

Capacitação em Inteligência Artificial e Aplicações

Introdução às Redes Neurais: Aprendizado do Neurônio

- Prof. Gerson Vieira Albuquerque Neto
- Prof. Rodrigo Carvalho Souza Costa
- Prof. Yves Augusto Romero



**IA**

Objetivos da Aula

- Compreender o treinamento e otimização de redes neurais;



Aula 12 - Aprendizado (Gradiente e Backpropagation)





Introdução às Redes Neurais: Aprendizagem RNA

- Revisão Gradiente Descendente
- Calculando o Gradiente Descendente
- Backpropagation





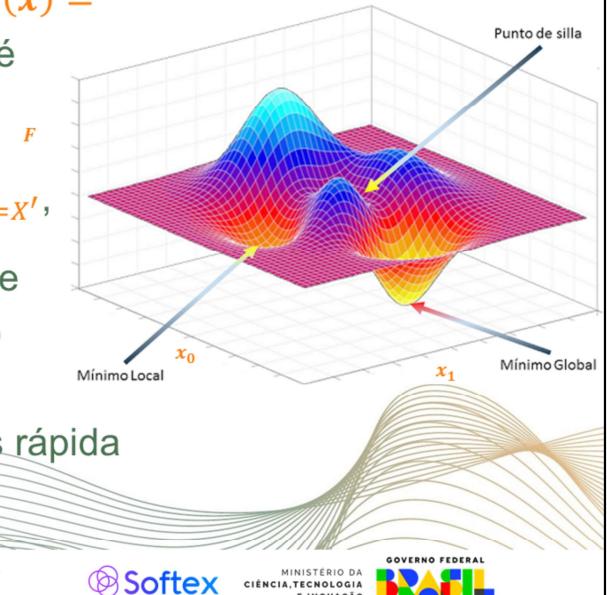
IA

Gradiente Descente e Função de Perda

- O gradiente da função multivariada $\mathbf{o} = \mathbf{f}(\mathbf{x}) = \mathbf{f}(x_0, x_1, \dots, x_n)$ em $\mathbf{X}' = [x_0', x_1', \dots, x_n']^T$ é mostrado da seguinte forma:

$$\nabla f(x_0', x_1', \dots, x_n') = [\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}]^T |_{\mathbf{X}=\mathbf{X}'},$$

A direção do vetor gradiente é a direção de crescimento mais rápido da função. Como resultado, a direção do vetor de gradiente negativo $-\nabla f$ é a direção de descida mais rápida da função.



**IA**

Gradiente Descente e Função de Perda

- Durante o treinamento da rede de aprendizado profundo, os erros de classificação de destino devem ser parametrizados.
- **Uma função de perda (função de erro)** é usado, o que reflete o erro entre a saída de destino e a saída real do perceptron.
- Para um único exemplo de treinamento x, a função de erro mais comum é a **Função de custo quadrático**.

$$E(w) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2,$$

- Na função anterior, d é um neurônio na camada de saída, D é todos os neurônios na camada de saída, t_d é a saída de destino (real), e o_d é a saída estimada (propagada).

- Esta função de perda tem as seguintes características:
 - Usa o vetor de peso como a variável independente.
 - Usa a soma dos quadrados da diferença entre a saída alvo t_d e a saída real o_d de cada amostra de treinamento como o corpo principal.
 - Existe um coeficiente 1/2.
- Uma vez que a amostra de treinamento é fornecida, a entrada e a saída de destino são constantes. A saída real varia com W. A variável independente da função de erro é W.
- O uso do coeficiente 1/2 que é difícil de entender é o seguinte: Quando você toma uma derivada de E em relação à variável independente, o coeficiente 1/2 é multiplicado pelo índice 2 e o número 1 é obtido. O cálculo específico será descrito no texto a seguir.



