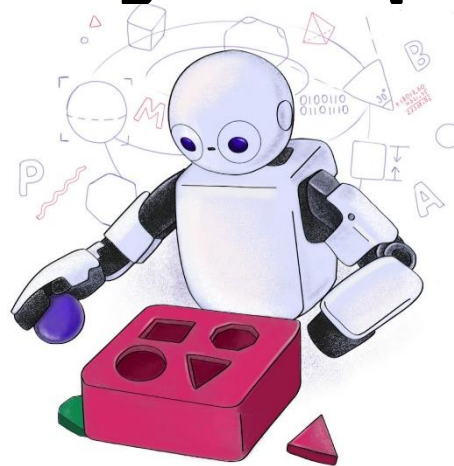


T320 - Introdução ao Aprendizado de Máquina II: *Classificação (Parte III)*



Resumindo

- Anteriormente, 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**.
- Como na **regressão linear**, o problema da classificação está em encontrar os pesos da **função discriminante** de tal forma que as classes sejam separadas da melhor forma possível.
- 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 **gradiente descendente** e não nos dar a **confiança de um resultado** de classificação.
- 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 se encontrar uma solução eficiente com o **gradiente descendente** e termos o **grau de confiança** de uma classificação.

Classificação com função de limiar logístico

- Como discutimos anteriormente, a **função hipótese**, $h_a(\mathbf{x}) = f(g(\mathbf{x}))$, com **limiar de decisão rígido** é descontínua em $g(\mathbf{x}) = 0$ e tem derivada igual a zero para todos os outros valores de $g(\mathbf{x})$.
- Além disso, o **classificador** sempre faz **previsões** completamente confiantes das classes (i.e., 0 ou 1), mesmo para exemplos muito próximos da **fronteira de decisão**.
- Em muitas situações, nós precisamos de previsões mais graduadas, que indiquem incertezas quanto à classificação.
- Todos esses problemas podem ser 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.

Classificação com função de limiar logístico

- A **função logística** (ou **sigmóide**), mostrada na figura ao lado e definida como

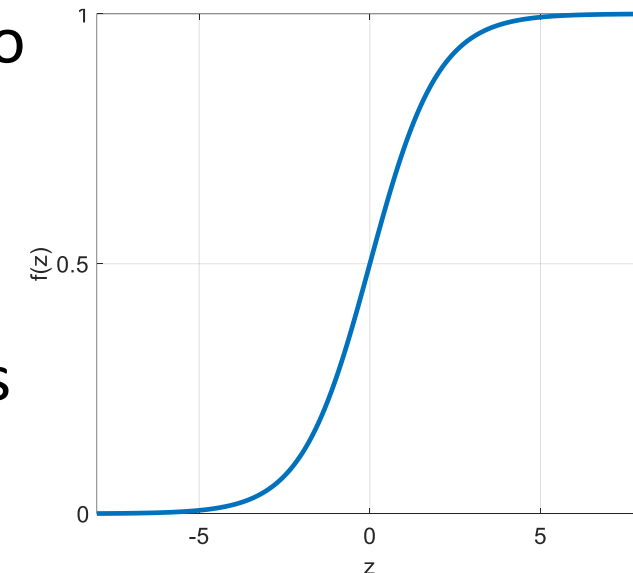
$$\text{Logistic}(z) = \frac{1}{1+e^{-z}} \in [0, 1],$$

apresenta tais propriedades matemáticas.

- Utilizando a **função logística** como **função de limiar**, temos

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

- $g(\mathbf{x})$ pode ser um **hiperplano**, um **polinômio**, etc.
- A saída será um número real entre 0 e 1, o qual pode ser interpretado como uma **probabilidade** de um dado exemplo pertencer à classe C_2 (ou seja, à **classe positiva**).
- A nova **função hipótese**, $h_a(\mathbf{x})$, forma uma **fronteira de decisão suave**, a qual confere a probabilidade de 0.5 para exemplos em cima da **fronteira de decisão** e se aproxima de 0 ou 1 conforme a posição do exemplo se distancia da fronteira.



A função logística realiza um mapeamento $\mathbb{R} \rightarrow [0,1]$.

Quanto mais longe da **fronteira de decisão**, mais próximo o valor de saída da **função hipótese** será de 0 ou de 1 e, portanto, mais certa teremos sobre uma classificação.

Em resumo, quanto mais longe, maior o valor de $g(\mathbf{x})$.

