AGA 0505 - Análise de Dados em Astronomia

11. Aprendizado de Máquina: Deep Learning

Laerte Sodré Jr.

1o. semestre, 2023

aula de hoje:

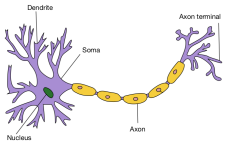
1. o que são redes de neurônios artificiais
2. funções de ativação
3. o perceptron multi-camadas
4. o teorema da universalidade
5. treinamento e back-propagation
6. redes convolucionais
7. redes convolucionais pré-treinadas
8. regressão e classificação
9. monitoramento do treinamento
10. dicas: *the universal workflow of ML* (Chollet)
11. a fauna das redes neurais

Learning is a lifelong journey, especially in the field of AI , where we have far more unknowns on our hands than certitudes. So please go on learning, questioning, and researching. Never stop. Because even given the progress made so far, most of the fundamental questions in AI remain unanswered. Many haven't even been properly asked yet.

François Chollet, Final Words, em Deep Learning with R

o que são redes de neurônios artificiais (RNA ou NN)

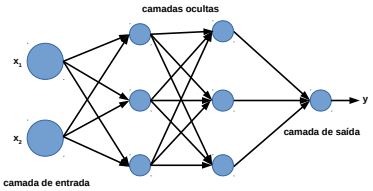
- método de processamento de informação inspirado no cérebro humano



- estrutura: grande número de *unidades de processamento* simples- os *neurônios artificiais*- conectadas

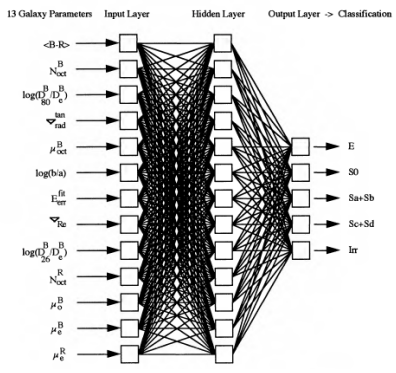
- uma RNA “aprende” com os dados- a “inteligência” da RNA está armazenada nos “pesos” entre as conexões dos neurônios

- vantagens:
  - não-linearidade: modela dados complexos
  - adaptatividade: aprende com os dados
  - tolerante a falta: natureza distribuída da informação
  - processamento massivamente paralelo



redes de neurônios artificiais

uma RNA aprende uma função: $y = f(\mathbf{x})$



Storrie-Lombardi et al. (1992)

- tipos de arquitetura:
  redes *feed-forward* rasas, profundas e convolucionais, recorrentes, autoencoders, GANs (redes adversariais gerativas),...
- estratégias de aprendizado:
  supervisionado (multilayer perceptron)
  não-supervisionado (Kohonen), com reforço (carros auto-conduzidos)
- funções de ativação:
  sigmoide, ReLU, linear
- parâmetros: pesos w
  medem a intensidade entre as conexões

funções de ativação (“neurônios artificiais”)

- função de ativação (não-linear!): calcula o sinal de saída de um neurônio a partir dos sinais de entrada
  - sigmoide:

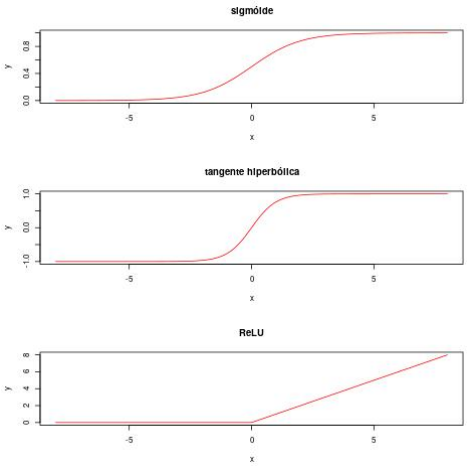
$$f(x) = \frac{1}{1 + e^{-x}}$$

  - tangente hiperbólica:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

  - ReLU (Rectified Linear Unit):

$$f(x) = max(0, x)$$

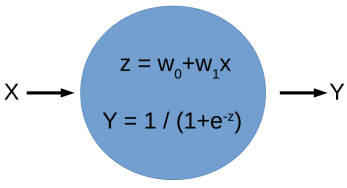


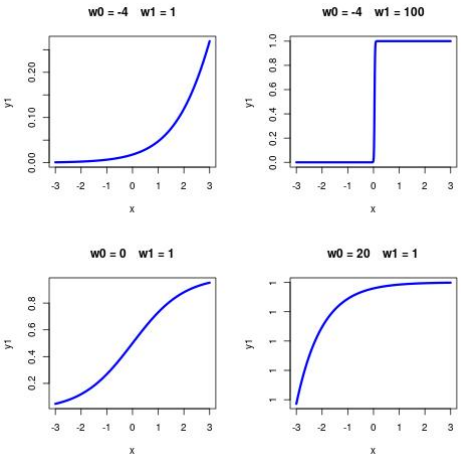
a sigmoide

- sigmoide como função:

$$f(z) = \frac{1}{1 + e^{-z}}$$

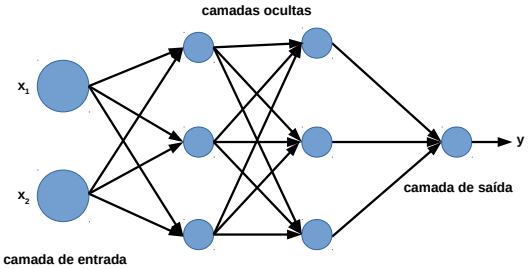
$$y = f(z = w_0 + w_1 x)$$





MLP: o perceptron multi-camadas

- MLP: *multilayer perceptron*
- arquitetura:
  - camada de entrada
  - uma ou mais camadas ocultas
  - camada de saída
  - cada camada é totalmente conectada à seguinte:
    redes densamente conectadas
- inferência: *forward pass*
  em cada camada a rede computa a saída de cada neurônio e passa para os neurônios da camada seguinte, até a camada de saída



como a rede aprende?

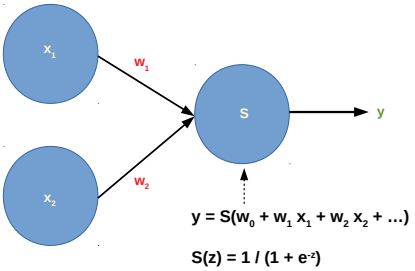
exemplo- *regressão logística* como uma RNA:

- classificação binária:
  classes 0 (estrela) e 1 (galáxia)
- x: input
  ex.: cores e outras medidas de um objeto
- ativação: sigmoide
- dado um objeto, a rede estima a probabilidade y de sua classe ser 1:

$$y = P(y = 1|x) = S(w) = \frac{1}{1 + e^{-z}}$$

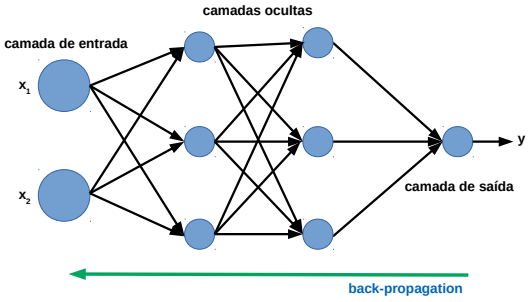
$$z = w_0 + w_1 x_1 + w_2 x_2 + ...$$

- aprendizagem: determinação dos pesos w



como a rede aprende?

- otimização: descida do gradiente por *back-propagation*:
  - a partir da diferença entre o valor y calculado e o *target*, ajustam-se os pesos para reduzir esta diferença
  - primeiro ajusta os pesos da última camada
  - propaga o erro para a camada anterior
  - ajusta os pessos dessa camada
  - repete este procedimento para todas as camadas



aprendizagem por back-propagation

exemplo- regressão logística como rede neural:

- output $y = S(w) = (1 + e^{-z})^{-1}$
  $z = w_0 + w_1 x_1 + w_2 x_2$ (para 2 atributos)

- função de custo para um target t (0 ou 1)
  entropia cruzada binária:
  $E = -[t \log y + (1 - t) \log(1 - y)]$

- regra da cadeia:
  $\frac{dE}{dw_k} = \frac{dE}{dy} \frac{dy}{dz} \frac{dz}{dw_k}$
  $\frac{dE}{dy} = -\left[\frac{t}{y} - \frac{1-t}{1-y}\right], \quad \frac{dy}{dz} = y(1 - y),$
  $\frac{dz}{dw_k} = x_k \qquad (x_0 = 1)$

logo,

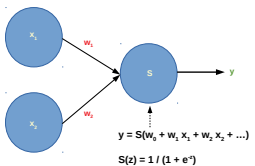
$$\frac{dE}{dw_k} = -[t - y]x_k$$

- ajuste dos pesos por descida do gradiente:

$$\Delta w_k = \alpha[t - y]x_k$$

$\alpha > 0$: coeficiente de aprendizado

- MLP: backpropagation camada por camada

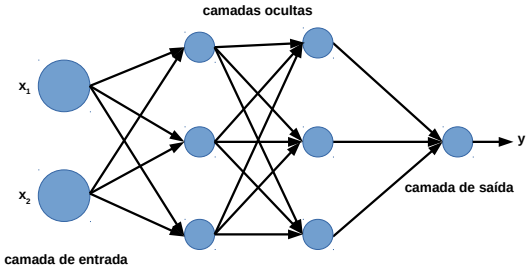


teorema da universalidade

- qualquer função real contínua $f : \mathbf{R}^N \longrightarrow \mathbf{R}^M$ pode ser realizada com uma única camada oculta, com um número suficiente de neurônios