IA

Gradiente Descente e Função de Perda

- O método de gradiente descendente permite que a função de perda pesquise ao longo da direção do gradiente negativo e atualize os parâmetros iterativamente, finalmente minimizando a função de perda

Ponto de partida

Iteração 3

Iteração 4

Convergência

Valor final

$$\Delta W = -\eta \nabla E$$

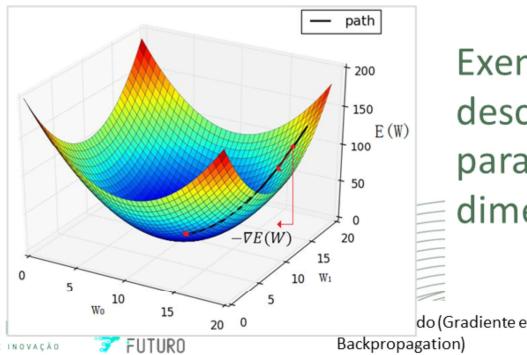


IA

Extremo da função de perda

- **Finalidade:** A função de perda $E(W)$ é definida no espaço de peso. O objetivo é buscar o vetor de peso W que possa minimizar $E(W)$.
- **Limitação:** Nenhum método eficaz pode resolver o extremo da função em matemática na complexa superfície de alta dimensão de

$$E(W) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2.$$



Exemplo de gradiente descendente de parabolóide de duas dimensões



IA

Funções de Custo

Função de Custo Quadrático	Função de Custo de Entropia Cruzada
$E(W) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$	$E(W) = - \sum_{d \in D} t_d \ln o_d$
representa a diferença entre a saída real e a saída estimada	representa a distância entre duas distribuições de probabilidade
usada para resolver o problema de regressão	amplamente utilizada para problemas de classificação



Introdução às Redes Neurais: Aprendizagem RNA

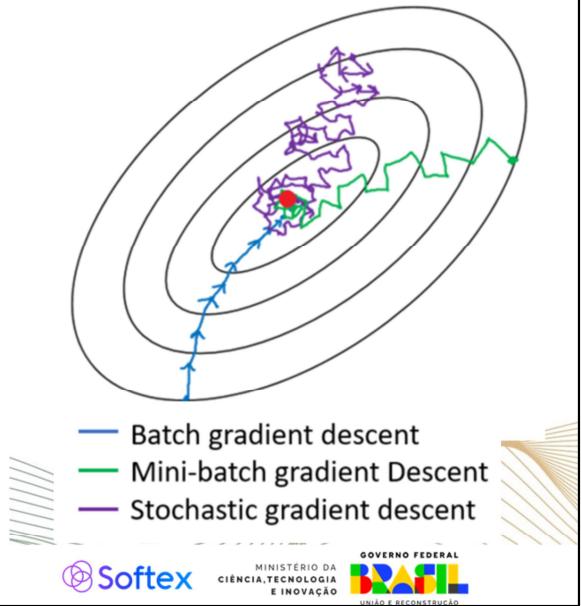
- Revisão sobre o gradiente descendente
- Calculando o Gradiente Descendente
- Backpropagation



**IA**

Formas de Cálculo do Gradiente Descendente

- **Algoritmo de descida de gradiente em lote (BGD)**
 - calcula-se o gradiente **usando todo o dataset de treinamento** em cada iteração, para atualizar os parâmetros.
- **Algoritmo de Descida de Gradiente Estocástico (SGD)**
 - Tomamos **um exemplo de treinamento para cálculo do gradiente** e usamos esse gradiente médio para atualizar nossos parâmetros.
- **Algoritmo de Descida de Gradiente de Mini-Lote (MBGD)**
 - O mini lote tenta encontrar um **equilíbrio entre BGD e SGD.**

