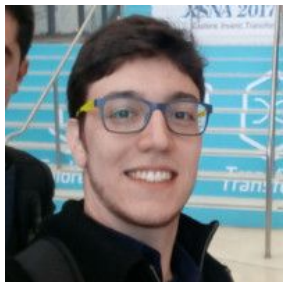


# Redes Neurais Profundas para aplicações de Visão Computacional (RNP-VC)

**Lucas Pereira, Rafael Teixeira, Lucas Assis, Anderson Soares**  
Instituto de Informática  
Universidade Federal de Goiás (UFG)

# Instrutores



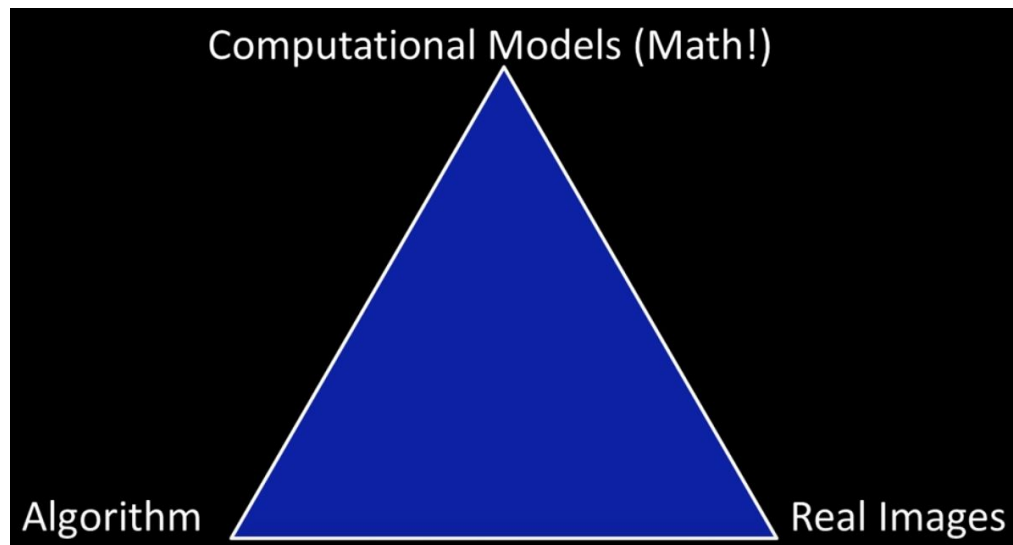
# Sumário

- O que o curso é e o que não é
- Cronograma e avaliação
- Revisão de tensores

# O que o curso não é:

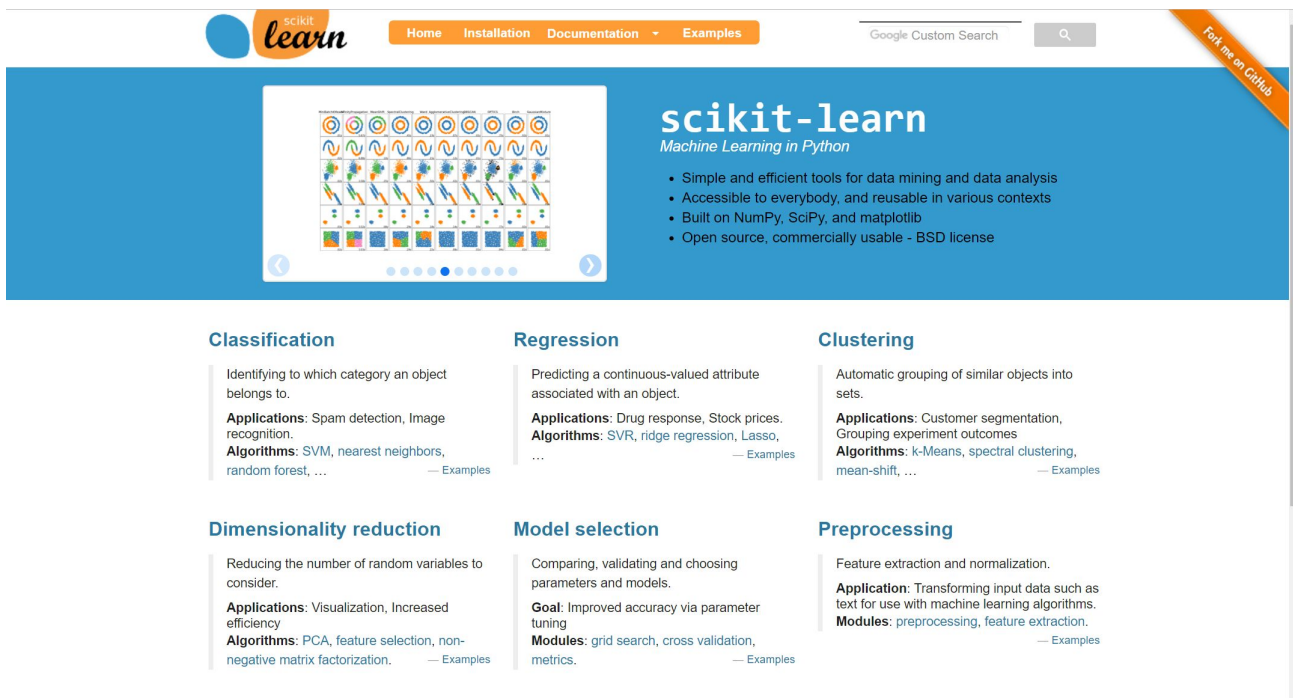
- Um curso de visão computacional

**IMAGE PROCESSING**  
**CAMERA MODELS AND VIEWS**  
**FEATURES AND MATCHING**  
**LIGHTNESS AND BRIGHTNESS**  
**IMAGE MOTION**  
**MOTION AND TRACKING**  
**CLASSIFICATION AND RECOGNITION**  
**MISCELLANEOUS OPERATIONS**  
**HUMAN VISION**



# O que o curso não é:

- Um curso de Machine Learning



The screenshot shows the scikit-learn website homepage. At the top, there is a navigation bar with links for Home, Installation, Documentation, and Examples. A Google Custom Search bar is also present. The main header features the scikit-learn logo and the text "Machine Learning in Python". Below this, a grid of 25 small icons represents various machine learning concepts. To the right of the grid, the text "scikit-learn" is displayed in a large font, followed by "Machine Learning in Python" in a smaller font. Below this, a list of bullet points describes the library's