- camada oculta: as camadas de neurônios entre a entrada e a saída de uma rede

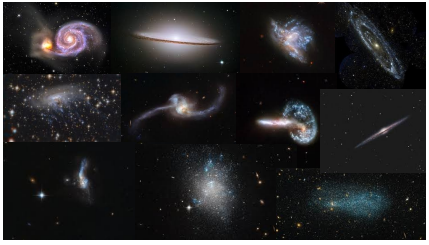
- porquê deep?
  - deep: muitas camadas ocultas
  - com várias camadas fica em geral mais fácil representar funções e treinar a rede

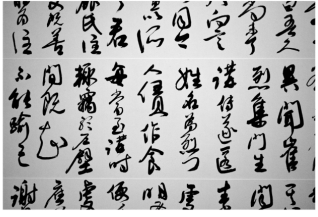


prova visual do teorema: http://neuralnetworksanddeeplearning.com/chap4.html

redes convolucionais- convnets ou CNNs

- porquê é difícil para um algoritmo reconhecer objetos?
  - variação imensa em imagens de um mesmo tipo de objeto
  - segmentação- identificação dos pixels de um certo objeto numa imagem: a um mesmo tipo de objeto podem corresponder um número grande de imagens
  - invariâncias: temos facilidade em reconhecer variações que não afetam a forma
  - deformações: as classes são muito variadas: forma das galáxias, da caligrafia...

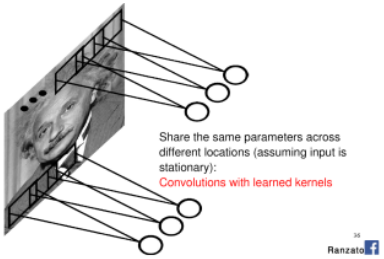




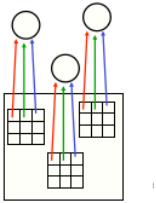
princípios das convnets

propostas por LeCun, 1998

- usadas principalmente com imagens
- usa camadas localmente conectadas, em contraposição a camadas totalmente conectadas, como no MLP
- usa múltiplas cópias de “detectores” ou “filtros”, em diferentes posições
- camada convolucional: cada unidade oculta conecta-se a uma pequena região da imagem, compartilhando os pesos
- cada camada contém múltiplos filtros

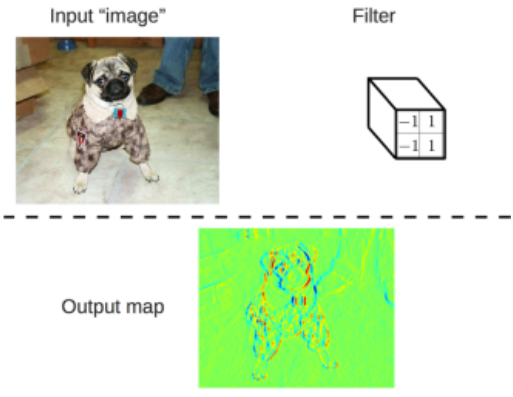


The red connections all have the same weight.



filtros

- If our filter is $[-1, 1]$, you get a vertical edge detector



os filtros são “aprendidos”

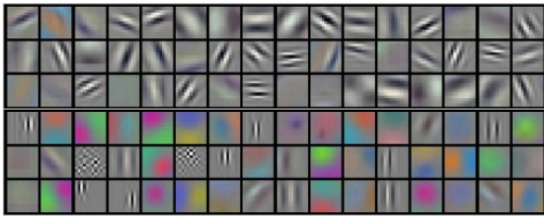
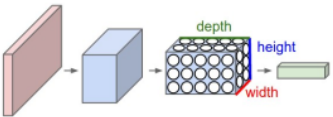


Figure : Filters in the first convolutional layer of Krizhevsky et al

hiper-parâmetros e *pooling*

hiperparâmetros:

- número de filtros: profundidade do volume de saída
- passo (*stride*): separação entre os filtros controla o tamanho do volume de saída
- tamanho $w \times h$ dos filtros



pooling:

- cada camada convolucional é seguida por uma de *pooling*
- extrai o valor máximo (ou médio) de um conjunto de filtros
- diminui a dimensão da camada anterior

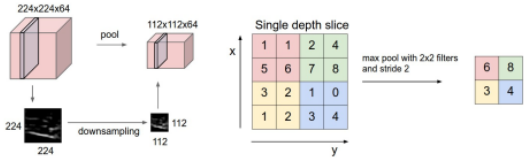
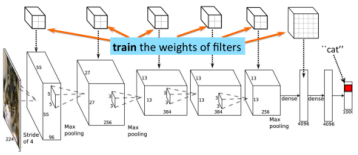
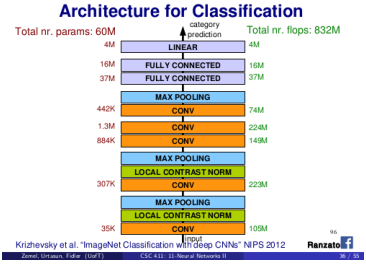


Figure : **Left:** Pooling, **right:** max pooling example

estrutura das convnets

- “pilhas” de camadas de convolução e pooling usando como input a saída da camada anterior
- isso permite à rede implementar uma variedade de filtros
- termine a rede com uma ou duas camadas densamente conectadas para fazer classificação ou regressão
- treinamento: variante de back-propagation

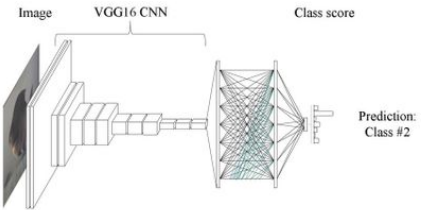




redes pré-treinadas

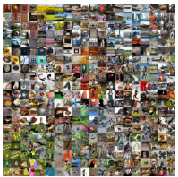
- uma abordagem interessante em problemas de classificação de imagens é usar uma rede pré-treinada com um conjunto muito grande de imagens
- Ex: ImageNet- base de dados contendo 14 milhões de imagens, classificadas em 1000 categorias diferentes

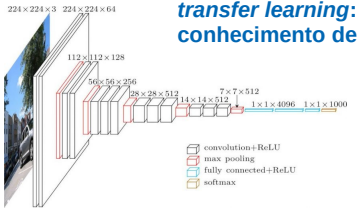
- em geral usa-se a parte convolucional da rede pré-treinada para gerar atributos que então são passados para uma rede densa que é treinada para classificar os dados de interesse
- a lógica é que a parte convolucional da rede aprendeu filtros que podem ser úteis em outros problemas de análise de imagens



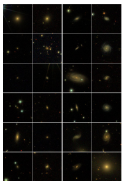
redes pré-treinadas

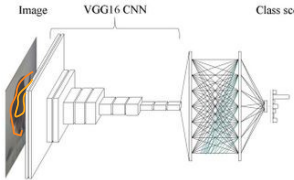
VGG16 (Simonyan & Zisserman): vencedora da competição ILSVR(Imagenet) em 2014





***transfer learning*: transferência do conhecimento de uma rede para outra**

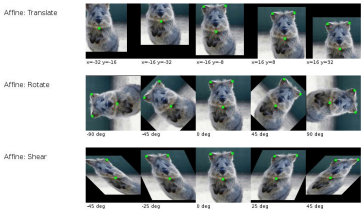




galáxia espiral

overfitting

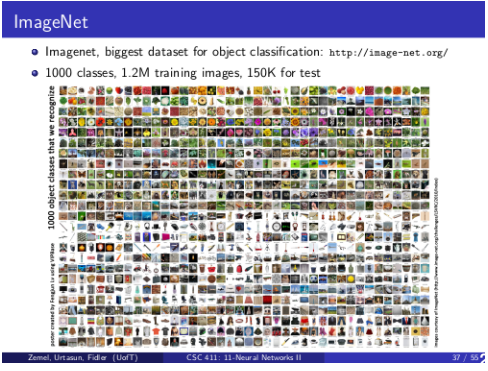
- tem-se que tomar cuidado com CNNs: como o número de parâmetros é muito alto, o risco de overfitting também é
- monitore a função de custo e a acurácia com o conjunto de validação
- duas estratégias são normalmente usadas para reduzir overfitting em análise de imagens: *data augmentation* e *dropout*



- *data augmentation*:
  - “multiplica-se” uma imagem incluindo-se variantes dela no treinamento
  - transformações: reflexão, translação, shear, etc
- *dropout*:
  - introduz-se uma camada de dropout antes da camada densa
  - isso significa remover aleatoriamente (pôr igual a 0) um certo número de saídas de uma camada durante o treinamento.