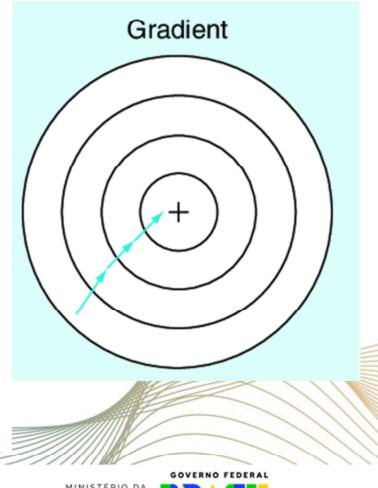




IA

Gradiente descendente em Lote (Batch) - BGD

- No conjunto de amostras de treinamento \mathbf{X} , cada amostra é registrada como $\langle \mathbf{x}, \mathbf{t} \rangle$, em que \mathbf{x} é o vetor de entrada, \mathbf{t} a saída alvo (a saída real) e η a taxa de aprendizado.
- Algoritmo:
 - Inicializa cada \mathbf{w}_i com um valor aleatório pequeno absoluto.
 - Antes que a condição de convergência seja atendida:
 - Inicializa cada $\Delta\mathbf{w}_i$ a zero.
 - Para cada amostra:
 - Calcule a saída \mathbf{o} para todas as amostras \mathbf{x} .
 - Para cada \mathbf{w}_i calcule $\Delta\mathbf{w}_i = \eta \frac{1}{n} \sum_{x \in X} \frac{\partial E(t_x, o_x)}{\partial w_i}$
 - Para atualize cada peso $\mathbf{w}_i = \mathbf{w}_i + \Delta\mathbf{w}_i$.



**IA**

Gradiente descendente em Lote (Batch) - BGD

- Prós:
 - Ideia simples e cada iteração é barata;
 - Garantia de convergência para o mínimo local;
 - Com vários algoritmos de segunda ordem para acelerar sua convergência;
 - Muito rápido para matrizes bem condicionadas e problemas fortemente convexos;
- Contras:
 - Frequentemente é lento, pois problemas interessantes não são fortemente convexos ou bem condicionados;
 - Não lida com funções não diferenciáveis
 - Utiliza todos os dados de treinamento para estimar os parâmetros.
 - Assim, para grandes bancos de dados, torna-se lento.

- se o número de exemplos de treinamento for grande, então o gradiente descendente em lote é computacionalmente muito caro!
- Imagine se você tem 10000 dados, cada dado com 10 características, são 100 mil valores para computar a cada iteração, em cada época.
- Não é comumente usado porque o processo de convergência é muito lento, pois todas as amostras de treinamento precisam ser calculadas toda vez que o peso é atualizado.



IA

Gradiente descendente Estocástico - SGD

- Consiste em uma variante comum, também chamada algoritmo Incremental Gradient Descent. Essa implementação é chamada de Aprendizagem Online, que atualiza o gradiente com base em uma única amostra:

$$\Delta w_i = -\eta \frac{1}{n} \sum_{x \in X} \frac{\partial E(t_x, o_x)}{\partial w_i} \Rightarrow \Delta w_i = -\eta \frac{\partial E(t_k, o_k)}{\partial w_i}$$

- Algoritmo de Aprendizado Online

- Inicializa cada w_i com um valor pequeno valor absoluto
- Antes que a condição de convergência seja atendida:
 - Seleciona aleatoriamente uma amostra k
 - Calcule a saída o para a amostra x_k .
 - Para atualize cada peso $w_i = w_i + \frac{\partial E(t_k, o_k)}{\partial w_i}$.

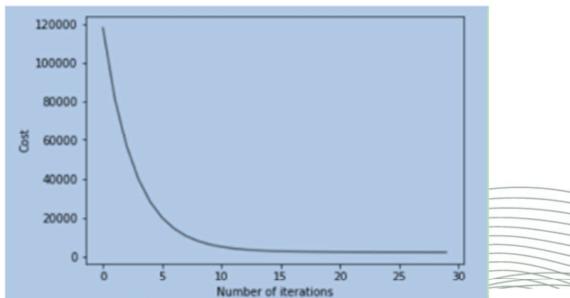
- Gradiente descendente estocástico (Stochastic Gradient Descent, SGD): calcula-se o gradiente usando $b = 1$ dados aleatórios de treinamento por iteração, para atualizar os parâmetros. O SGD converge mais rapidamente para conjuntos de dados maiores. Porém, como no SGD usamos apenas um dado de cada vez, não podemos usar implementação vetorizada. Isso pode desacelerar os cálculos.