features: "Simple and efficient tools for data mining and data analysis", "Accessible to everybody, and reusable in various contexts", "Built on NumPy, SciPy, and matplotlib", and "Open source, commercially usable - BSD license". On the right side of the header, there is a yellow banner that says "Fork me on GitHub". The main content area is divided into six sections, each with a title, a brief description, applications, algorithms, and examples. The sections are: Classification, Regression, Clustering, Dimensionality reduction, Model selection, and Preprocessing.

**scikit-learn**  
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

**Classification**

Identifying to which category an object belongs to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ... — Examples

**Regression**

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, ridge regression, Lasso, ... — Examples

**Clustering**

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, ... — Examples

**Dimensionality reduction**

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, feature selection, non-negative matrix factorization. — Examples

**Model selection**

Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning

**Modules:** grid search, cross validation, metrics. — Examples

**Preprocessing**

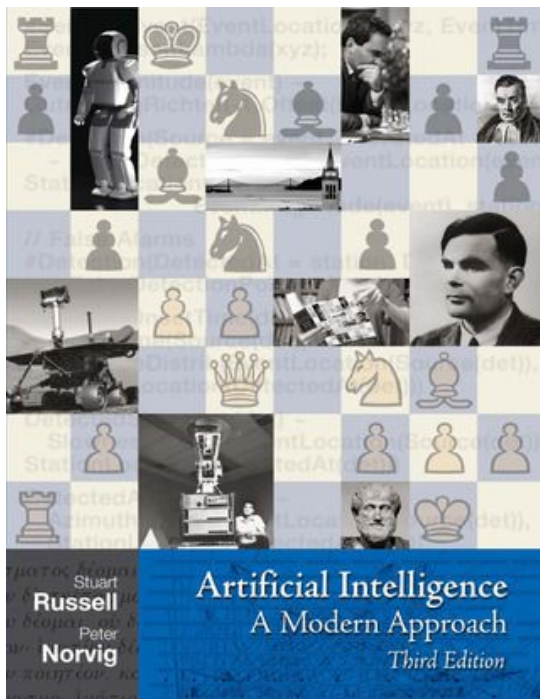
Feature extraction and normalization.

