Deep Learning: e suas unidades de processamento

Moacir Antonelli Ponti
ICMC, Universidade de São Paulo
Escola Avançada em Big Data Analysis

`www.icmc.usp.br/~moacir` — `moacir@icmc.usp.br`

São Carlos-SP/Brasil – 2020

Agenda

Machine vs Deep Learning

Dados estruturados e independentes: Perceptron

Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

Convergência e aprendizado

Agenda

Machine vs Deep Learning

Dados estruturados e independentes: Perceptron

Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

Convergência e aprendizado

Machine learning: dois exemplos

Precisamos inferir uma função $f(x) = y$

— o significado de f, x e y dependem da tarefa

Machine learning: dois exemplos

Precisamos inferir uma função $f(\mathrm{x}) = \mathrm{y}$
— o significado de f, x e y dependem da tarefa

1 – classificação de imagens de paisagens

- Dados disponíveis: pares (imagens, rótulos) obtidas de desertos e praias,
- Entrada: pixels da imagem organizados na forma x,
- Saída: rótulo y (e.g. praia) atribuído à imagem de entrada.

Machine learning: dois exemplos

2 – predição de fraude em transação de cartão de crédito

- Dados disponíveis: transações legítimas de um cliente,
- Entrada: dados incluindo: localização, moeda, valor, data e hora, na forma x,
- Saída: probabilidade y de observar uma transação fraudulenta (anômala).

Machine Learning (ML) vs Deep Learning (DL)

Machine Learning

Uma área mais geral que inclui DL.

Algoritmos comumente aprendem uma função $f : X \to Y$, a partir de um espaço de funções admissíveis f e dados de treinamento

- métodos rasos ("shallow") comumente inferem uma única $f(.)$. e.g. uma função linear $f(x) = w \cdot x + b$,
  - aprendizado de máquina seria ajustar os valores para w e b
  - examplos: Perceptron, Support Vector Machines (SVM), Logistic Regression Classifier, Linear Discriminant Analysis (LDA).

Machine Learning (ML) vs Deep Learning (DL)

Deep Learning

Envolve aprender uma representações, aprendidas de forma hierárquica por funções compostas.
Por exemplo, dada uma entrada x_1 produzir diversas representações intermediárias:

$$x_2 = f_1(x_1)$$
$$x_3 = f_2(x_2)$$
$$x_4 = f_3(x_3)$$
$$\dots$$

A saída é obtida pelo aninhamento de L funções:

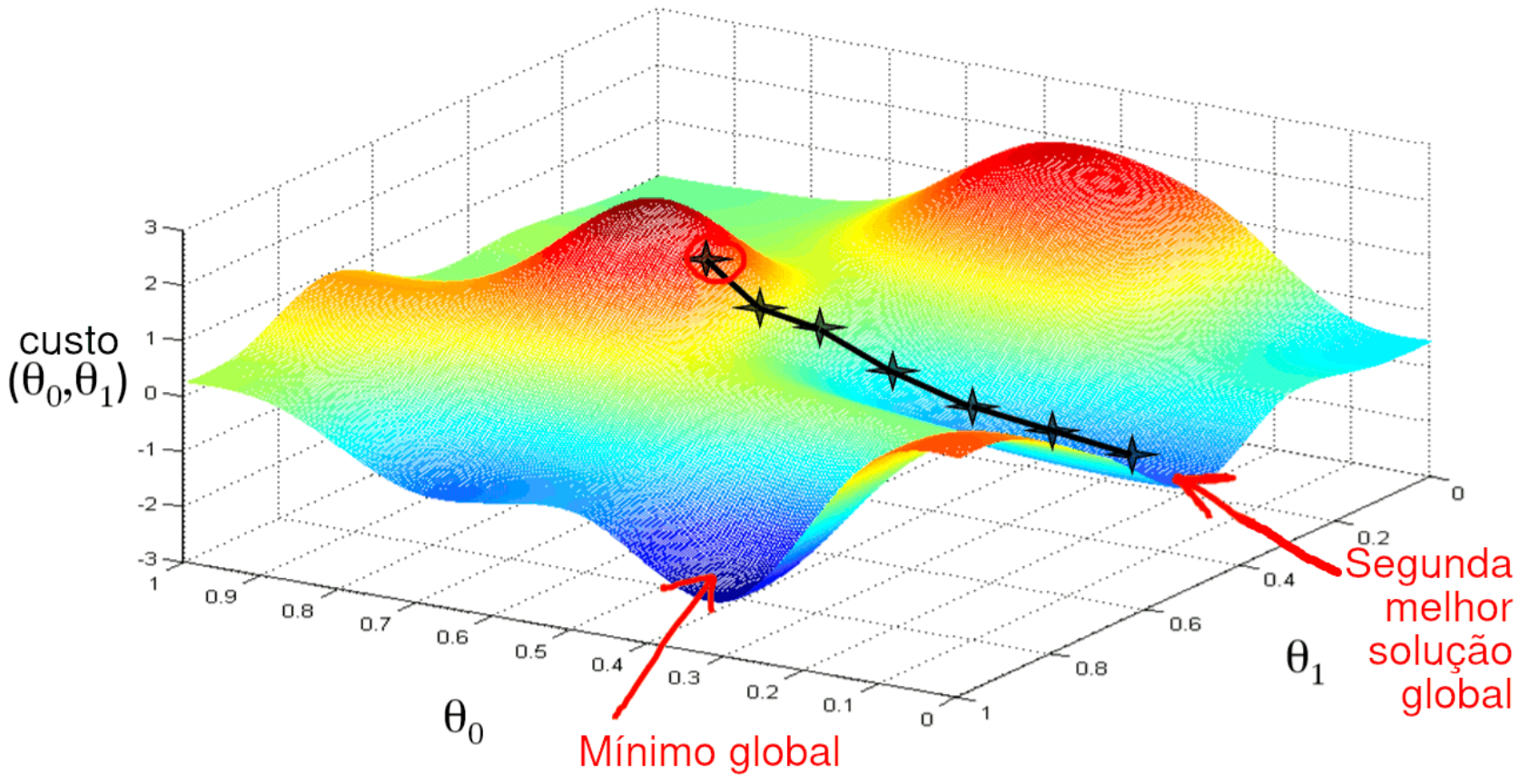
$$f_L\left(\cdots f_3(f_2(f_1(x_1, \Theta_1), \Theta_2), \Theta_3) \cdots, \Theta_L\right),$$

Θ_i são os parâmetros associados a cada função i.

Componentes importantes: treinamento

Processo de ajuste dos pesos com base em uma **função de custo** de escolher determinados parâmetros

- queremos andar na direção do vale, em busca do mínimo global
- tipicamente atualizações são feitas usando um *batch* (subconjunto) de instâncias



Agenda

Machine vs Deep Learning

Dados estruturados e independentes: Perceptron

Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

Convergência e aprendizado

Montando um classificador

Entrada  $\rightarrow$ x

Seja Θ uma matriz W de pesos e um vetor b de termos "bias"

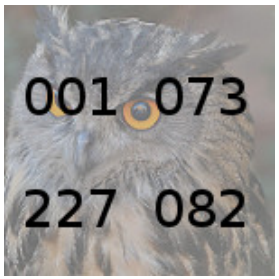
matriz de pesos | imagem | bias

$$f(\Theta, x) = W \quad x \quad + \quad b$$

$$= \text{ scores para possíveis classes de x}$$

Montando um classificador

- Entrada: imagem (com $N \times M \times 3$ pixels) vetorizada em x
- Classes: gato, tartaruga, coruja
- Saída: scores para cada classe



$= \mathrm{x} = [1, 73, 227, 82]$

saída $f(\Theta, \mathrm{x}) = s \quad \rightarrow$ 3 números com os scores das classes

Montando um classificador

- Entrada: imagem (com $N \times M \times 3$ pixels) vetorizada em x
- Classes: gato, tartaruga, coruja
- Saída: scores para cada classe

$= \mathrm{x} = [1, 73, 227, 82]$

saída $f(\Theta, \mathrm{x}) = s \quad \rightarrow$ 3 números com os scores das classes

$$\underbrace{\begin{bmatrix} 0.1 & -0.25 & 0.1 & 2.5 \\ 0 & 0.5 & 0.2 & -0.6 \\ 2 & 0.8 & 1.8 & -0.1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 73 \\ 227 \\ 82 \end{bmatrix} + \begin{bmatrix} -2.0 \\ 1.7 \\ -0.5 \end{bmatrix}}_{W\mathrm{x}+\mathrm{b}} = \begin{bmatrix} -1.3 \\ 0.3 \\ 8.6 \end{bmatrix}$$

Montando um classificador: aplicando ativação

Função Sigmóide (mapeia valores para intervalo 0-1)

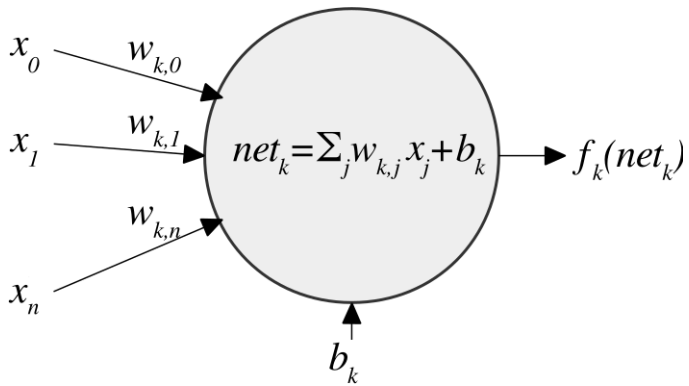
$$sig\left(\begin{bmatrix} -1.3 \\ 0.3 \\ 8.6 \end{bmatrix}\right) = \begin{bmatrix} 0.21 \\ 0.57 \\ 0.99 \end{bmatrix}$$

Função Softmax (garante valores positivos e vetor com soma unitária)

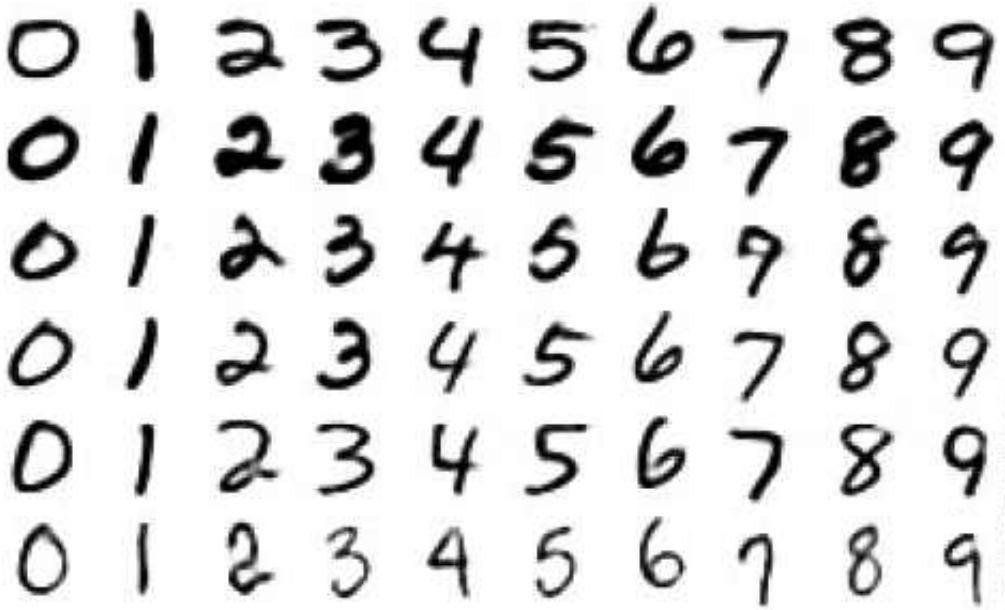
$$softmax\left(\begin{bmatrix} -1.3 \\ 0.3 \\ 8.6 \end{bmatrix}\right) = \begin{bmatrix} 0.0 \\ 0.13 \\ 0.87 \end{bmatrix}$$

Neurônio Perceptron: unidade densa

- entrada: valores organizados em um vetor
- saída: um único valor
  - cada valor de entrada é associado a um peso w (força da conexão)
  - o bias b funciona como intercepto da função
- aprender é ajustar w's e b's aos dados de treinamento