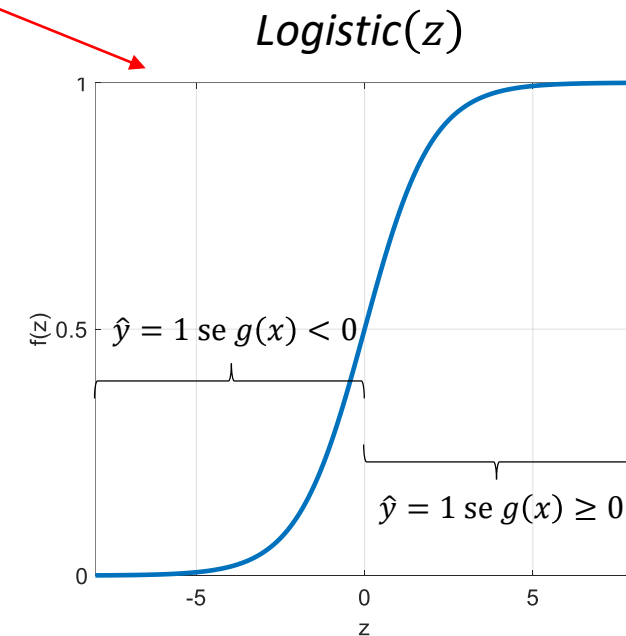
Regressão logística

- Esse classificador com função de **limiar logístico** é conhecido como **regressor logístico**.
- O **regressor logístico** estima a **probabilidade** de um exemplo pertencer a uma classe específica.
 - Por exemplo, qual é a probabilidade de uma dado email ser um spam?
- O **regressor logístico** é um algoritmo usado para **classificação binária**, mas precisamos quantizar sua saída.
- Ele é ótimo para situações em que precisamos classificar entre duas classes, negativa (C_1) e positiva (C_2).
- Normalmente, se quantiza a saída da **função hipótese**, $h_a(\mathbf{x})$, em dois valores, 0 ou 1.
- Se a **probabilidade** estimada para um exemplo for igual ou maior que 50%, o classificador **prediz** que o exemplo pertence à **classe positiva**, rotulada como 1, ou então **prediz** que não pertence, ou seja, pertence à **classe negativa**, rotulada como 0.
- 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

- Note que $\text{Logistic}(z) < 0.5$ quando $z < 0$ e $\text{Logistic}(z) \geq 0.5$ quando $z \geq 0$, portanto, o modelo de **regressão logística** prediz a classe positiva, C_2 , (i.e., $\hat{y} = 1$) se $g(x) \geq 0$ e C_1 (i.e., $\hat{y} = 0$) se $g(x) < 0$.
- A **regressão logística** funciona usando uma **combinação** (linear ou não linear) dos **atributos**, para que várias fontes de informação (i.e., atributos) possam ditar a saída do modelo.
- Os **parâmetros do modelo** são os **pesos** associados aos vários **atributos** e representam a importância relativa de cada **atributo** para o resultado.
- 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, saúde pública entre outras.
- Além disto, toda a teoria por trás da **regressão logística** foi a base para a criação das primeiras **redes neurais**.



Exemplos: classificar críticas de filmes como positivas ou negativas, probabilidade de um paciente desenvolver uma doença, detecção de spam, classificar transações como fraudulentas ou não, etc.

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** do vetor de atributos \mathbf{x} pertencer à classe positiva, C_2 , para qual a saída quantizada desejada é $y = 1$.
- Ou seja, $h_a(\mathbf{x})$ dá a probabilidade condicional da **classe positiva**, C_2 , i.e., $h_a(\mathbf{x}) = P(C_2 \mid \mathbf{x}; \mathbf{a})$.
- Assim, consequentemente, $(1 - h_a(\mathbf{x})) = P(C_1 \mid \mathbf{x}; \mathbf{a})$ é a probabilidade condicional da **classe negativa**, C_1 .
- A **fronteira de decisão** é determinada quando há uma **indecisão** entre as classes, ou seja, quando $P(C_1 \mid \mathbf{x}; \mathbf{a}) = P(C_2 \mid \mathbf{x}; \mathbf{a})$, que ocorre quando $P(C_2 \mid \mathbf{x}; \mathbf{a}) = h_a(\mathbf{x}) = 0.5$.
- Observando a figura da **função logística**, nós percebemos que $\text{Logistic}(z) = 0.5$ quando $z = 0$.
- Desta forma, a **fronteira de decisão** é caracterizada por

$$g(\mathbf{x}) = 0,$$

onde $g(\mathbf{x})$ pode ser uma reta, um plano, um círculo, uma hipérbole, etc.