**IA**

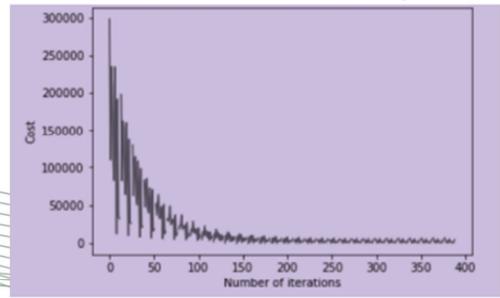
Gradiente Descendente Estocástico (SGD)

- Prós:
 - Convergência mais rápida, especialmente com grandes bancos de dados ou dados redundantes,
 - A trajetória estocástica permite escapar de um mínimo local;
- Contras:
 - Prova da convergência é probabilística;
 - Muitos métodos de segunda ordem não funcionam

Batch Gradient Descent, BGD



Stochastic Gradient Descent, SGD



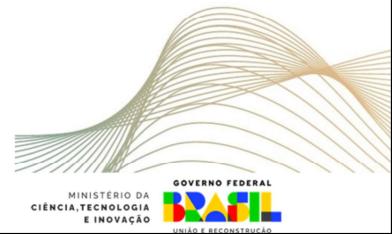
- Esse algoritmo de GD vai para outro extremo, ou seja, atualizando o peso com base em uma amostra.
- A maioria das amostras de treinamento contém ruídos. Como resultado, quando o extremo é aproximado, a direção do gradiente é orientada para cima e para baixo perto do extremo, mas difícil de convergir para o extremo.
- Exemplo
 - Imagine que temos dados de treino de tamanho 100.000; Mas na verdade são 100 cópias de 1000;- Ou seja, padrões parecidos, com mesmo efeito;- Batch será, pelo menos, 100 vezes mais devagar.



IA

Mini-batch Gradient Descent Algorithm (MBGD)

- Resolve os defeitos dos dois algoritmos **BGD** e **SGD**
- Foi proposto e tem sido mais amplamente utilizado. Um pequeno número de amostras de tamanho de lote (**BS**) são usados de cada vez para calcular Δw_i e, em seguida, o peso é atualizado
- Algoritmo:
 - Inicializa cada w_i com um valor aleatório pequeno absoluto.
 - Antes que a condição de convergência seja atendida:
 - Inicializa cada Δw_i a zero.
 - Para cada amostra dentro de um lote **B**:
 - Calcule a saída o para todas as amostras x .
 - Para cada w_i calcule $\Delta w_i = \eta \frac{1}{n} \sum_{x \in B} \frac{\partial E(t_x, o_x)}{\partial w_i}$
 - Para atualize cada peso $w_i = w_i + \Delta w_i$.
 - Para o próximo lote, embaralhe as amostras aleatoriamente

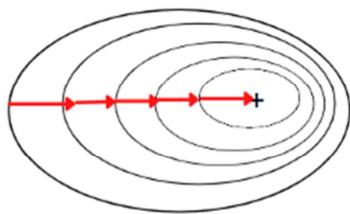


- Gradiente descendente em lote (BGD) a média dos gradientes de todos os exemplos de treinamento e usamos esse gradiente médio para atualizar nossos parâmetros.
- Gradiente descendente estocástico (SGD): um exemplo de treinamento para cálculo do gradiente e usamos esse gradiente médio para atualizar nossos parâmetros.
- Gradiente descendente em mini lotes (MBGD) O mini lote tenta encontrar um equilíbrio entre BGD e SGD.

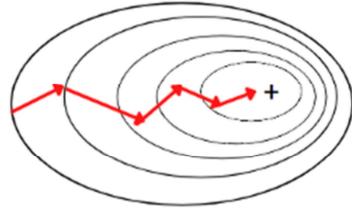
**IA**

Comparação

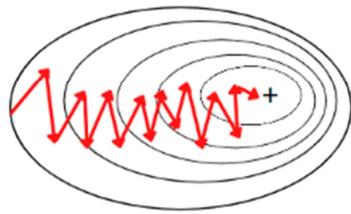
Batch Gradient Descent

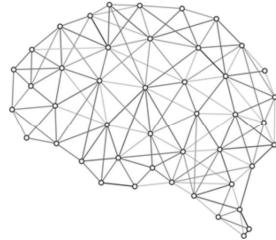
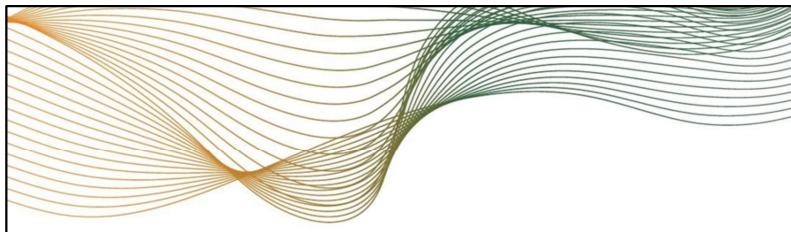