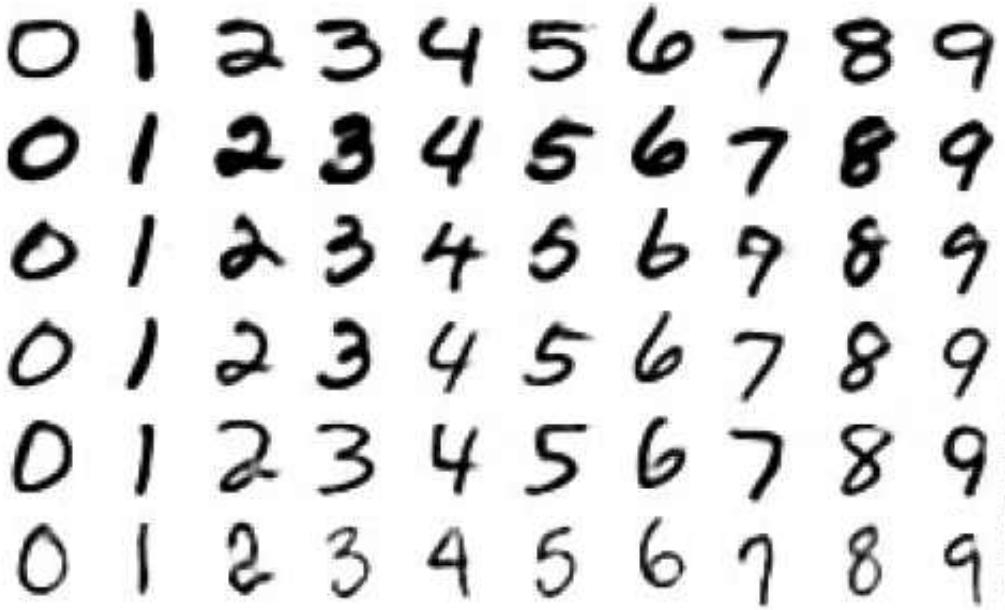


Exemplo de problema: classificação de dígitos



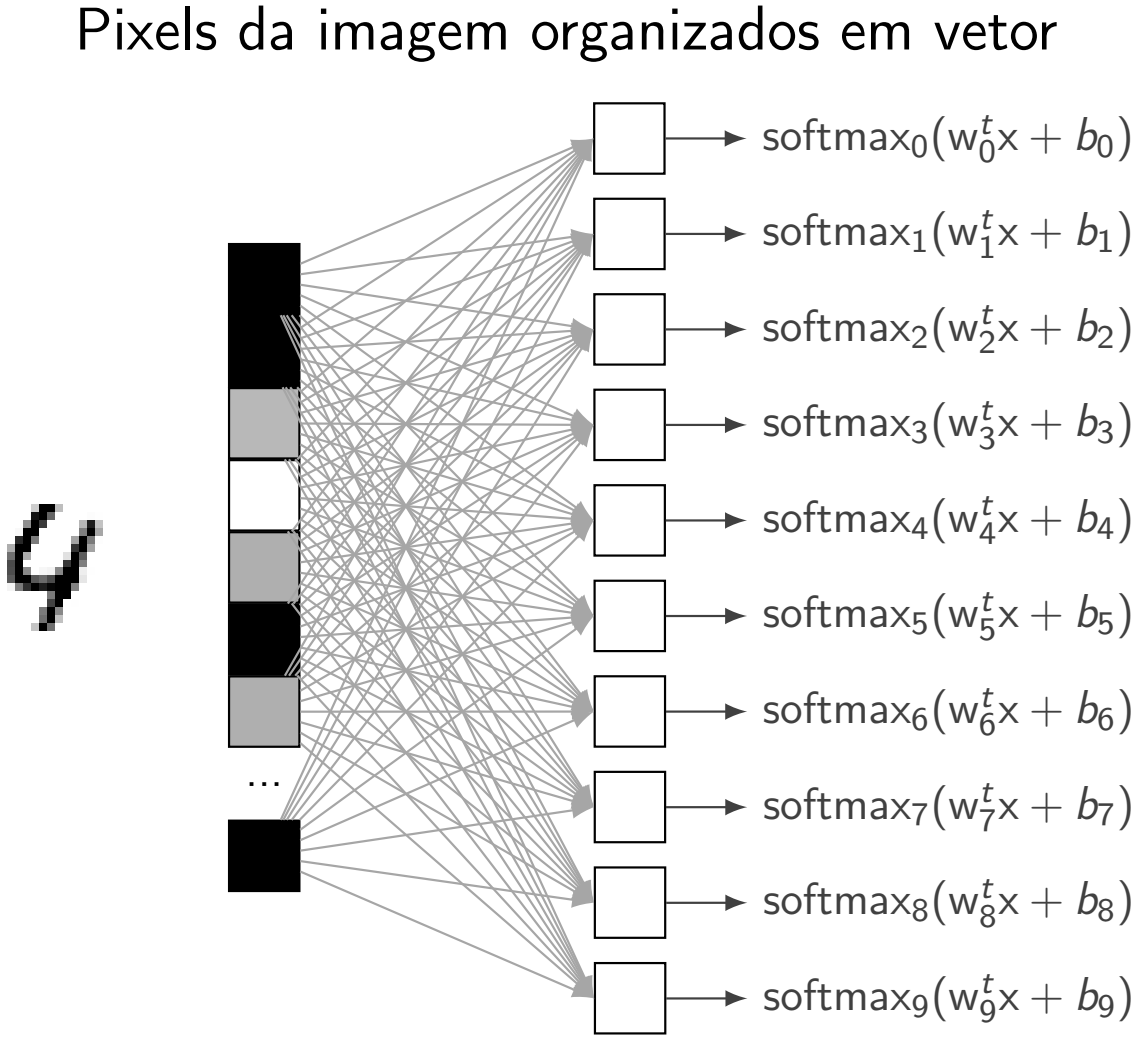
- Imagens com $28 \times 28 = 784$ pixels,

Exemplo de problema: classificação de dígitos

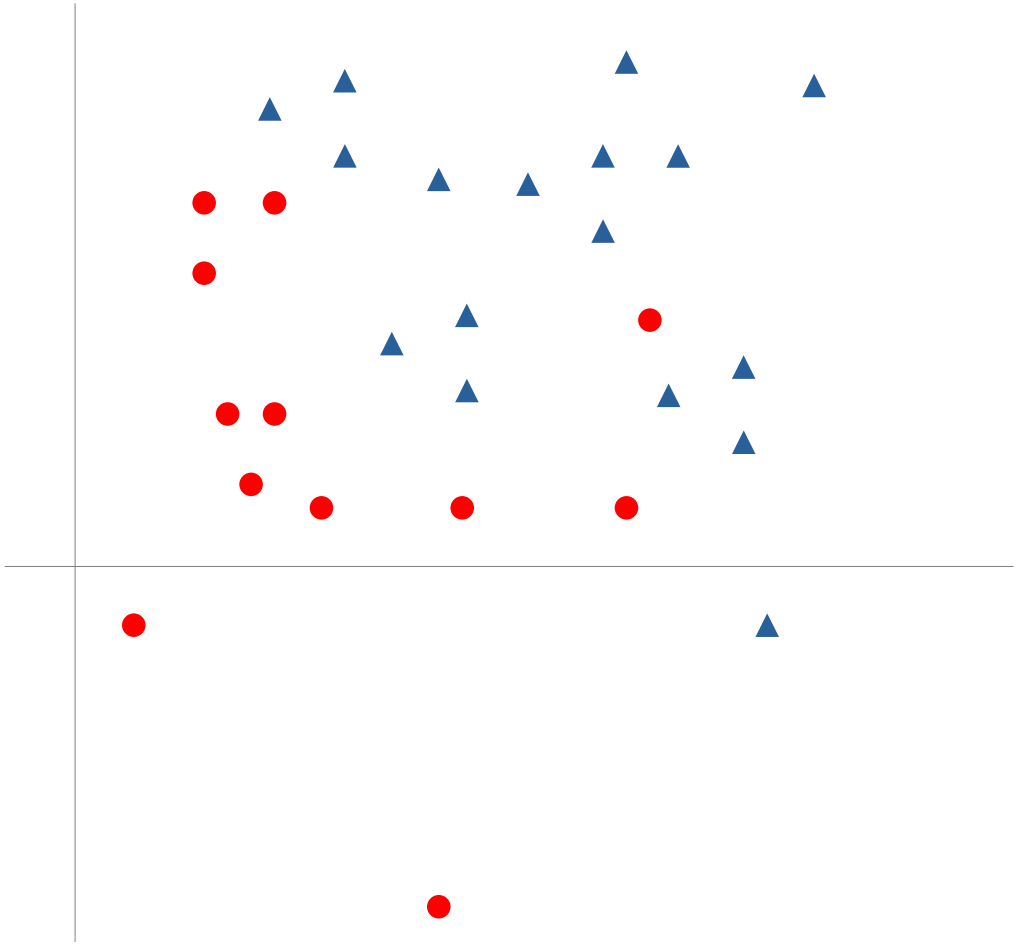


- Imagens com $28 \times 28 = 784$ pixels,
- Redes do tipo Perceptron,
- Algoritmo SGD com 32 imagens no batch,
- Camada de saída normalizada de forma a somar 1: softmax.

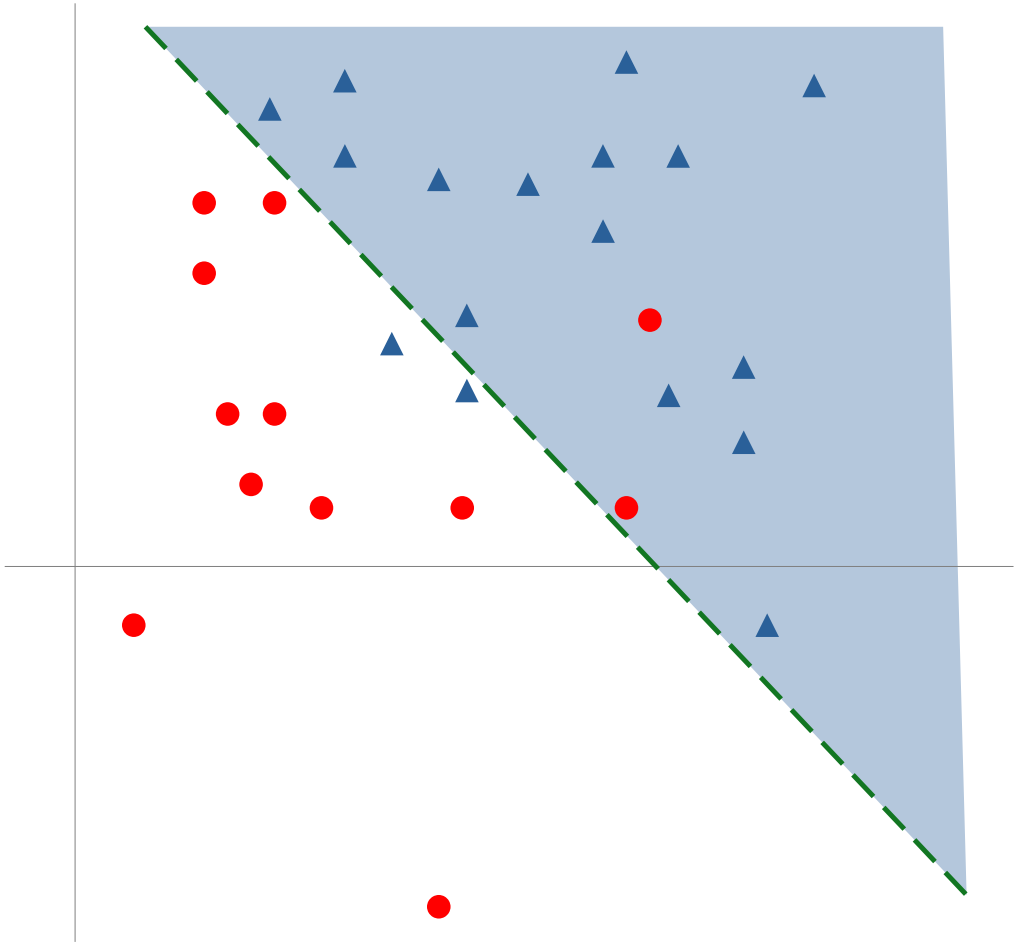
Rede neural rasa, com uma única camada



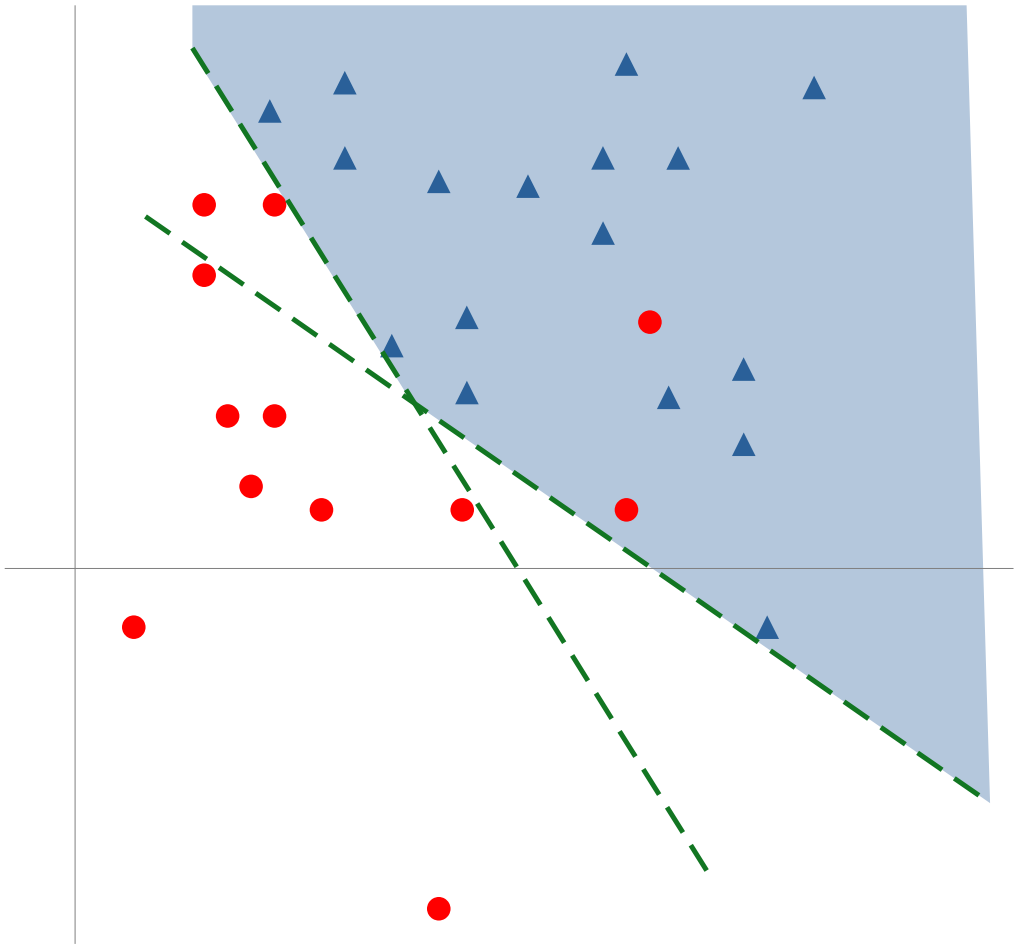
Perceptron Multicamadas: número de neurônios



