac{1}{N} \sum_{i=0}^{N-1} y(i) \mathbf{x}(i)^T \mathbf{a} - \log(1 + e^{x(i)^T \mathbf{a}}).$$


- Em seguida, encontramos o **vetor gradiente** de cada termo da equação acima.

Encontrando o vetor gradiente

- Assim, o **vetor gradiente** do primeiro termo da equação anterior é dado por

$$\frac{\partial [y(i)\mathbf{x}(i)^T \mathbf{a}]}{\partial \mathbf{a}} = y(i)\mathbf{x}(i)$$

- O **vetor gradiente** do segundo termo da equação anterior é dado por

$$\begin{aligned} \frac{\partial \left[\log \left(1 + e^{\mathbf{x}(i)^T \mathbf{a}} \right) \right]}{\partial \mathbf{a}} &= \frac{1}{1 + e^{\mathbf{x}(i)^T \mathbf{a}}} e^{\mathbf{x}(i)^T \mathbf{a}} \mathbf{x}(i) \\ &= \frac{1}{1 + e^{-\mathbf{x}(i)^T \mathbf{a}}} \mathbf{x}(i) \\ &= h_{\mathbf{a}}(\mathbf{x}(i)) \mathbf{x}(i). \end{aligned}$$



- Usamos a **regra da cadeia** para encontrar o vetor gradiente do segundo termo.

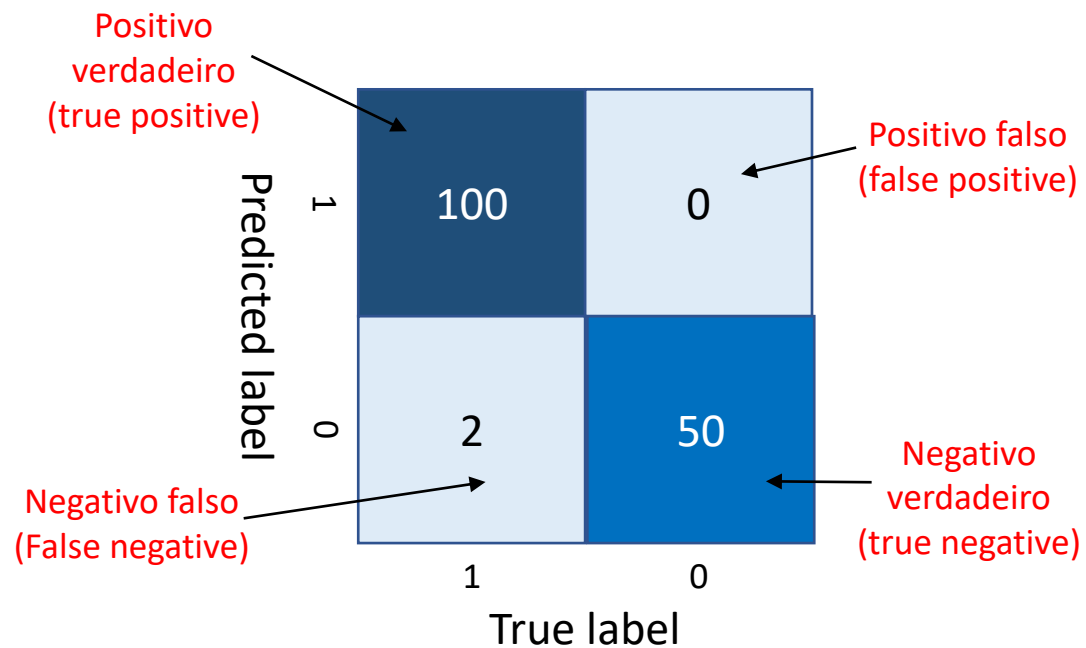
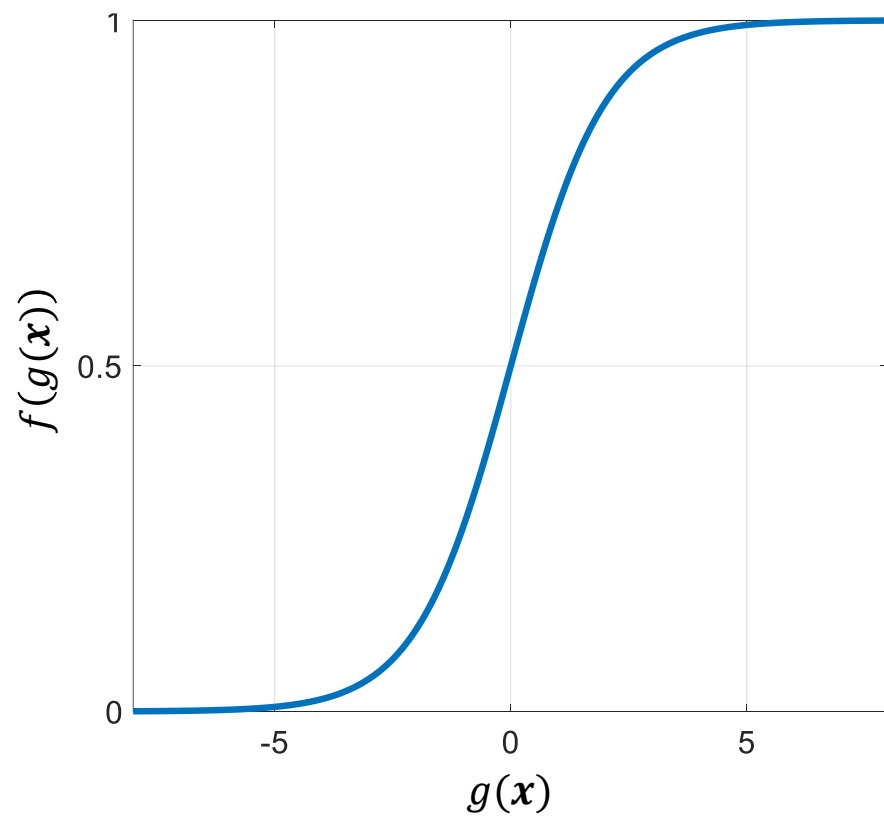
Encontrando o vetor gradiente