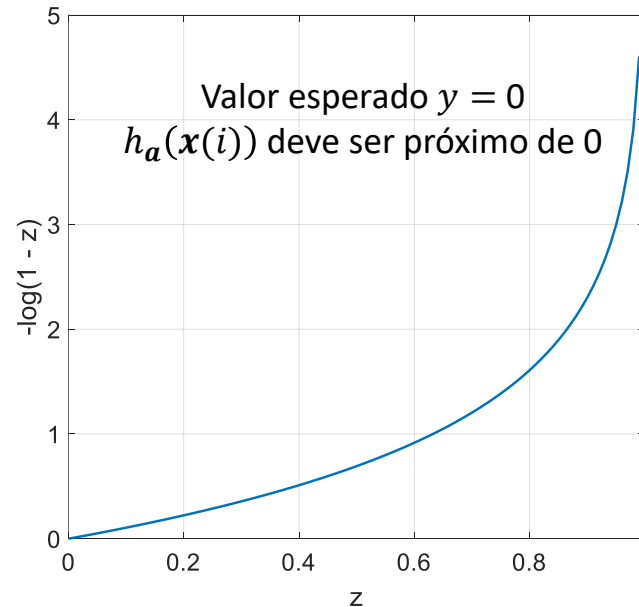
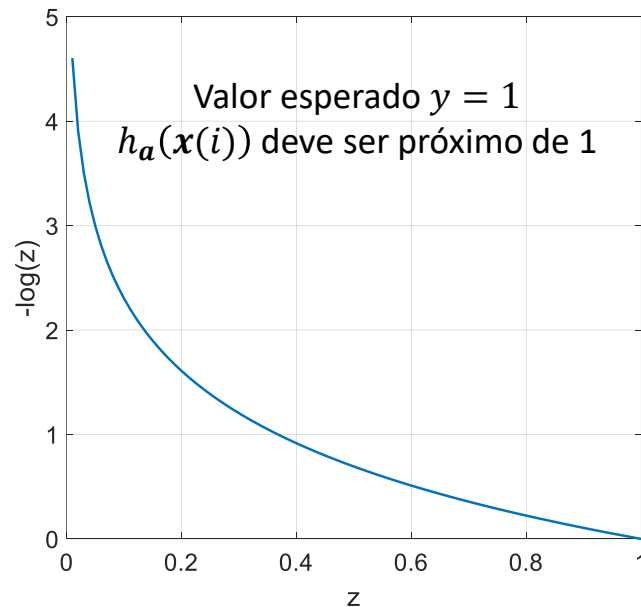
Função de erro

- Para treinarmos um **regressor logístico** e encontrar os **pesos** da **função hipótese**, nós precisamos, assim como fizemos com a **regressão linear**, definir uma **função de erro**.
- Porém, adotar o **erro quadrático médio** como **função de erro** não é uma boa escolha para a **adaptação dos pesos** no caso da **regressão logística** como veremos a seguir.
- A **função de erro** utilizando o **erro quadrático médio** é dada por

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

- Como $\text{Logistic}(\cdot)$ 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 (e.g., o algoritmo pode ficar preso em um mínimo local).
- **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
$$\text{Erro}(h_{\mathbf{a}}(\mathbf{x}(i)); y(i)) = \begin{cases} -\log(h_{\mathbf{a}}(\mathbf{x}(i))), & \text{se } y(i) = 1 \\ -\log(1 - h_{\mathbf{a}}(\mathbf{x}(i))), & \text{se } y(i) = 0 \end{cases}$$
- Veremos a seguir o motivo desta escolha.

Função de erro



- As figuras ao lado mostram as duas situações possíveis para a **função de erro**.
- Como podemos observar, a penalização aplicada a cada saída reflete o **erro de classificação**.

- O uso dessa **função de erro** faz sentido pois:
 - O valor de $-\log(z)$ se torna muito grande quando z se aproxima de 0, então o erro será grande se o classificador estimar uma probabilidade próxima a 0 para um exemplo positivo (i.e., pertencente à classe C_2)
 - O valor de $-\log(1-z)$ será muito grande se o classificador estimar uma probabilidade próxima de 1 para um exemplo negativo (i.e., pertencente à classe C_1).
 - Por outro lado, $-\log(z)$ se torna próximo de 0 quando z se aproxima de 1, portanto, o erro será próximo de 0 se a probabilidade estimada for próxima de 1 para um exemplo positivo.
 - O valor $-\log(1-z)$ se torna próximo de 0 quando z se aproxima de 0, portanto, o erro será próximo de 0 para um exemplo negativo.