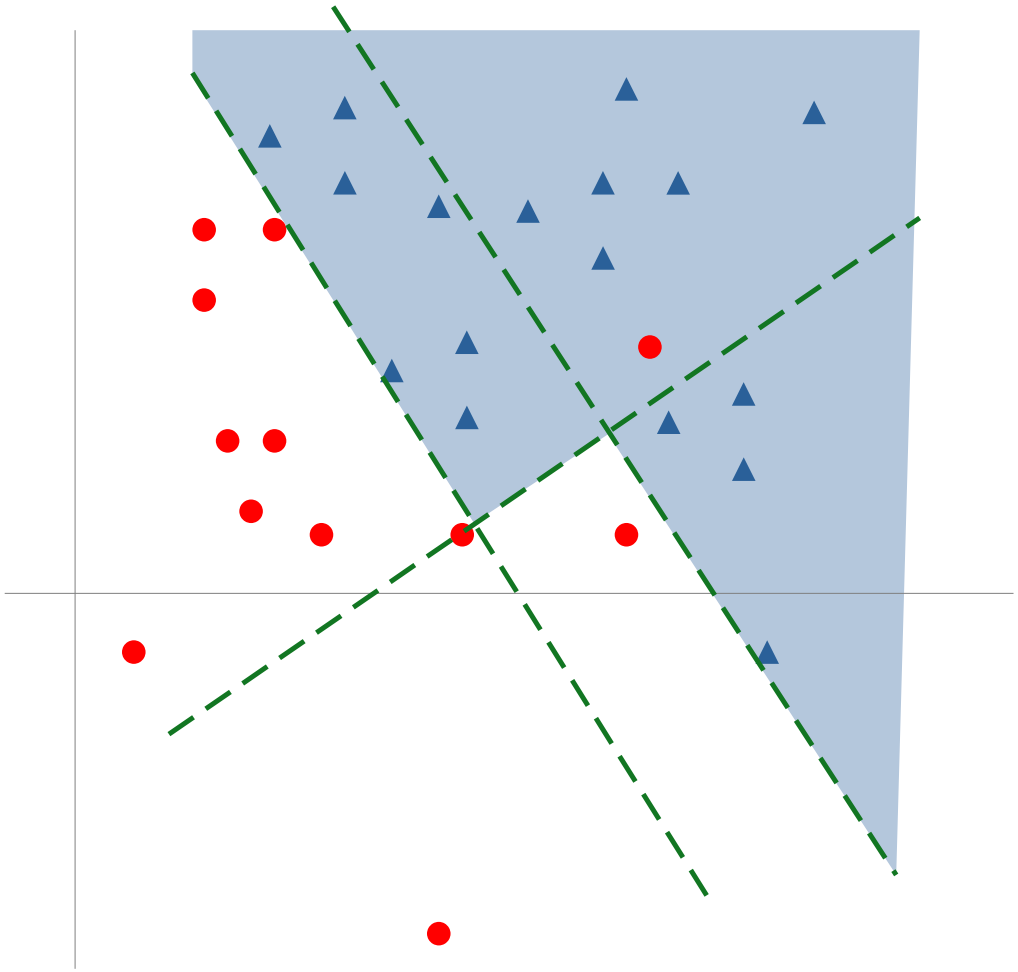
Perceptron Multicamadas: número de neurônios



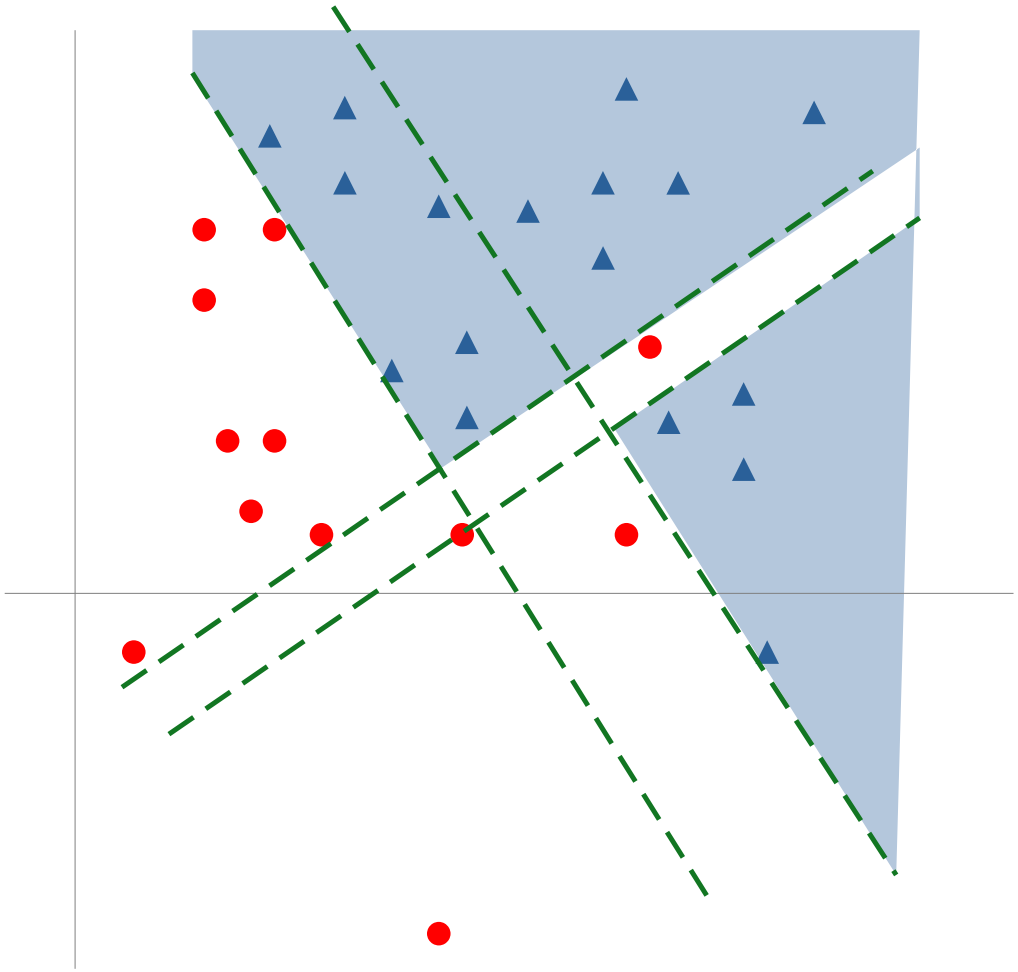
Perceptron Multicamadas: número de neurônios



Perceptron Multicamadas: número de neurônios



Perceptron Multicamadas: número de neurônios



Formulação da rede neural

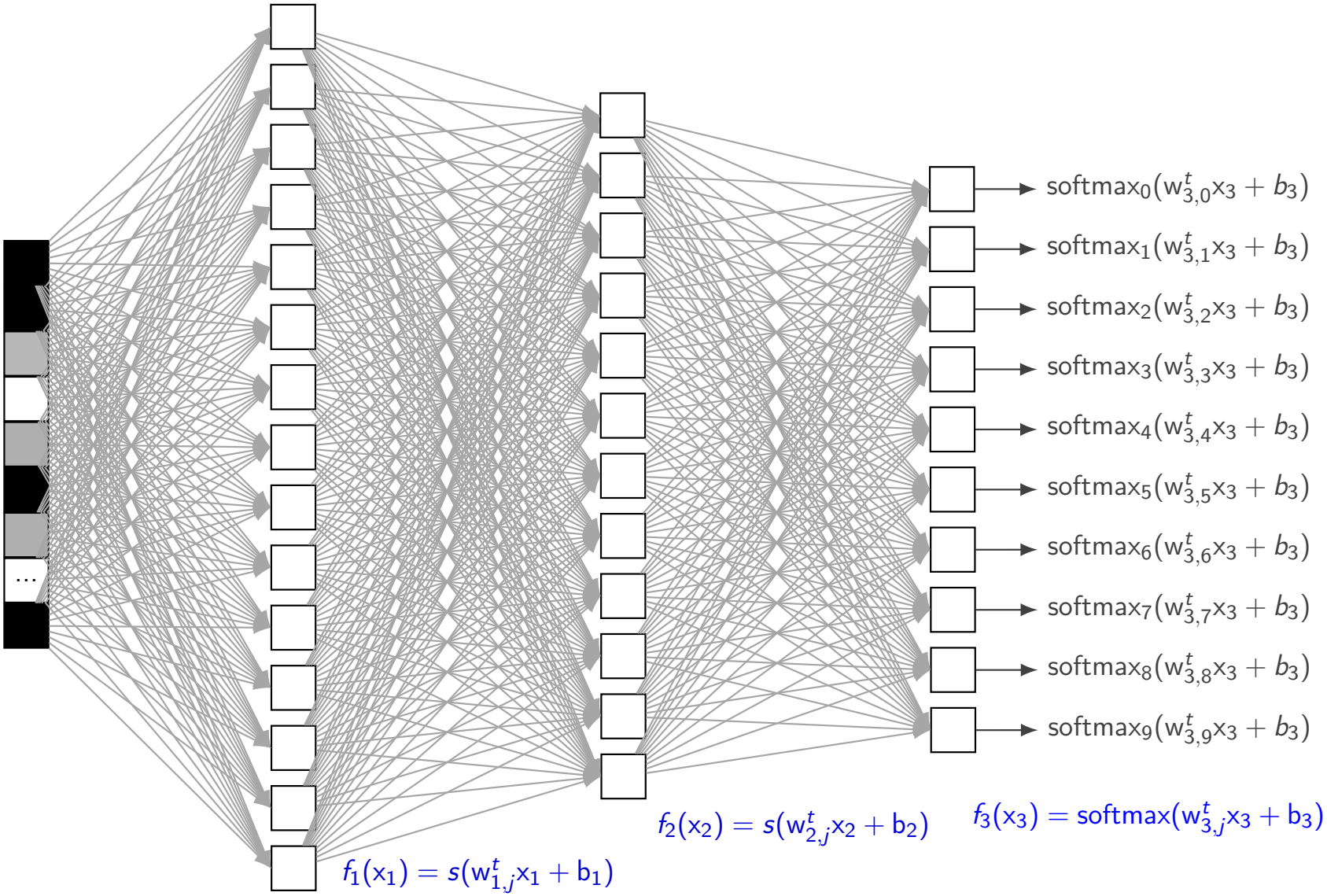
10 classes, batch-size 32, e 784 características (pixels) por imagem

$$\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \cdots & x_{0,783} \\ x_{1,0} & x_{0,1} & x_{1,2} & \cdots & x_{0,783} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{31,0} & x_{31,1} & x_{31,2} & \cdots & x_{31,783} \end{bmatrix} \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,9} \\ w_{2,0} & w_{2,1} & \cdots & w_{2,9} \\ \vdots & \vdots & \ddots & \vdots \\ w_{783,0} & w_{783,1} & \cdots & w_{783,9} \end{bmatrix} + \begin{bmatrix} b_0 & b_1 & b_2 & \ldots & b_9 \end{bmatrix}$$