**Application:** Transforming input data such as text for use with machine learning algorithms.

**Modules:** preprocessing, feature extraction. — Examples

# O que o curso não é:

- Um curso de Inteligência Artificial



## Part I Artificial Intelligence

- 1 Introduction ... 1
- 2 Intelligent Agents ... 34

## Part II Problem Solving

- 3 Solving Problems by Searching ... 64
- 4 Beyond Classical Search ... 120
- 5 Adversarial Search ... 161
- 6 Constraint Satisfaction Problems ... 202

## Part III Knowledge and Reasoning

- 7 Logical Agents ... 234
- 8 First-Order Logic ... 285
- 9 Inference in First-Order Logic ... 322
- 10 Classical Planning ... 366
- 11 Planning and Acting in the Real World ... 401
- 12 Knowledge Representation ... 437

## Part IV Uncertain Knowledge and Reasoning

- 13 Quantifying Uncertainty ... 480
- 14 Probabilistic Reasoning ... 510
- 15 Probabilistic Reasoning over Time ... 566
- 16 Making Simple Decisions ... 610
- 17 Making Complex Decisions ... 645

## Part V Learning

- 18 Learning from Examples ... 693
- 19 Knowledge in Learning ... 768
- 20 Learning Probabilistic Models ... 802
- 21 Reinforcement Learning ... 830

## Part VII Communicating, Perceiving, and Acting

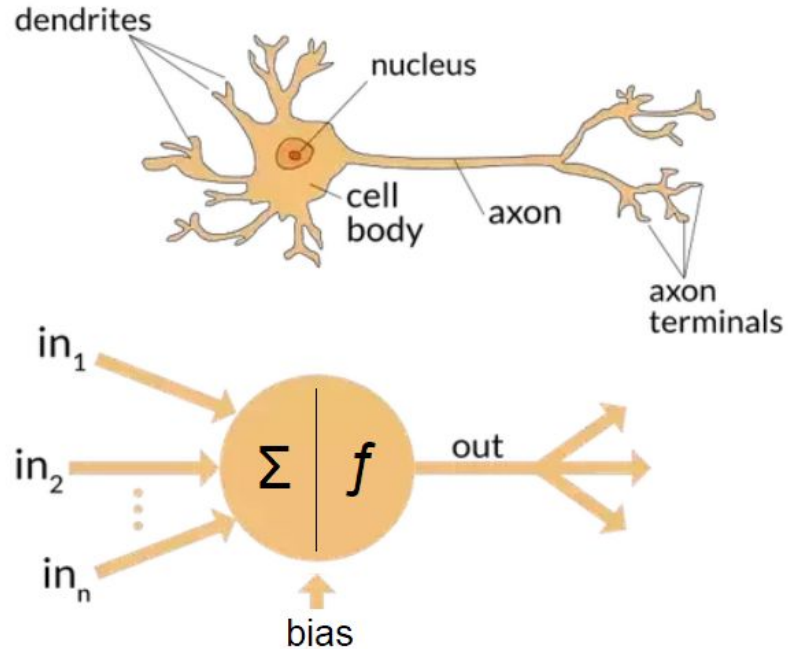
- 22 Natural Language Processing ... 860
- 23 Natural Language for Communication ... 888
- 24 Perception ... 928
- 25 Robotics ... 971