- Portanto, combinando os 2 resultados anteriores, temos que o **vetor gradiente** da **função de erro médio** é dado por

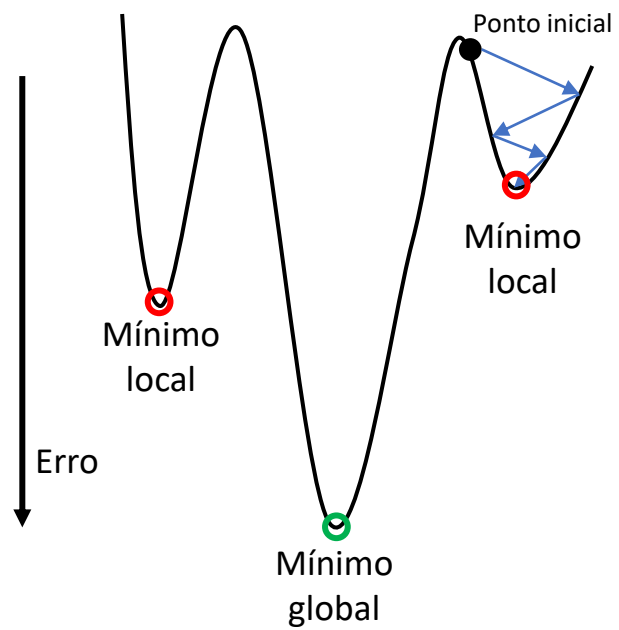
$$\frac{\partial J_e(\mathbf{a})}{\partial \mathbf{a}} = -\frac{1}{N} \sum_{i=0}^{N-1} [y(i) - h_a(\mathbf{x}(i))] \mathbf{x}(i) = -\frac{1}{N} \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}}).$$

Forma matricial:
 $\mathbf{X} \in \mathbb{R}^{N \times K+1}$, \mathbf{y}
e $\hat{\mathbf{y}} \in \mathbb{R}^{N \times 1}$





Função não-convexa



Função convexa