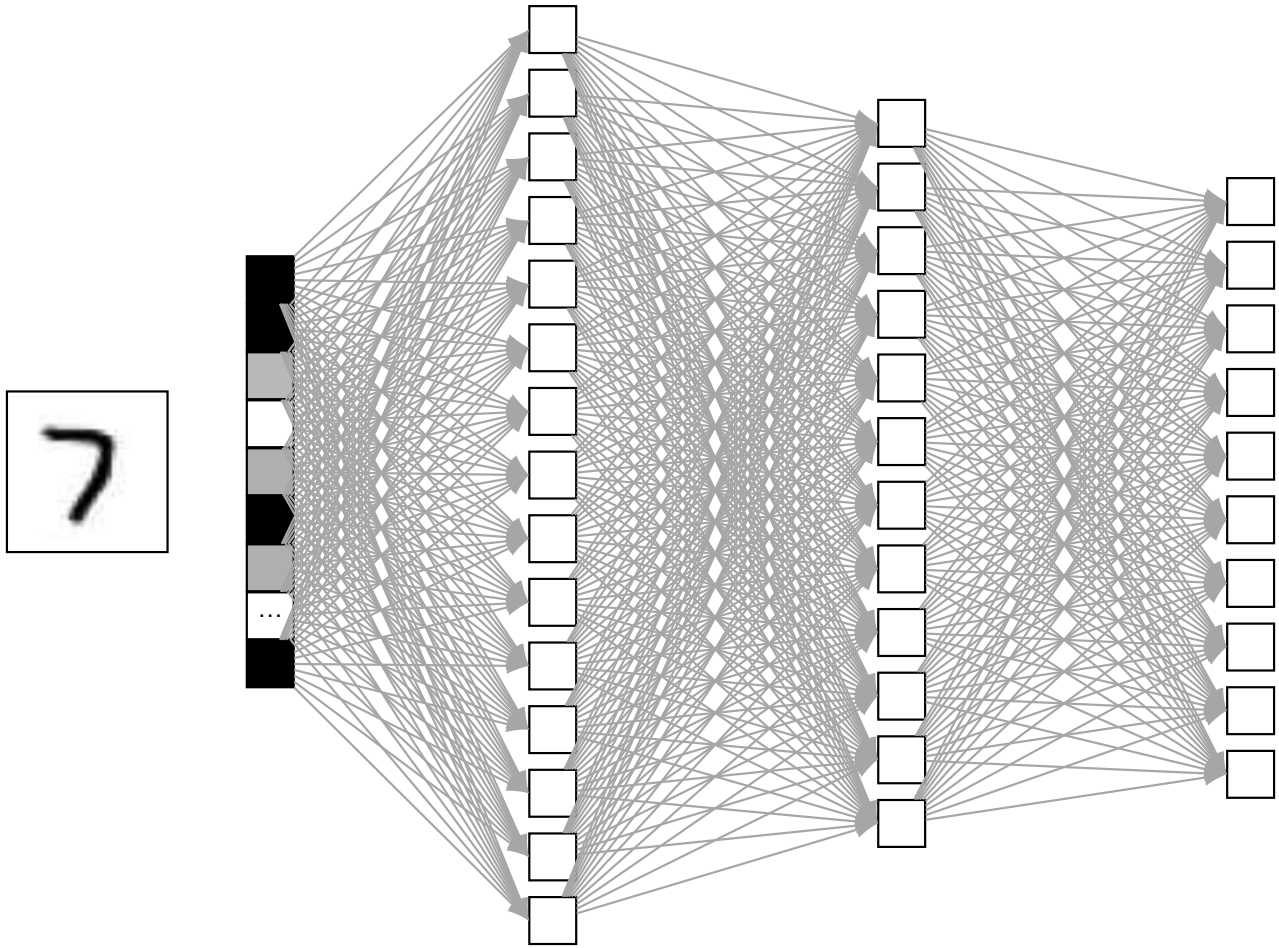
$$\mathrm{Y} = \mathrm{softmax}(\mathrm{X} \cdot \mathrm{W} + \mathrm{b})$$

$$\mathrm{Y} = \begin{bmatrix} y_{0,0} & y_{0,1} & y_{0,2} & \cdots & y_{0,9} \\ y_{1,0} & y_{1,1} & y_{1,2} & \cdots & y_{1,9} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{31,0} & y_{31,1} & y_{31,2} & \cdots & y_{31,9} \end{bmatrix}$$

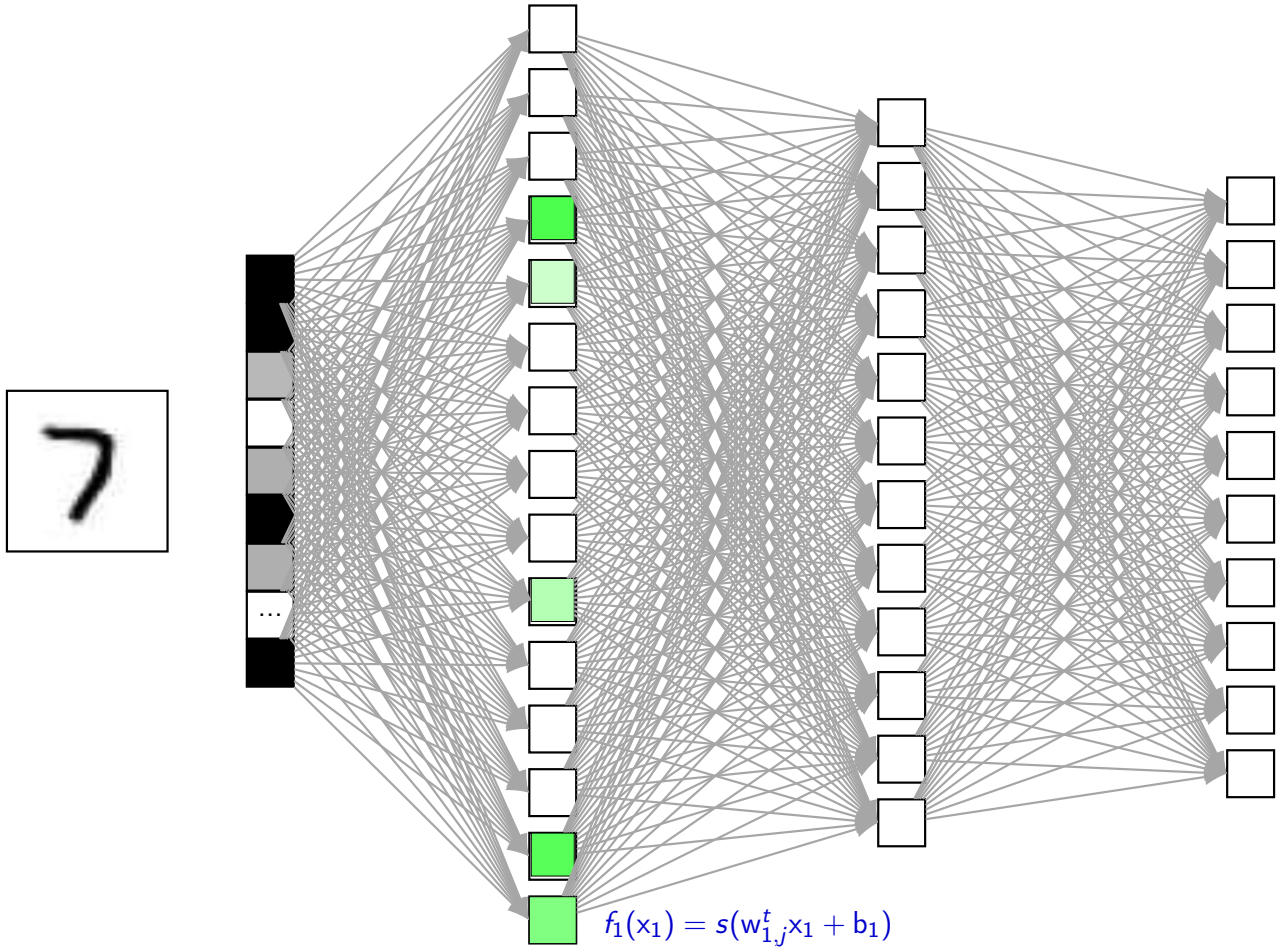
Rede MLP "profunda" com 2 camadas ocultas



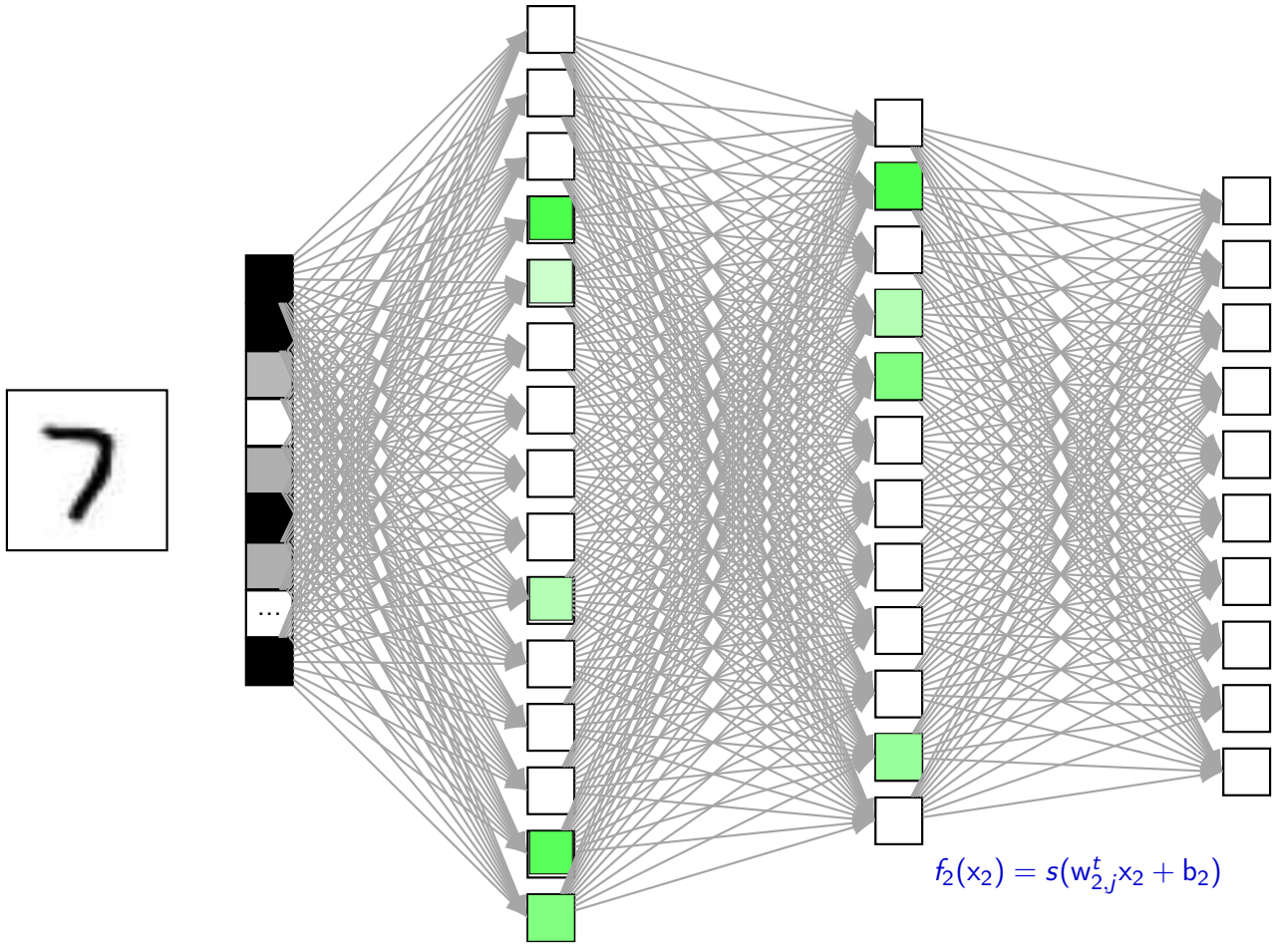
Rede MLP "profunda" com 2 camadas ocultas : Input



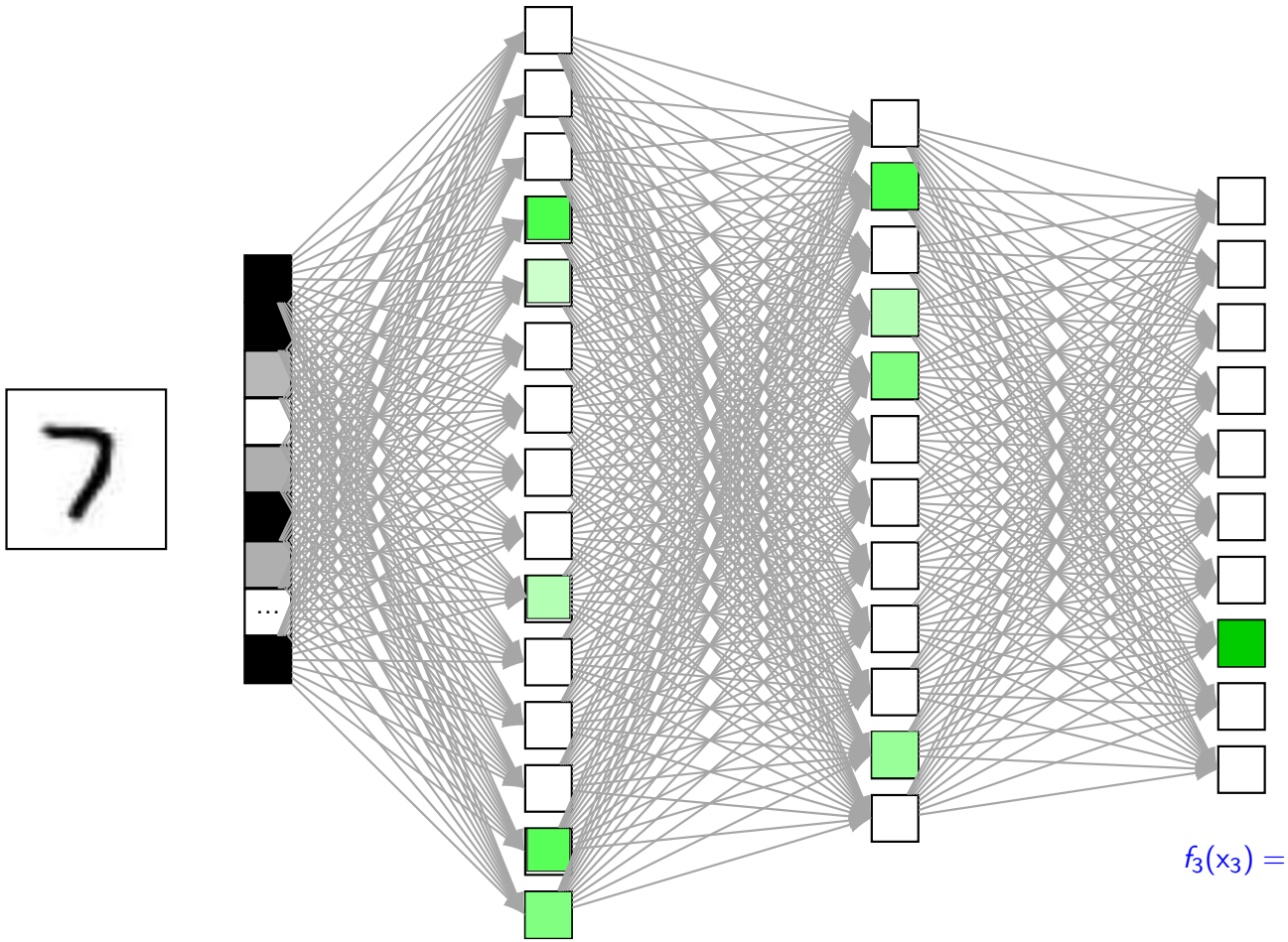
Rede MLP "profunda" com 2 camadas ocultas