## Part VIII Conclusions

- 26 Philosophical Foundations ... 1020
- 27 AI: The Present and Future ... 1044
- A Mathematical Background [[pdf](#)] ... 1053
- B Notes on Languages and Algorithms [[pdf](#)] ... 1060
- Bibliography [[pdf](#) and [histograms](#)] ... 1063
- Index [[html](#) or [pdf](#)] ... 1109

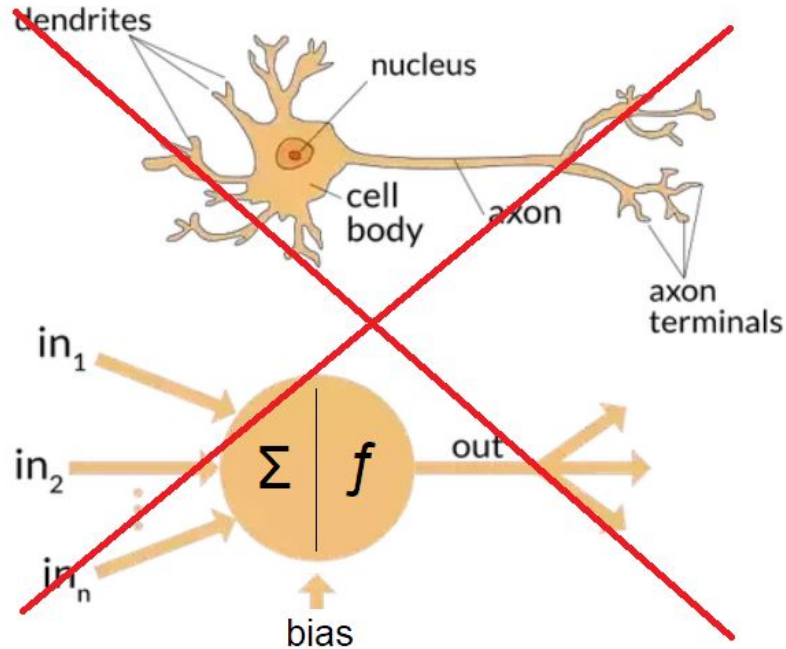
# O que o curso não é:

- Um curso de neurociência



O que o curso não é:

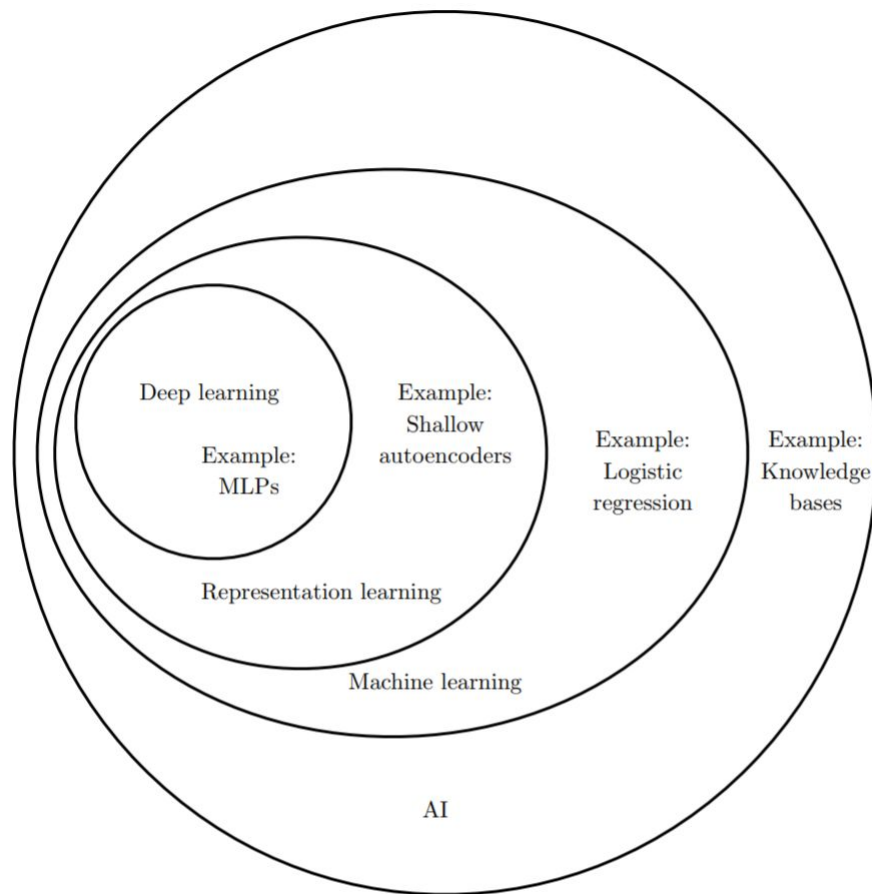
- Um curso de neurociência





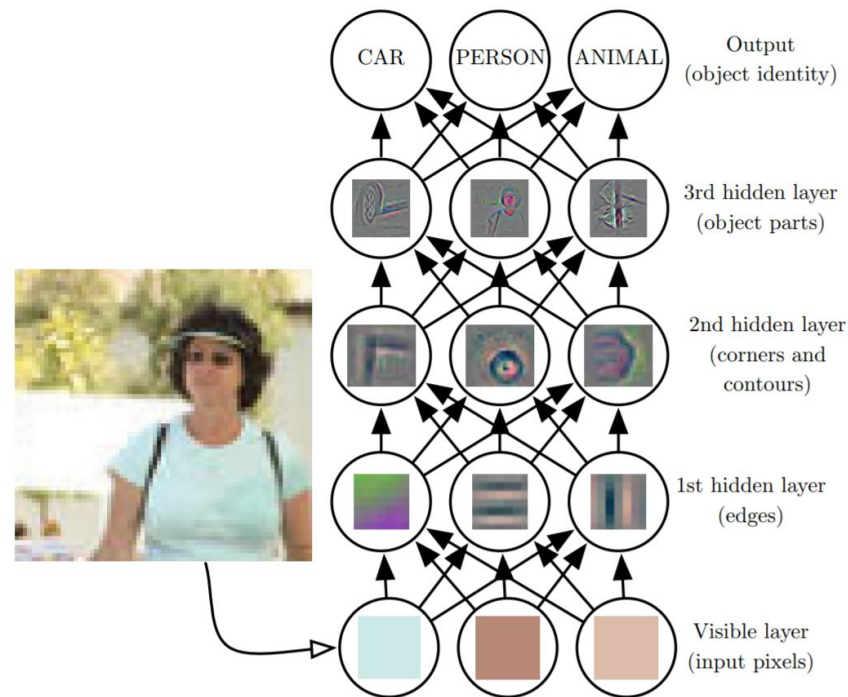
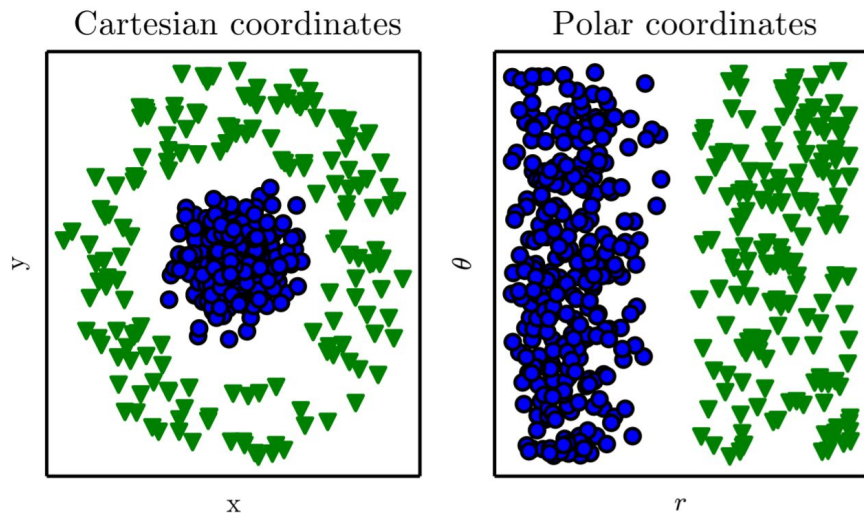
# O que o curso é:

- Introdução a Redes Neurais Profundas (RNP)
- Aplicação de RNP em tarefas de Visão Computacional



# O que o curso é:

- A importância do aprendizado de representações
- Introdução ao conceito de composição de representações



# Cronograma

- Aula 0-0: Visão geral sobre o curso
- Aula 0-1: Revisão de tensores
- Aula 1: Introdução a ML e VC
- Aula 2: Introdução a Redes Neurais
- Aula 3: Desenvolvimento de modelos - parte 1
- Aula prática
- Aula 4: Desenvolvimento de modelos - parte 2
- Aula 5: Transfer learning e zoológico de modelos
- Aula prática
- Aula 6: Detecção de objetos
- Aula 7: Segmentação semântica
- Aula 8: Aplicações com dados sequenciais
- Aula prática
- Aula 9: Modelos generativos
- Aula 10: Visualização e interpretabilidade

# Atividades avaliativas

- Participação das aulas práticas
- Prova teórica (não espere questões fáceis)
  - Provas de turmas anteriores duraram 24h... 26h...
- Projeto final para apresentação em workshop
  - Resolva o problema de alguém disposto a pagar pelo que você fez
  - Traremos pessoas de mercado para avaliar a solução

# Atividades avaliativas

- Entre para a história  
(se puder...)

Ranking histórico da disciplina de Deep Learning			
Posição	Nome	Turma	Nota
1	José Adenaldo	2017/2	9,5
2	Manoel Veríssimo	2018/1	9,2
3	Gabriel Horikawa	2018/1	9
4	Rafael Teixeira	2017/1	8,8
5	Leandro Leal	2017/2	8,7
6	Pedro Vitor	2017/1	8,7
7	Larissa Moraes	2017/1	8,6
8	Alexandre Barbosa	2016/2	8,6
9	Vinícius Araújo	2017/1	8,5
	Thiago Mendes	2018/1	8,5
	Fred Oliveira	2018/1	8,5

# Aula 0-1: Revisão de tensores:

## Algebra Linear

- Escalar x Vetor x Matriz

$$x \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} X_{0,0} & X_{0,1} & \cdots & X_{0,n-1} \\ X_{1,0} & X_{1,1} & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ X_{m-1,0} & \cdots & \cdots & X_{m-1,n-1} \end{bmatrix}$$

$$x \in \mathbb{R} \quad \mathbf{x} \in \mathbb{R}^n$$

$$\mathbf{X} \in \mathbb{R}^{m,n}$$

# Algebra Linear

- Tensor?

# Algebra Linear

- Tensor  $\mathbf{X} \in \mathbb{R}^{n_0, n_1, \dots, n_k}$

Quiz:

- Tensor de 0 dimensões?
- Tensor de 1 dimensão?
- Tensor de 2 dimensões?



# Algebra Linear

- Tensor  $\mathbf{X} \in \mathbb{R}^{n_0, n_1, \dots, n_k}$

Quiz:

- Tensor de 0 dimensões? *Escalar*
- Tensor de 1 dimensão? *Vetor*
- Tensor de 2 dimensões? *Matriz*

# Algebra Linear


## - Operações com matrizes

- Soma e subtração

$$\mathbf{C} = \mathbf{A} + \mathbf{B} \Leftrightarrow C_{i,j} = A_{i,j} + B_{i,j}$$

- Transposta

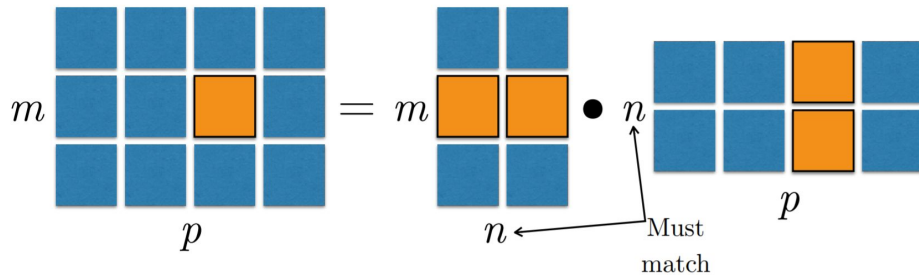
$$(\mathbf{A}^T)_{i,j} = A_{j,i}$$

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \mathbf{A}^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$


- Produto (matmul ou dot product)

$$\mathbf{C} = \mathbf{AB} \Leftrightarrow C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

$$\mathbf{AB} \neq \mathbf{BA}$$



# Algebra Linear

- Divisão?