Função de erro

- Nós podemos reduzir a definição da **função de erro** para **cada exemplo** a uma expressão única, dada por

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

- Com isto, podemos definir a seguinte **função de erro médio**:

$$\begin{aligned} J_e(\mathbf{a}) &= -\frac{1}{N} \sum_{i=0}^{N-1} y(i) \log(h_a(\mathbf{x}(i))) + (1 - y(i)) \log(1 - h_a(\mathbf{x}(i))) \\ &= -\frac{1}{N} \sum_{i=0}^{N-1} y(i) \log(P(C_2 | \mathbf{x}(i); \mathbf{a})) + (1 - y(i)) \log(P(C_1 | \mathbf{x}(i); \mathbf{a})). \end{aligned}$$

- A má notícia 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**).
- A boa notícia é que essa **função de erro** é **convexa** e portanto, é garantido que o algoritmo do **gradiente descendente** encontre o mínimo global (dado que a **taxa de aprendizagem** não seja muito grande e você espere tempo suficiente).

Processo de treinamento

[Exemplo: logistic regression with gradient descent.ipynb](#)

- Semelhante ao que 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

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

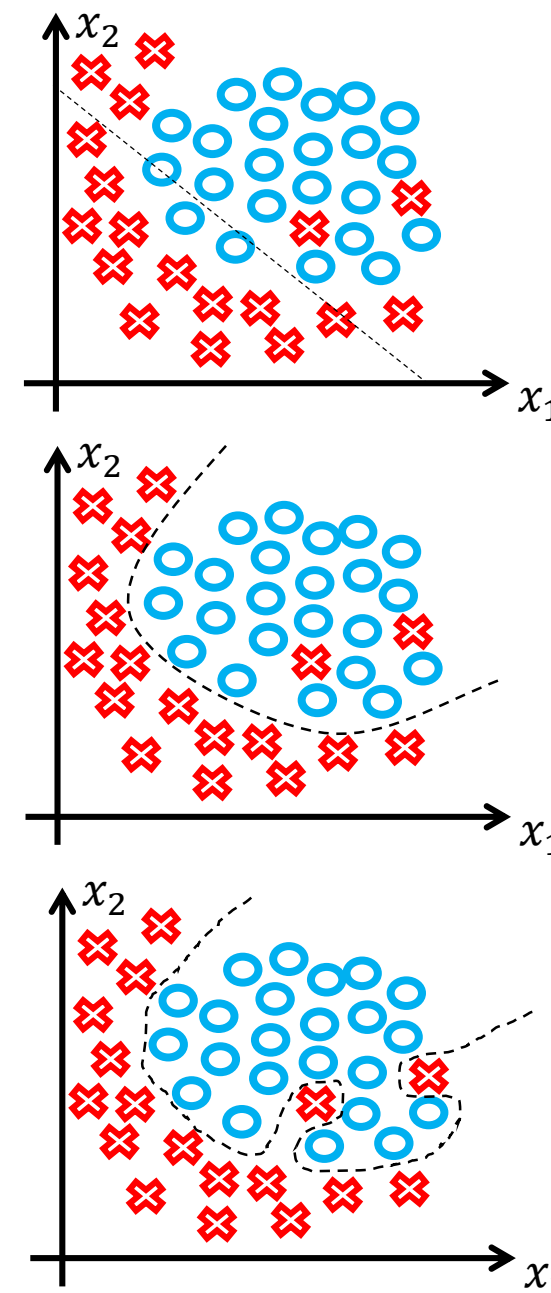
Aqui consideramos $g(\mathbf{x})$ como sendo a equação de um **hiperplano**: $g(\mathbf{x}) = \sum_{k=0}^K a_k x_k$, mas o resultado é facilmente estendido para polinômios.

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

- Percebam que o **vetor gradiente** da **função de erro médio** para a **regressão logística** é similar àquele obtido para a **regressão linear** com a função de **erro quadrático médio**.
- Agora, de posse do **vetor gradiente**, podemos usá-lo no algoritmo do **gradiente descendente** (nas versões em batelada, estocástico ou mini-batch).