Rede MLP "profunda" com 2 camadas ocultas



Rede MLP "profunda" com 2 camadas ocultas : output



$f_3(x_3) = \text{softmax}(w_{3,j}^t x_3 + b_3)$

Agenda

Machine vs Deep Learning

Dados estruturados e independentes: Perceptron

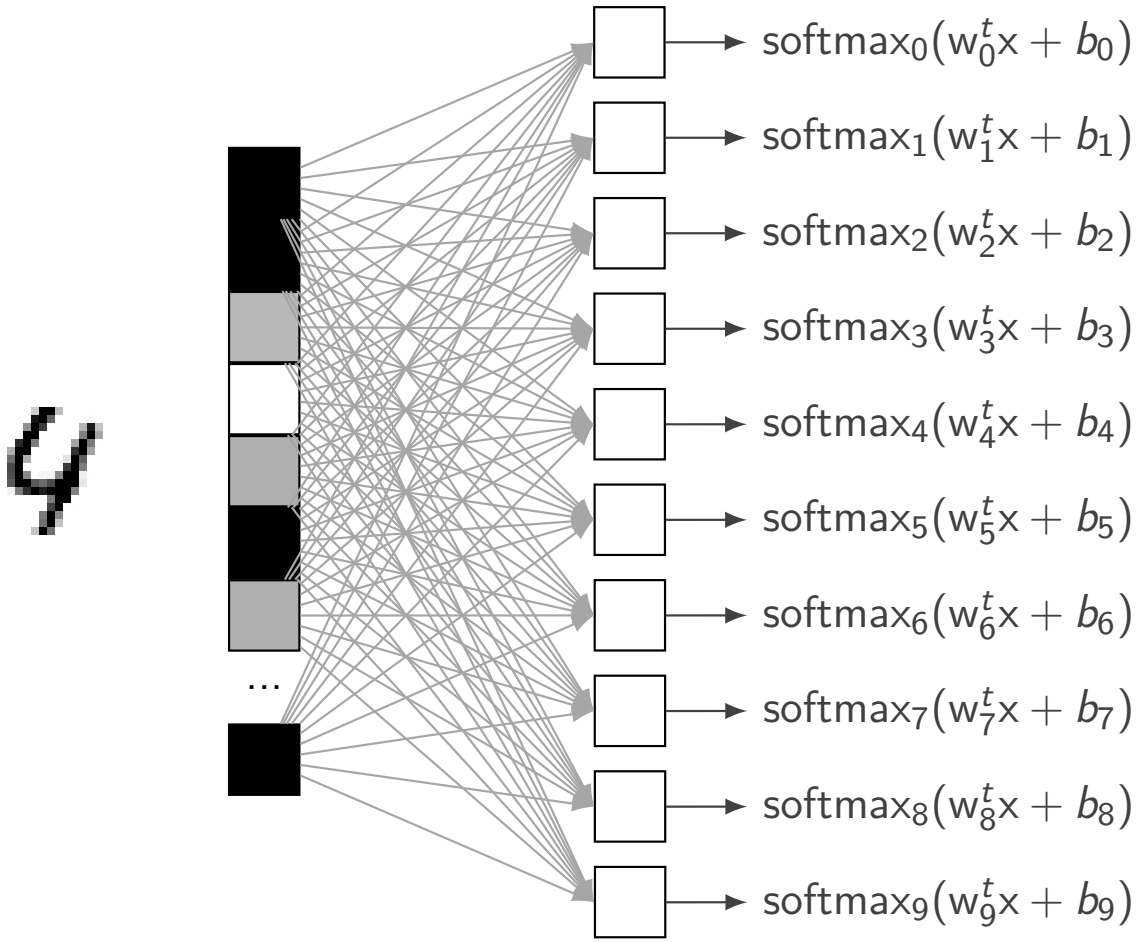
Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

Convergência e aprendizado

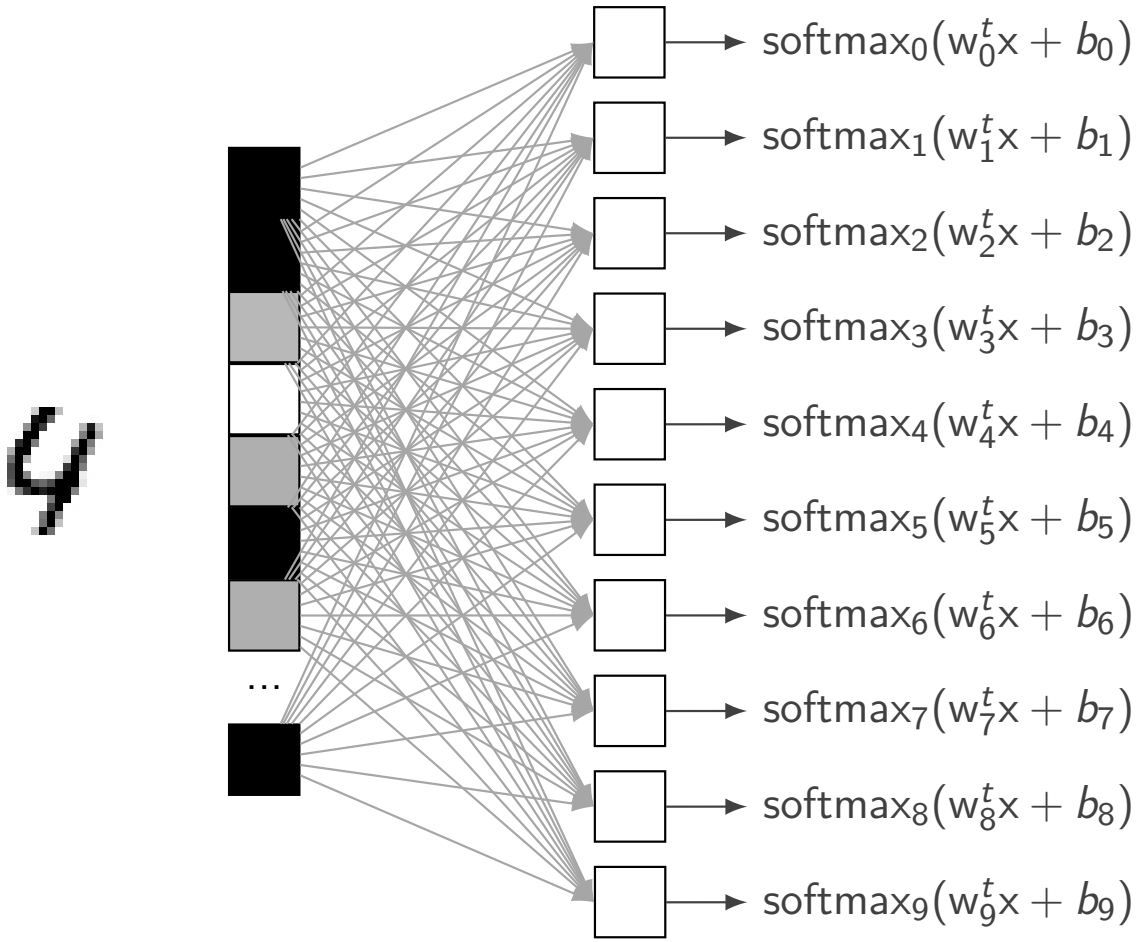
Questões importantes sobre as redes MLP (densas)

1. Valores de entrada (atributos) são considerados independentes



Questões importantes sobre as redes MLP (densas)

1. Valores de entrada (atributos) são considerados independentes
2. Não são aproveitadas relações locais entre os dados



Questões importantes sobre as redes MLP (densas)

1. Grande número de parâmetros: memória e processamento

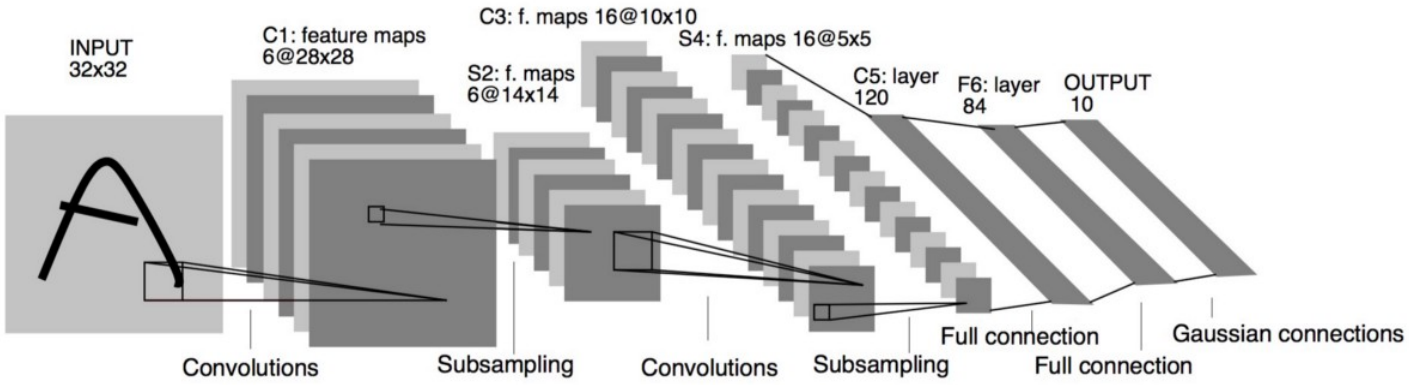
Questões importantes sobre as redes MLP (densas)

1. Grande número de parâmetros: memória e processamento
   - Exemplo: entrada imagem de $28 \times 28 = 784$
   - Uma camada com 100 neurônios teria..

Questões importantes sobre as redes MLP (densas)

1. Grande número de parâmetros: memória e processamento
   - Exemplo: entrada imagem de $28 \times 28 = 784$
   - Uma camada com 100 neurônios teria..
   - $78400 + 100 = 78500$ parâmetros a serem aprendidos e mantidos na memória durante o treinamento

Redes Neurais Convolucionais (CNNs)



(Arquitetura LeNet)

Nova terminologia:

- Camada convolucional (convolutional layer)
- Subamostragem (pooling)
- Mapas de Ativação (activation/feature maps)
- Camada densa (dense/fully connected, tipo MLP)

Convolução

- Operador que visa realizar uma combinação linear de valores locais da entrada
- Centrado em uma posição, e.g. (x, y), gera como saída um único valor de saída

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0							
1		1.5					
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5				
1					
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5			
1					
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1		
1					
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	
1					
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1					
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1	0.5				
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1	0.5	1.5			
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1	0.5	1.5	-2.5		
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1	0.5	1.5	-2.5	-2.5	
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1	0.5	1.5	-2.5	-2.5	1
2					
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1	0.5	1.5	-2.5	-2.5	1
2	3				
3					
4					

Convolução

Volume de entrada 7 x 7

	0	1	2	3	4	5	6
0	2	2	2	2	3	3	3
1	1	0	1	1	1	1	0
2	1	1	3	3	0	0	0
3	1	1	3	2	0	0	3
4	1	1	3	2	0	0	3
5	1	3	3	2	0	0	3
6	3	3	3	2	0	0	3

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4
0	1.5	2.5	1	1.5	0.5
1	0.5	1.5	-2.5	-2.5	1
2	3	3.5	-4.5	-5	1.5
3	3	2.5	-5	-4	4.5
4	3	0.5	-5	-4	4.5

Convolução

- **Zero-padding**: para compensar a impossibilidade de computar todos os valores;

Convolução

- **Zero-padding**: para compensar a impossibilidade de computar todos os valores;
  - Amplia-se a entrada de forma que o volume de saída seja igual ao de entrada

Convolução

Volume de entrada 7 x 7 + zero padding

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	3	0
1	0	1	0	1	1	1	1	0	0
2	0	1	1	3	3	0	0	0	0
3	0	1	1	3	2	0	0	3	0
4	0	1	1	3	2	0	0	3	0
5	0	1	3	3	2	0	0	3	0
6	0	3	3	3	2	0	0	3	0
7	0	0	0	0	0	0	0	0	0