Mini-Batch Gradient Descent



Stochastic Gradient Descent





Introdução às Redes Neurais: Aprendizagem RNA

- Revisão Gradiente Descendente
- Calculando o Gradiente Descendente
- Back propagation



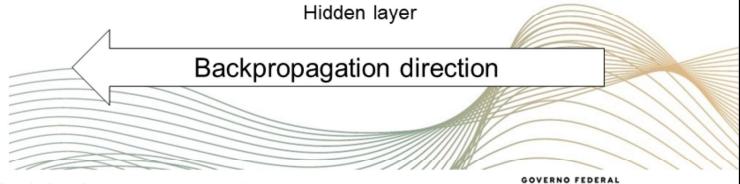
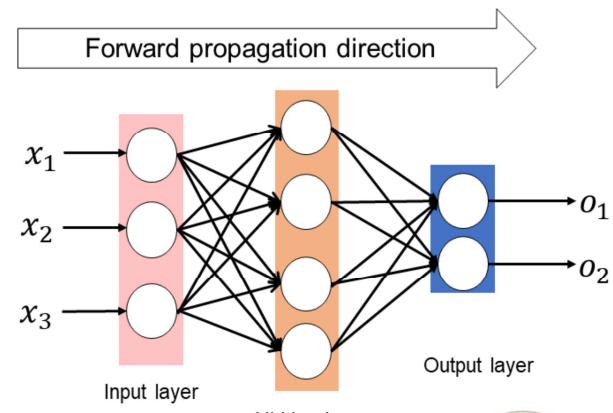


IA

Algoritmo de Backpropagation

- Os sinais são propagados na direção para a frente e os erros são propagados na direção contrária.
- No conjunto de amostras de treinamento D, cada amostra é registrada como $\langle X, t \rangle$, em que X é o vetor de entrada, t a saída alvo, o a saída real e w o coeficiente de peso.
- Função de perda:

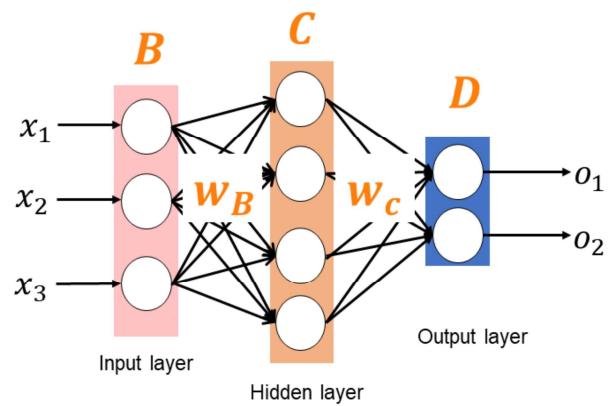
$$E(w) = \frac{1}{2} \sum_{(d \in D)} (t_d - o_d)^2$$



**IA**

Algoritmo de Backpropagation

- De acordo com as fórmulas a seguir, erros nas camadas de entrada, oculta e saída são acumulados para gerar o erro na função de perda.
- Em que:
 - w_c é o coeficiente de peso entre a camada oculta e a camada de saída,
 - w_b é o coeficiente de peso entre a camada de entrada e a camada oculta.
 - f é a função de ativação, D é o conjunto de camadas de saída, e C e B são o conjunto de camadas ocultas e o conjunto de camadas de entrada, respectivamente.

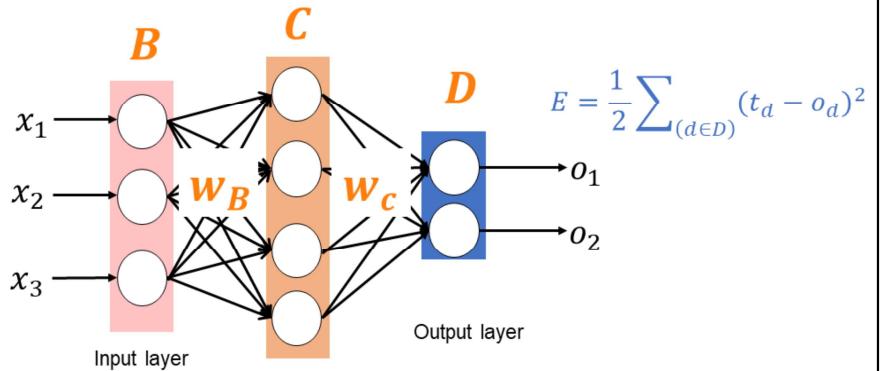




IA

Algoritmo de Backpropagation

- Supondo que a função de perda é uma função de custo quadrática:



$$E = \frac{1}{2} \sum_{(d \in D)} \left[t_d - f \left(\sum_{(c \in C)} w_c f(\text{net}_c) \right) \right]^2 =$$
$$\frac{1}{2} \sum_{(d \in D)} \left[t_d - f \left(\sum_{(c \in C)} w_c f \left(\sum_{b \in B} w_b x_b \right) \right) \right]^2$$
$$E = \frac{1}{2} \sum_{(d \in D)} [t_d - f(\text{net}_d)]^2$$
$$E = \frac{1}{2} \sum_{(d \in D)} [t_d - f(\sum_{(c \in C)} w_c y_c)]^2$$



**IA**

Algoritmo de Backpropagation

- Para minimizar o erro E, o cálculo iterativo de descida de gradiente pode ser usado para resolver W_c e W_b , ou seja, calcular W_c e W_b para minimizar o erro E:

$$\Delta w_c = -\eta \frac{\partial E}{\partial w_c}, c \in C$$

$$\Delta w_b = -\eta \frac{\partial E}{\partial w_b}, b \in B$$

- Se houver várias camadas ocultas, as regras de cadeia serão usadas para obter uma derivada para cada camada para obter os parâmetros otimizados por iteração.

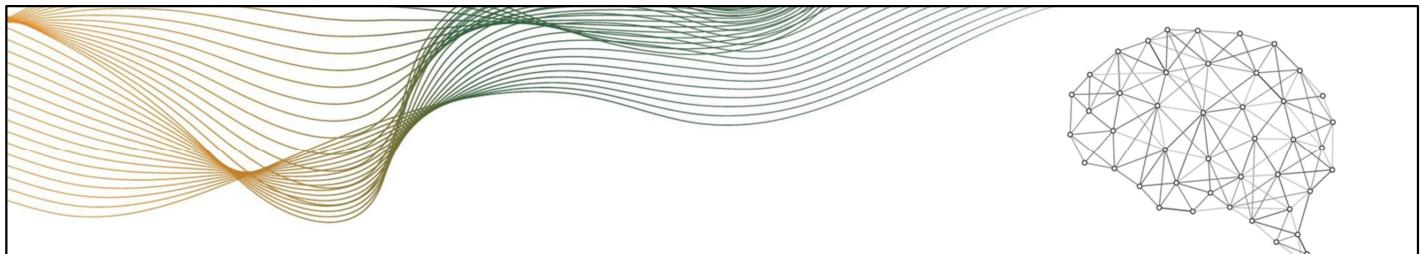
**IA**

Algoritmo de Backpropagation

- Para uma rede neural com qualquer número de camadas, a fórmula organizada para o treinamento é a seguinte:

$$\Delta w_{jk}^l = -\eta \delta_k^{l+1} f_j(z_j^l)$$
$$\delta_j^l = \begin{cases} f_j'(z_j^l)(t_j - f_j(z_j^l)), & l \in \text{saída}, (1) \\ \sum_k \delta_k^{l+1} w_{jk}^l f_j'(z_j^l), & \text{outras camadas}, (2) \end{cases}$$

- O algoritmo BP é usado para treinar a rede da seguinte maneira:
 - Tira a próxima amostra de treinamento $\langle X, T \rangle$, propaga X na rede e obtém a saída real o .
 - Calcula a camada de saída δ de acordo com a fórmula de erro da camada de saída (1).
 - Calcula δ de cada camada oculta da saída para a entrada por iteração de acordo com a fórmula de propagação de erro da camada oculta (2).
 - De acordo com a δ de cada camada, os valores de peso de toda a camada são atualizados.



Introdução às Redes Neurais: Perceptron Simples

- Prática de redes neurais



Dúvidas?

Módulo de Inteligência Artificial



MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