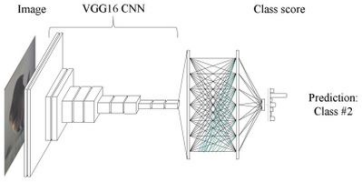
dicas: *the universal workflow of ML* (Chollet)

- normalização dos dados:
  subtrai-se a média e divide-se pelo desvio padrão OU normalização entre 0 e 1
- quanto mais exemplos, melhor!
  data augmentation: adicione inversões e rotações nas imagens, etc
- classificação: treine com dados balanceados
- randomize os dados antes dos mini-batchs
- quanto mais camadas, melhor! primeiro crie uma rede complexa o suficiente para overfitar e depois a simplifique ou introduza regularização/dropout para evitar o overfitting
- se sua amostra é pequena, use uma rede treinada num grande conjunto de dados (tipo Imagenet) e depois treine com seus dados num módulo adicional (*transfer learning*)



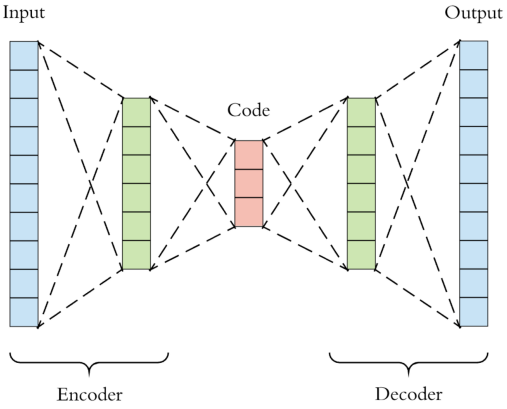
a fauna das redes neurais

- redes densas: MLP - multi-layer perceptron totalmente conectado
- redes convolucionais
- redes recorrentes
- long/short term memory (LSTM) - um tipo de rede recorrente
- generative adversarial networks (GAN)
- redes bayesianas
- auto-encoders: simples, variacionais, esparsos, ...
- Boltzmann machine
- redes de Kohonen
- ... e muito mais!!!!



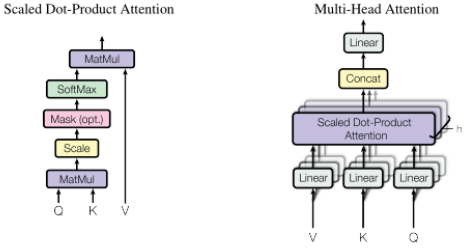
autoencoders

- objetivo: obter uma versão comprimida dos dados *o espaço latente*
- treinada com entrada = saída
- o *encoder* codifica/comprime os dados
- a camada central tem menos neurônios que a entrada ou a saída
- espaço latente: formado pelas ativações da cmada central (*code*)
- se as ativações são lineares, o espaço latente é equivalente ao PCA



atenção

Vaswani et al., arXiv:1706.03762.pdf



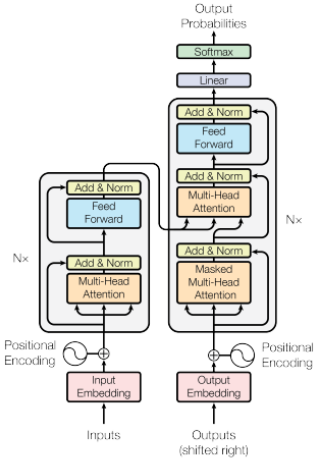
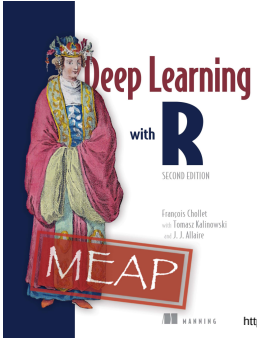


Figure 1: The Transformer - model architecture.

bibliografia

- Deep Learning with R, François Chollet, com J.J. Allaire, 2018 (https://www.manning.com/books/deep-learning-with-r)
- Deep Learning, Ian Goodfellow, Yoshua Bengio & Aaron Courville, 2016 (https://www.deeplearningbook.org/)



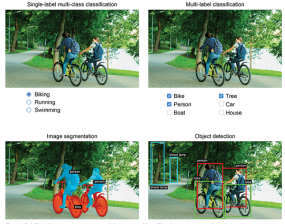


Figure 9.1 The three main computer vision tasks: Classification, segmentation, detection

https://www.manning.com/books/deep-learning-with-r-second-edition

muitos recursos...