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7 + padding

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	3	0
1	0	1	0	1	1	1	1	0	0
2	0	1	1	3	3	0	0	0	0
3	0	1	1	3	2	0	0	3	0
4	0	1	1	3	2	0	0	3	0
5	0	1	3	3	2	0	0	3	0
6	0	3	3	3	2	0	0	3	0
7	0	0	0	0	0	0	0	0	0

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída 7 x 7

	0	1	2	3	4	5	6
0	0						
1							
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7 + padding

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	3	0
1	0	1	0	1	1	1	1	0	0
2	0	1	1	3	3	0	0	0	0
3	0	1	1	3	2	0	0	3	0
4	0	1	1	3	2	0	0	3	0
5	0	1	3	3	2	0	0	3	0
6	0	3	3	3	2	0	0	3	0
7	0	0	0	0	0	0	0	0	0

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0	0	-1.5					
1							
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7 + padding

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	3	0
1	0	1	0	1	1	1	1	0	0
2	0	1	1	3	3	0	0	0	0
3	0	1	1	3	2	0	0	3	0
4	0	1	1	3	2	0	0	3	0
5	0	1	3	3	2	0	0	3	0
6	0	3	3	3	2	0	0	3	0
7	0	0	0	0	0	0	0	0	0

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0	0	-1.5	-1.5				
1							
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7 + padding

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	3	0
1	0	1	0	1	1	1	1	0	0
2	0	1	1	3	3	0	0	0	0
3	0	1	1	3	2	0	0	3	0
4	0	1	1	3	2	0	0	3	0
5	0	1	3	3	2	0	0	3	0
6	0	3	3	3	2	0	0	3	0
7	0	0	0	0	0	0	0	0	0

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0	0	-1.5	-1.5	-1.5			
1							
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7 + padding

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	3	0
1	0	1	0	1	1	1	1	0	0
2	0	1	1	3	3	0	0	0	0
3	0	1	1	3	2	0	0	3	0
4	0	1	1	3	2	0	0	3	0
5	0	1	3	3	2	0	0	3	0
6	0	3	3	3	2	0	0	3	0
7	0	0	0	0	0	0	0	0	0

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0	0	-1.5	-1.5	-1.5	-1.5	-3	-3
1							
2							
3							
4							
5							
6							

Convolução

Volume de entrada 7 x 7 + padding

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	3	0
1	0	1	0	1	1	1	1	0	0
2	0	1	1	3	3	0	0	0	0
3	0	1	1	3	2	0	0	3	0
4	0	1	1	3	2	0	0	3	0
5	0	1	3	3	2	0	0	3	0
6	0	3	3	3	2	0	0	3	0
7	0	0	0	0	0	0	0	0	0

Filtro W (3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

Volume de saída

	0	1	2	3	4	5	6
0	0	-1.5	-1.5	-1.5	-1.5	-3	-3
1	3.5	1.5	2.5	1	1.5	0.5	-2.5
2	1	0.5	1.5	-2.5	-2.5	1	-1
3	2	3	3.5	-4.5	-5	1.5	0
4	3	3	2.5	-5	-4	4.5	1.5
5	3	3	0.5	-5	-4	4.5	1.5
6	3.5	0.5	-2.5	-5	-4	3	1.5

Convolução

- **Convolução em profundidade**: quando a entrada possui mais do que 1 canal

Convolução

- **Convolução em profundidade**: quando a entrada possui mais do que 1 canal
  - O filtro terá $k \times k \times p$, onde p é a quantidade de canais de entrada

Convolução

Volume de entrada 6 x 6 x 3 (RGB) + zero padding

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	0
1	0	1	0	1	1	1	1	0
2	0	1	1	3	3	0	0	0
3	0	1	1	3	2	0	0	0
4	0	1	1	3	2	0	0	0
5	0	1	3	3	2	0	0	0
7	0	0	0	0	0	0	0	0

Filtro W (3 x 3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

1	0	1	
-1	1	0	
0	0	-0.5	

	1	0	1
	1	0.5	-1
	0	1	0

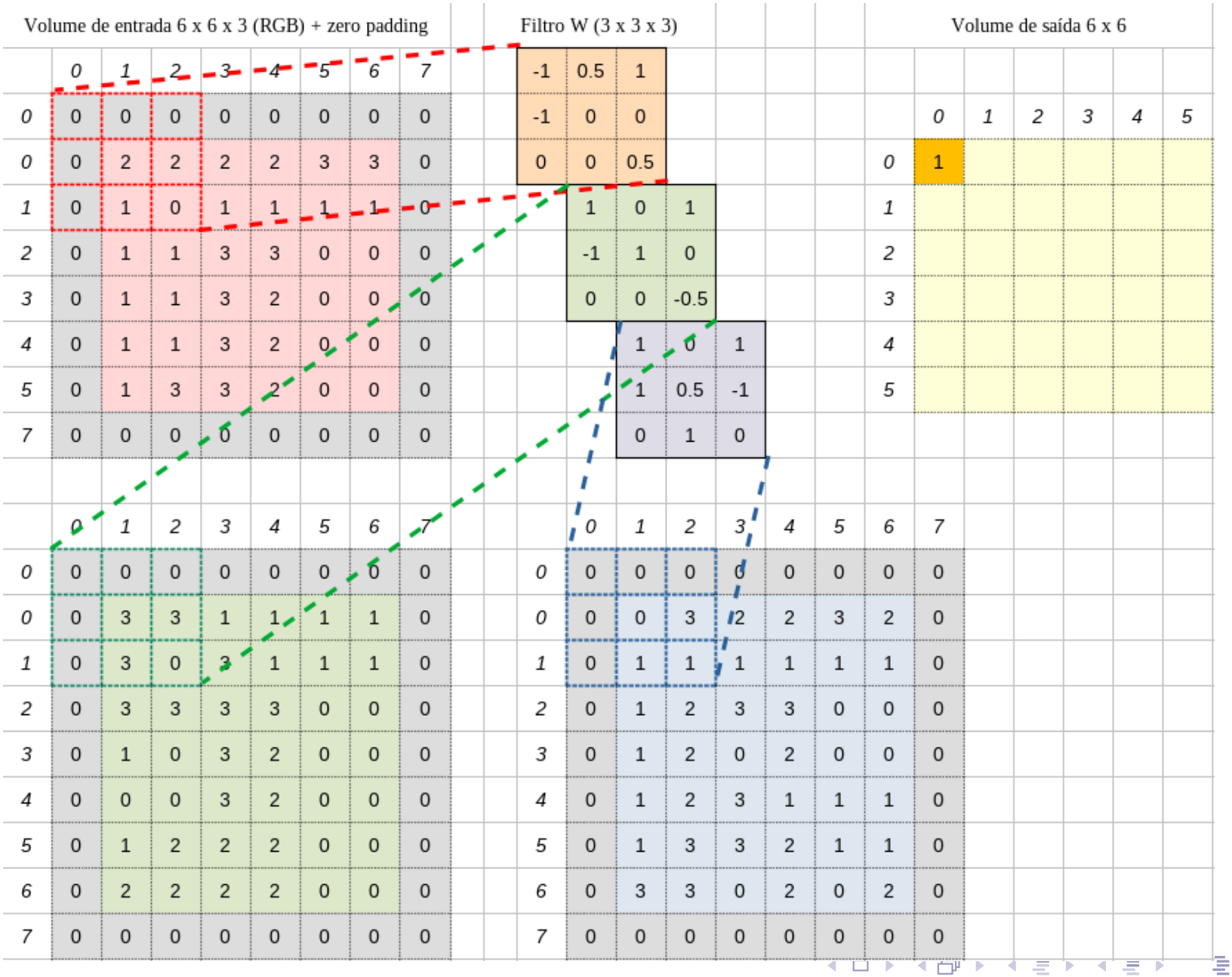
Volume de saída 6 x 6

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	3	3	1	1	1	1	0
1	0	3	0	3	1	1	1	0
2	0	3	3	3	3	0	0	0
3	0	1	0	3	2	0	0	0
4	0	0	0	3	2	0	0	0
5	0	1	2	2	2	0	0	0
6	0	2	2	2	2	0	0	0
7	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	0	3	2	2	3	2	0
1	0	1	1	1	1	1	1	0
2	0	1	2	3	3	0	0	0
3	0	1	2	0	2	0	0	0
4	0	1	2	3	1	1	1	0
5	0	1	3	3	2	1	1	0
6	0	3	3	0	2	0	2	0
7	0	0	0	0	0	0	0	0

Convolução


Volume de entrada 6 x 6 x 3 (RGB) + zero padding
Filtro W (3 x 3 x 3)
Volume de saída 6 x 6

Convolução

Volume de entrada 6 x 6 x 3 (RGB) + zero padding

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	0
1	0	1	0	1	1	1	1	0
2	0	1	1	3	3	0	0	0
3	0	1	1	3	2	0	0	0
4	0	1	1	3	2	0	0	0
5	0	1	3	3	2	0	0	0
7	0	0	0	0	0	0	0	0

Filtro W (3 x 3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

1	0	1
-1	1	0
0	0	-0.5

1	0	1
1	0.5	-1
0	1	0

Volume de saída 6 x 6

	0	1	2	3	4	5
0	1	-2.5				
1						
2						
3						
4						
5						

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	3	3	1	1	1	1	0
1	0	3	0	3	1	1	1	0
2	0	3	3	3	3	0	0	0
3	0	1	0	3	2	0	0	0
4	0	0	0	3	2	0	0	0
5	0	1	2	2	2	0	0	0
6	0	2	2	2	2	0	0	0
7	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	0	3	2	2	3	2	0
1	0	1	1	1	1	1	1	0
2	0	1	2	3	3	0	0	0
3	0	1	2	0	2	0	0	0
4	0	1	2	3	1	1	1	0
5	0	1	3	3	2	1	1	0
6	0	3	3	0	2	0	2	0
7	0	0	0	0	0	0	0	0

Convolução

Volume de entrada 6 x 6 x 3 (RGB) + zero padding

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	3	3	0
1	0	1	0	1	1	1	1	0
2	0	1	1	3	3	0	0	0
3	0	1	1	3	2	0	0	0
4	0	1	1	3	2	0	0	-2
5	0	1	3	3	2	0	0	-1
7	0	0	0	0	0	0	0	0

Filtro W (3 x 3 x 3)

-1	0.5	1
-1	0	0
0	0	0.5

1	0	1
-1	1	0
0	0	-0.5

1	0	1
1	0.5	-1
0	1	0

Volume de saída 6 x 6

	0	1	2	3	4	5
0	1	-2.5	-1	-1	0.5	2
1	11.5	5.5	16.5	9.5	8	3
2	4.5	8	4	10	1.5	1.5
3	7.5	14.5	19.5	2.5	2	1
4	3.5	9	16	1.5	-0.5	2.5
5	4	11	6	7	-0.5	4.5

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	3	3	1	1	1	1	0
1	0	3	0	3	1	1	1	0
2	0	3	3	3	3	0	0	0
3	0	1	0	3	2	0	0	0
4	0	0	0	3	2	0	0	0
5	0	1	2	2	2	0	0	0
6	0	2	2	2	2	0	0	0
7	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
0	0	0	3	2	2	3	2	0
1	0	1	1	1	1	1	1	0
2	0	1	2	3	3	0	0	0
3	0	1	2	0	2	0	0	0
4	0	1	2	3	1	1	1	0
5	0	1	3	3	2	1	1	0
6	0	3	3	0	2	0	2	0
7	0	0	0	0	0	0	0	0

Camada convolucional

Entrada ($m \times n \times p$)  e.g. $32 \times 32 \times 3$

Filtro (kernel ou neurônio convolutional) w com tamanho $k \times k \times p$, e.g. $5 \times 5 \times 3$

- Cada neurônio realiza a convolução da entrada e gera um volume (matriz/tensor) de saída

Centrado em um pixel específico, temos, matematicamente

$$\mathrm{w}^t \mathrm{x} + b$$

- sim, há a soma de **bias** para além dos pesos da convolução.

Camada convolucional

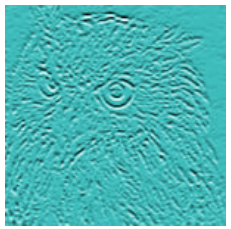
- Mapas de ativação (ou características) são obtidos após convolução e função de ativação (e.g. ReLU);
- Empilhados formam um tensor que será a entrada da próxima camada.