Observações

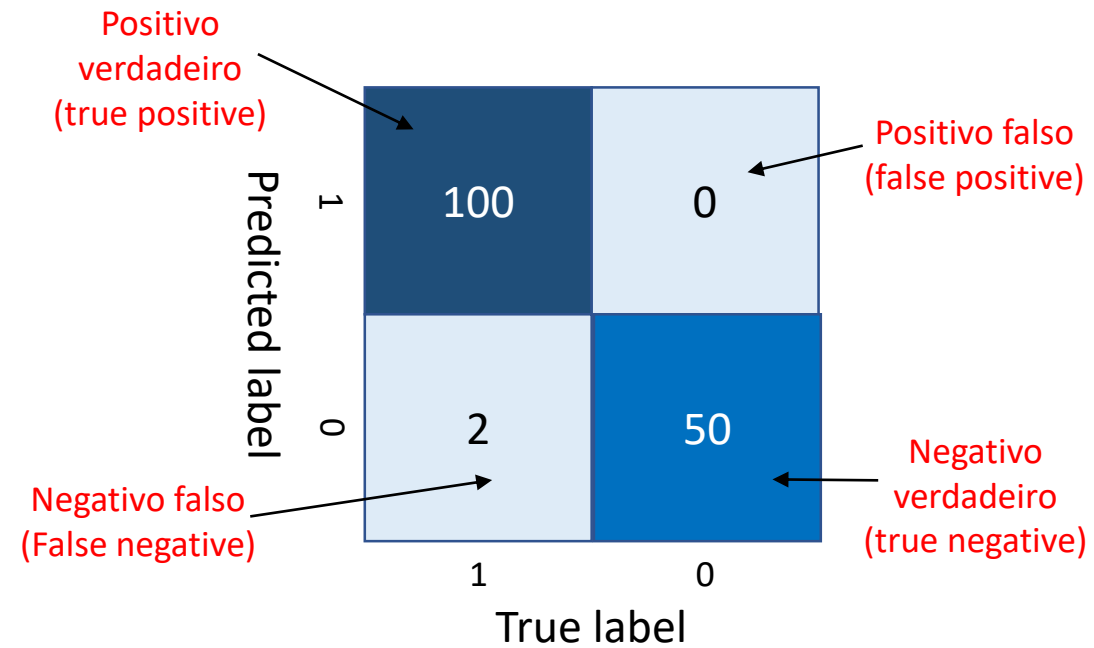
- Como vimos, a **função discriminante**, $g(x)$, pode também assumir a forma de um **polinômio** e, muitas vezes, nós não sabemos qual a melhor ordem para este polinômio.
- Assim, como nós discutimos no caso da **regressão linear**, modelos de **regressão logística** também estão sujeitos à ocorrência de **sobreajuste** e **subajuste**.
 - Na primeira figura, a **falta de flexibilidade** da reta usada faz com que o erro de classificação seja alto.
 - Na última figura, a **flexibilidade excessiva** do modelo (explorando um polinômio de 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 modelo ficou mais susceptível a erros de classificação para novos dados, ou seja, não irá generalizar bem.
 - Já a figura do meio mostra o que seria uma boa **hipótese de classificação**.
- Por isso, **técnicas de regularização** (e.g., LASSO, Ridge, Elastic-Net, Early-stop) também podem ser empregadas em seu treinamento, assim como **validação cruzada**.



Tarefas

- **Quiz:** “*T320 - Quiz - Classificação (Parte III)*” 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](#).
 - **Atividades podem ser feitas em grupo, mas as entregas devem ser individuais.**

Obrigado!



Recapitulando

- Aprendemos outra função de limiar, chamada de ***função logística***, com a qual é possível
 - encontrar uma solução eficiente para o problema da ***classificação binária*** com o ***gradiente descendente***.
 - termos o ***grau de confiança*** de uma classificação, ou seja, a ***probabilidade*** de um exemplo de entrada pertencer a uma das duas classes (Positiva ou Negativa).
- Vimos também que a ***função discriminate***, $g(x)$, pode ser a equação de um ***polinômio***, incluindo a equação do ***hiperplano***.
- O ***vetor gradiente*** da ***função de erro médio*** vai variar dependendo da ***função discriminate*** adotada.

Resumindo

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

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

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

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

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

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

Recapitulando

- 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

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

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.

- De posse do **vetor gradiente**, podemos usá-lo com o **gradiente descendente** para atualizar os pesos.

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