# Algebra Linear

- Divisão = produto do inverso!

$$\mathbf{A}/\mathbf{B} = \mathbf{A}\mathbf{B}^{-1}$$

- Inverso\*

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n$$

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & 1 \end{bmatrix}, \mathbf{I}_n \in \mathbb{R}^{n,n}$$

\* nem toda matriz tem inverso!

# Algebra Linear

- Operações com tensores
  - Operações elemento por elemento (element-wise)

$$\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n_0, n_1, \dots, n_k} :$$

$$C_{i_0, i_1, \dots, i_k} = A_{i_0, i_1, \dots, i_k} + B_{i_0, i_1, \dots, i_k}$$

$$C_{i_0, i_1, \dots, i_k} = A_{i_0, i_1, \dots, i_k} B_{i_0, i_1, \dots, i_k}$$

- Operações com escalares (broadcast)

$$\mathbf{A}, \mathbf{C} \in \mathbb{R}^{n_0, n_1, \dots, n_k}, b \in \mathbb{R} :$$

$$C_{i_0, i_1, \dots, i_k} = A_{i_0, i_1, \dots, i_k} + b$$

$$C_{i_0, i_1, \dots, i_k} = b A_{i_0, i_1, \dots, i_k}$$

# Algebra Linear

- Operações com tensores
  - Transposta

`numpy.transpose(a, axes=None)`

[\[source\]](#)

Permute the dimensions of an array.

**Parameters:** *a : array\_like*

Input array.

*axes : list of ints, optional*

By default, reverse the dimensions, otherwise permute the axes according to the values given.

**Returns:** *p : ndarray*

*a* with its axes permuted. A view is returned whenever possible.

```
>>> x = np.ones((1, 2, 3))
>>> np.transpose(x, (1, 0, 2)).shape
(2, 1, 3)
```

# Algebra Linear

- Operações com tensores
  - Transposta

```
torch.transpose(input, dim0, dim1) → Tensor
```

Returns a tensor that is a transposed version of `input`. The given dimensions `dim0` and `dim1` are swapped.

The resulting `out` tensor shares it's underlying storage with the `input` tensor, so changing the content of one would change the content of the other.

## Parameters

- **input** (*Tensor*) – the input tensor
- **dim0** (*int*) – the first dimension to be transposed
- **dim1** (*int*) – the second dimension to be transposed

# Algebra Linear

- Operações com tensores
  - Transposta

```
tf.transpose(  
    a,  
    perm=None,  
    name='transpose',  
    conjugate=False  
)
```



Permutes the dimensions according to `perm`.

The returned tensor's dimension `i` will correspond to the input dimension `perm[i]`. If `perm` is not given, it is set to `(n-1...0)`, where `n` is the rank of the input tensor. Hence by default, this operation performs a regular matrix transpose on 2-D input Tensors. If `conjugate` is `True` and `a.dtype` is either `complex64` or `complex128` then the values of `a` are conjugated and transposed.



# Algebra Linear

- Multiplicação de tensores (matmul  $\neq$  dot product)

```
>>> a = np.ones([9, 5, 7, 4])
>>> c = np.ones([9, 5, 4, 3])
>>> np.dot(a, c).shape
(9, 5, 7, 9, 5, 3)
>>> np.matmul(a, c).shape
(9, 5, 7, 3)
>>> # n is 7, k is 4, m is 3
```

# Algebra Linear

- Multiplicação de tensores (matmul  $\neq$  dot product)

- `tf.matmul`

```
tf.linalg.matmul(  
    a,  
    b,  
    transpose_a=False,  
    transpose_b=False,  
    adjoint_a=False,  
    adjoint_b=False,  
    a_is_sparse=False,  
    b_is_sparse=False,  
    name=None  
)
```



The inputs must, following any transpositions, be tensors of rank  $\geq 2$  where the inner 2 dimensions specify valid matrix multiplication arguments, and any further outer dimensions match.