$(32 \times 32 \times 3)$

camada convolucional

10 filtros $5 \times 5 \times 3$

$\rightarrow$



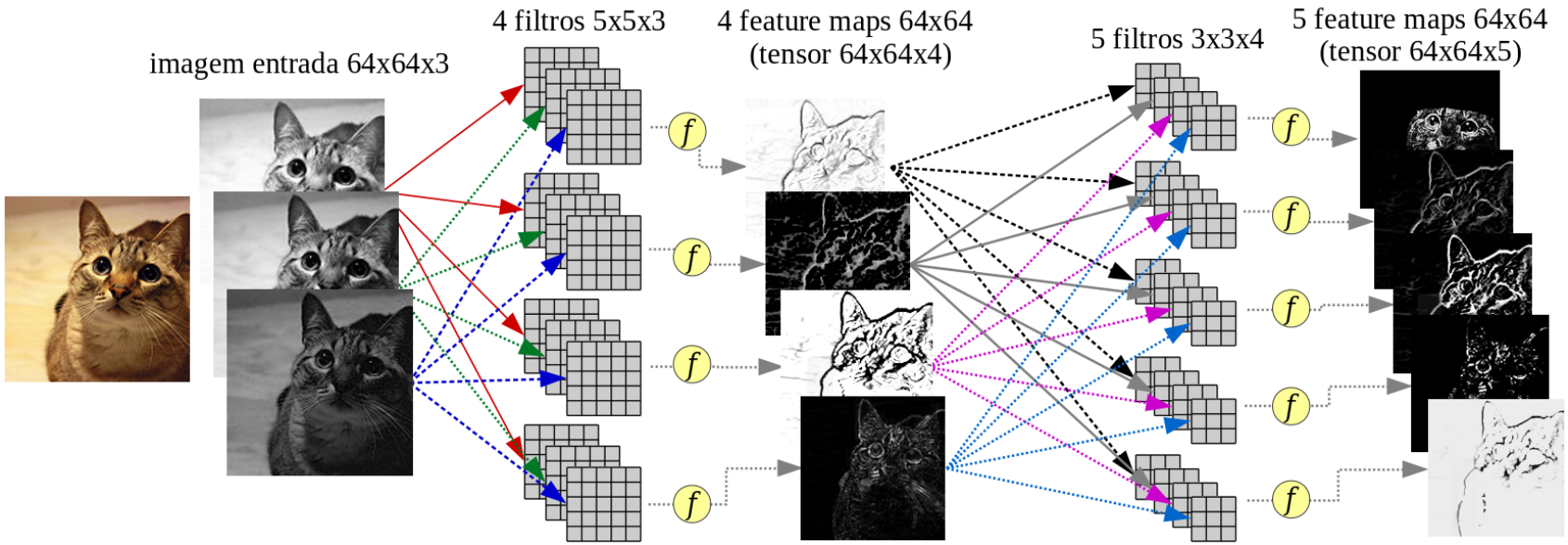
01

02

...

10

Camada convolucional: feature maps



Camada convolucional: entrada, filtro, passo

A camada convolucional tem que levar em conta:

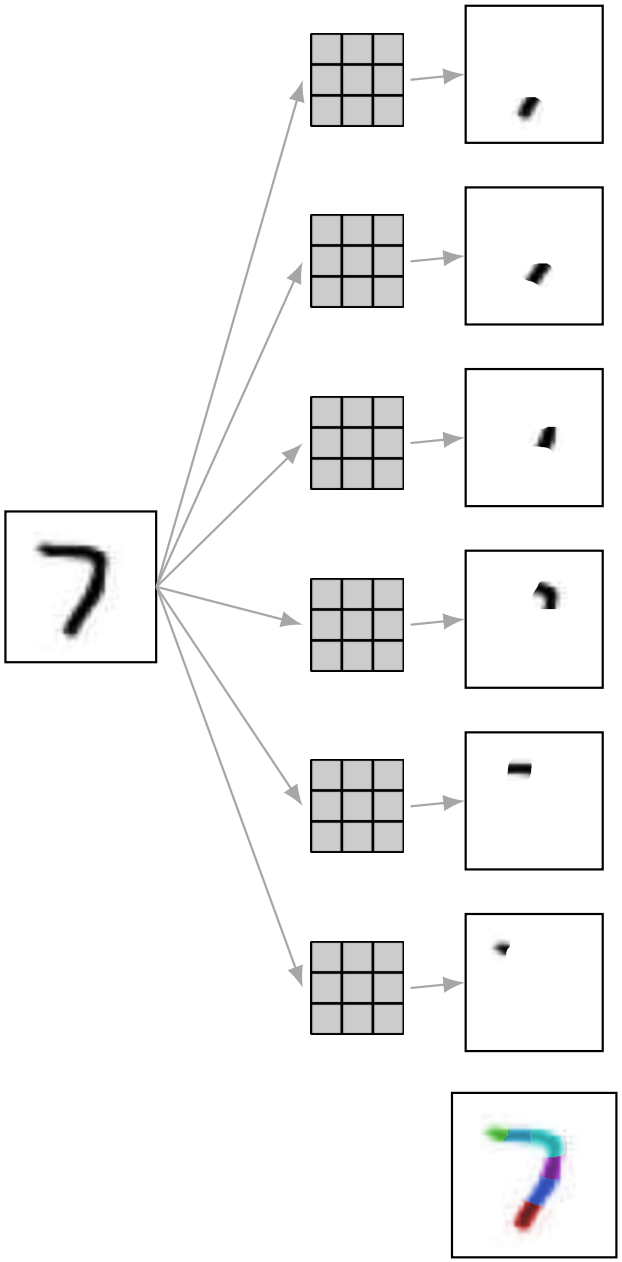
- tamanho da entrada (largura, altura, profundidade)
- tamanho do filtro
  - a profundidade deve ser igual à da entrada
  - altura e largura afetam o *campo receptivo local*

Camada convolucional: entrada, filtro, passo

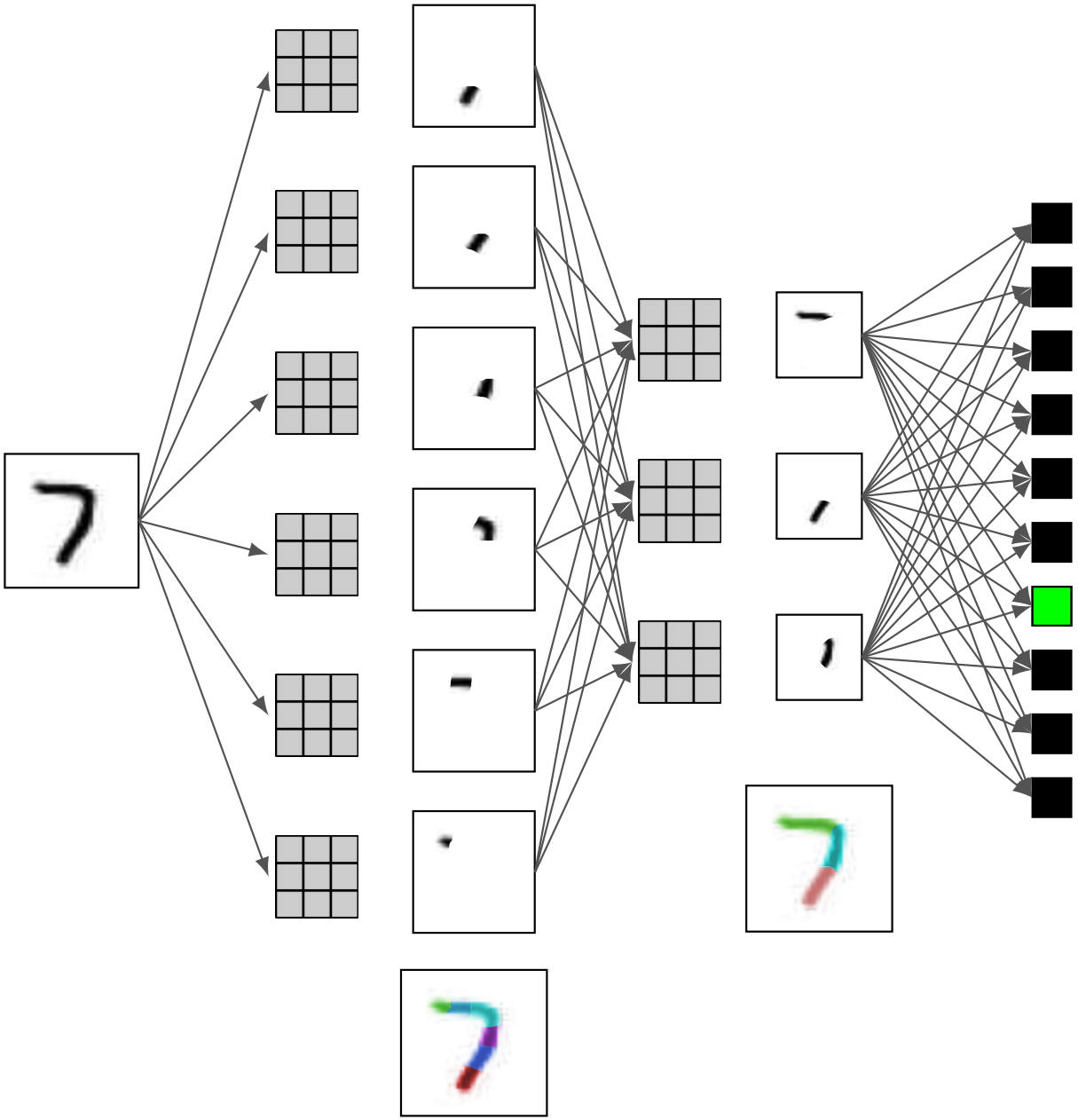
A camada convolucional tem que levar em conta:

- tamanho da entrada (largura, altura, profundidade)
- tamanho do filtro
  - a profundidade deve ser igual à da entrada
  - altura e largura afetam o *campo receptivo local*
- stride (passo)
  - 1 : todos os pixels são filtrados pelo neurônio
  - > 1 : salta um número de pixels em determinada direção, a cada convolução.
    - nesse caso o volume de saída tem tamanho reduzido, ex. com passo 2

Classificação de dígitos com conv.layers



Classificação de dígitos com conv.layers

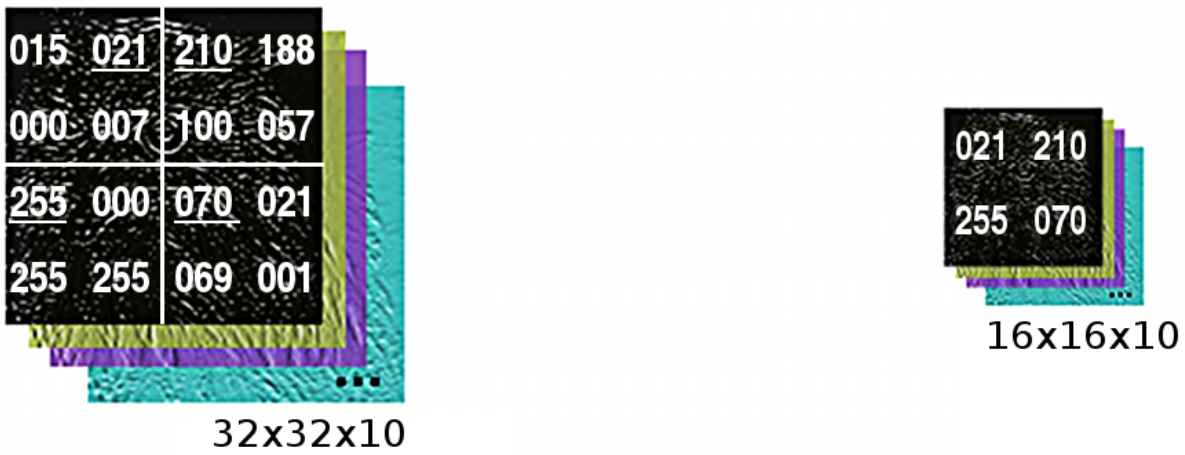


Subamostragem: Pooling layer

Opera sobre cada mapa de ativação, reduzindo a dimensão lateral

- max pooling: aplica a operação de máximo local
- average pooling: aplica operação de média local

Ex.: max pooling com tamanho de pool 2 e passo 2.



Usar camadas convolucionais com passo/stride > 1 pode substituir pooling

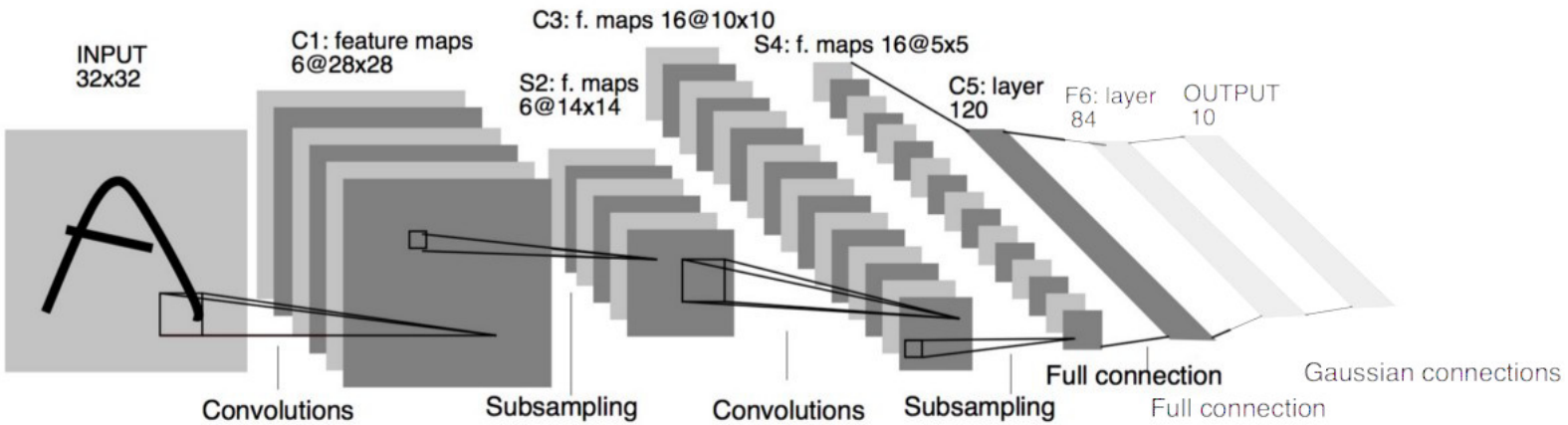
Pooling layer

Reduzir o tamanho da entrada permite que o filtro opere em regiões maiores da imagem.