# Algebra Linear

- Multiplicação de tensores (matmul  $\neq$  dot product)

```
>>> # vector x vector
>>> tensor1 = torch.randn(3)
>>> tensor2 = torch.randn(3)
>>> torch.matmul(tensor1, tensor2).size()
torch.Size([1])
>>> # matrix x vector
>>> tensor1 = torch.randn(3, 4)
>>> tensor2 = torch.randn(4)
>>> torch.matmul(tensor1, tensor2).size()
torch.Size([3])
>>> # batched matrix x broadcasted vector
>>> tensor1 = torch.randn(10, 3, 4)
>>> tensor2 = torch.randn(4)
>>> torch.matmul(tensor1, tensor2).size()
torch.Size([10, 3])
>>> # batched matrix x batched matrix
>>> tensor1 = torch.randn(10, 3, 4)
>>> tensor2 = torch.randn(10, 4, 5)
>>> torch.matmul(tensor1, tensor2).size()
torch.Size([10, 3, 5])
>>> # batched matrix x broadcasted matrix
>>> tensor1 = torch.randn(10, 3, 4)
>>> tensor2 = torch.randn(4, 5)
>>> torch.matmul(tensor1, tensor2).size()
torch.Size([10, 3, 5])
```

# Algebra Linear

## - Broadcasting

1. they are equal, or
2. one of them is 1

```
>>> x=torch.empty(5,7,3)
>>> y=torch.empty(5,7,3)
# same shapes are always broadcastable (i.e. the above rules always hold)

>>> x=torch.empty((0,))
>>> y=torch.empty(2,2)
# x and y are not broadcastable, because x does not have at least 1 dimension

# can line up trailing dimensions
>>> x=torch.empty(5,3,4,1)
>>> y=torch.empty( 3,1,1)
# x and y are broadcastable.
# 1st trailing dimension: both have size 1
# 2nd trailing dimension: y has size 1
# 3rd trailing dimension: x size == y size
# 4th trailing dimension: y dimension doesn't exist

# but:
>>> x=torch.empty(5,2,4,1)
>>> y=torch.empty( 3,1,1)
# x and y are not broadcastable, because in the 3rd trailing dimension 2 != 3
```

# Algebra Linear

- Broadcasting

1. they are equal, or
2. one of them is 1

```
>>> x = np.arange(4)
>>> xx = x.reshape(4,1)
>>> y = np.ones(5)
>>> z = np.ones((3,4))
>>> y.shape
(5,)
>>> x.shape
(4,)
>>> x + y
ValueError: operands could not be broadcast together with shapes (4,) (5,)
>>> xx.shape
(4, 1)
>>> xx + y
array([[ 1.,  1.,  1.,  1.,  1.],
       [ 2.,  2.,  2.,  2.,  2.],
       [ 3.,  3.,  3.,  3.,  3.],
       [ 4.,  4.,  4.,  4.,  4.]])
```

# Algebra Linear

- Por que matrizes?

# Algebra Linear

- Por que matrizes?
  - Sistemas lineares

$$\begin{cases} ax + by = c \\ dx + ey = f \end{cases} \Leftrightarrow \mathbf{Ax} = \mathbf{b}, \mathbf{A} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \mathbf{b} = \begin{bmatrix} c \\ f \end{bmatrix}$$

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{I}_n \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

# Algebra Linear

- Por que matrizes?
  - Sistemas lineares
  - Interpretação geométrica: conjunto de vetores (tamanho, direção e sentido)

$$A\mathbf{v} = \lambda\mathbf{v}$$

$$A = V \text{diag}(\lambda) V^{-1}$$

autovetores e autovalores  
ou  
decomposição de valor singular  
(SVD)