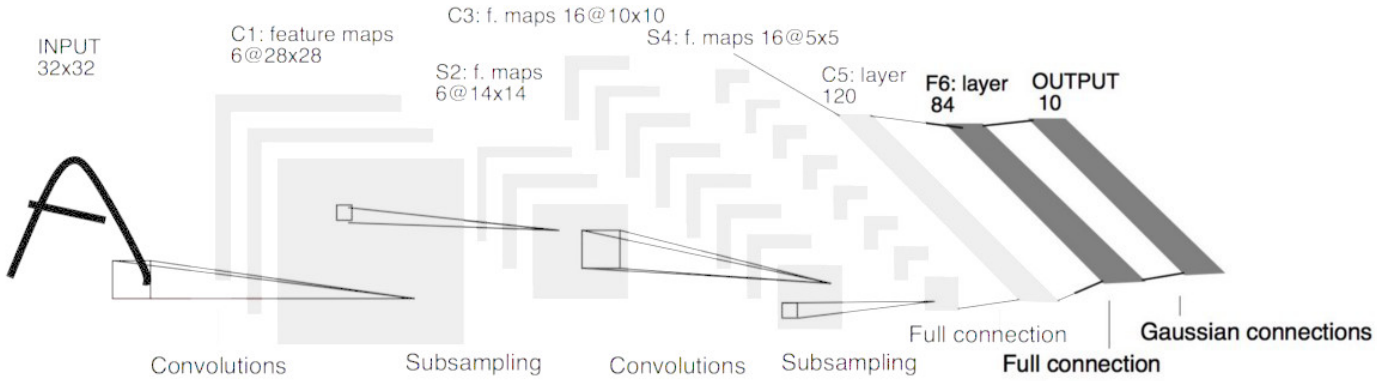
Empilhamento de camadas convolucionais aumenta o campo receptivo local não necessitando manter a resolução de entrada

(uso de filtro de mesmo tamanho em imagens progressivamente menores)

Voltando à arquitetura



Camadas densas e saída



Dense/fully connected (FC) layer:

- similar à de uma MLP
- pode ser vista como uma **projeção** dos dados em uma dimensionalidade arbitrária

Saída: comumente densa (ex: classificação e regressão)

- pode ser vista como um vetor de distribuição de probabilidades
- não é densa em redes completamente convolucionais (Fully Convolutional Networks)

Agenda

Machine vs Deep Learning

Dados estruturados e independentes: Perceptron

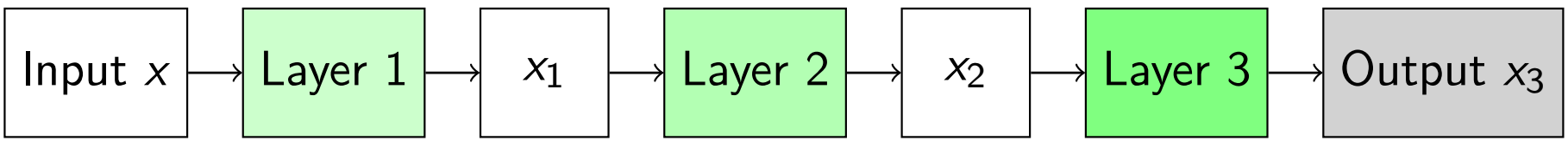
Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

Convergência e aprendizado

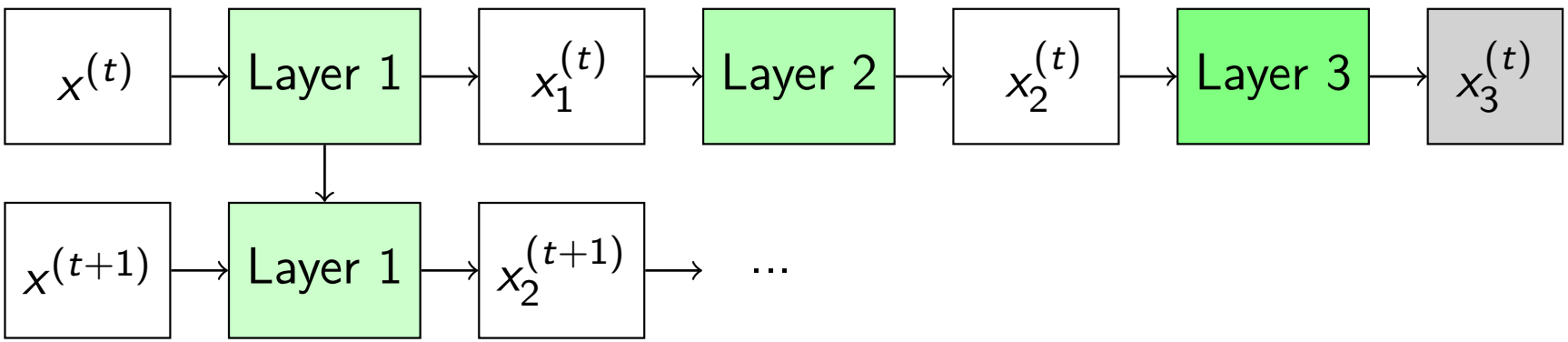
Para dados não sequenciais

- Camadas densas e convolucionais consideram apenas o exemplo atual para computar a saída
- Em cada iteração, cada entrada vai passando pelas camadas até atingir a saída



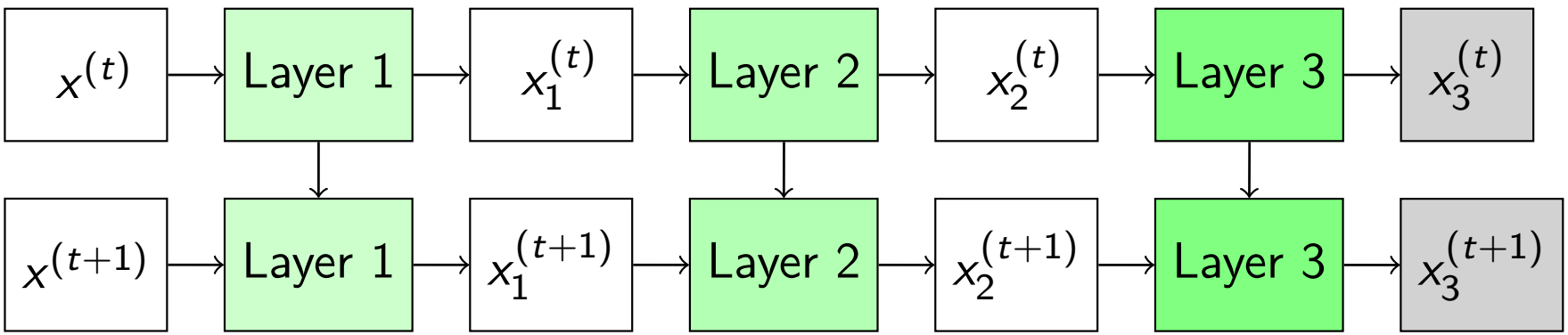
Para dados sequenciais

- Se a iteração $t+1$ depende da anterior t, usamos a saída de cada camada para alimentar a camada na entrada da iteração $t+1$



Para dados sequenciais

- Dessa forma, a saída (após a primeira), dependerá não apenas da entrada atual, mas das saídas computadas anteriormente para cada unidade

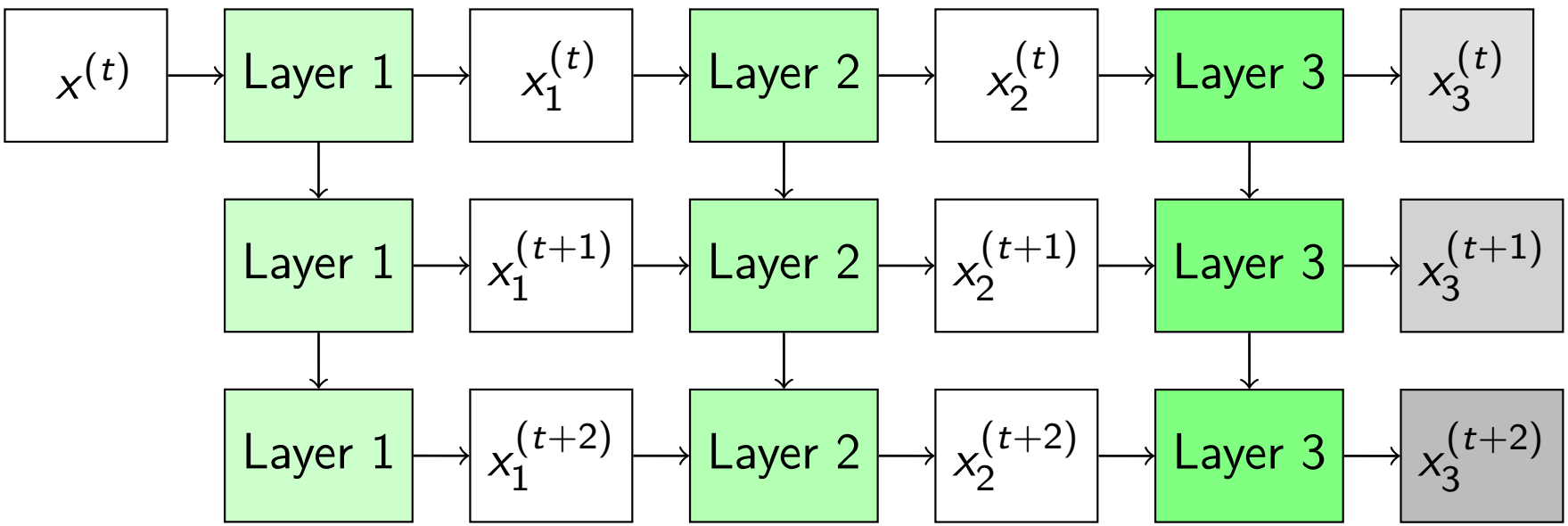


Configurações de sequências

- Uma entrada, saída sequencial
- Entrada sequencial, uma saída
- Entrada sequencial, saída sequencial

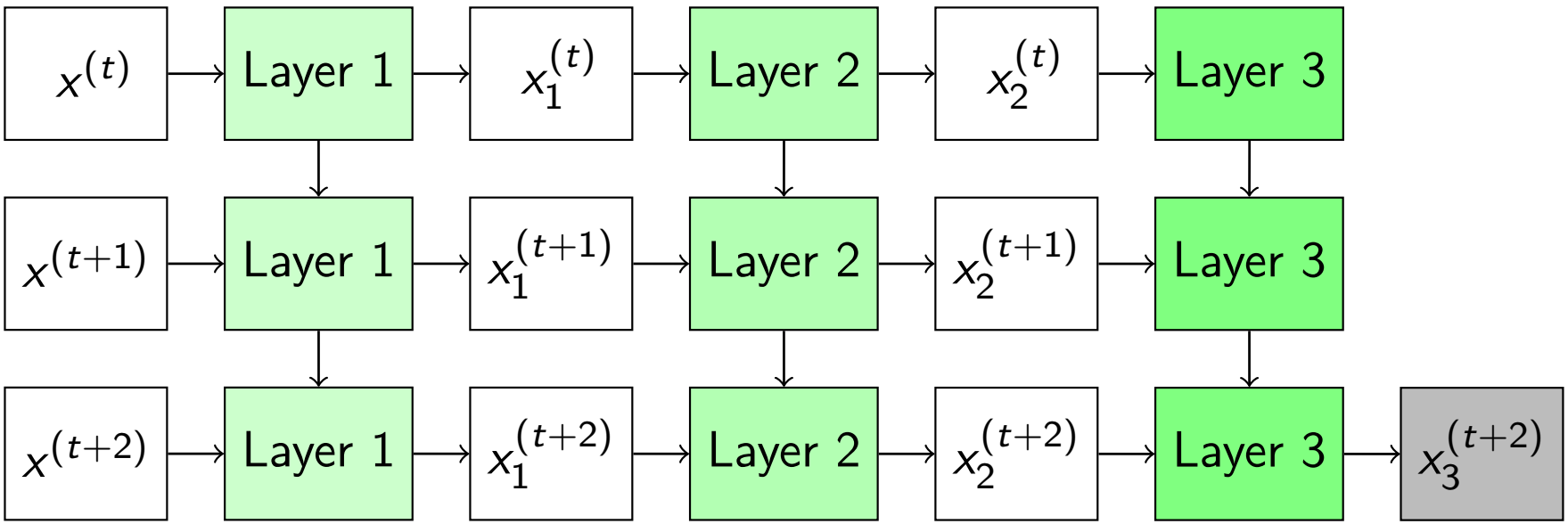
Configurações de sequências

- Uma entrada, saída sequencial: e.g. um áudio ou imagem é dado como entrada e a rede produz uma sequência de palavras que os descrevem



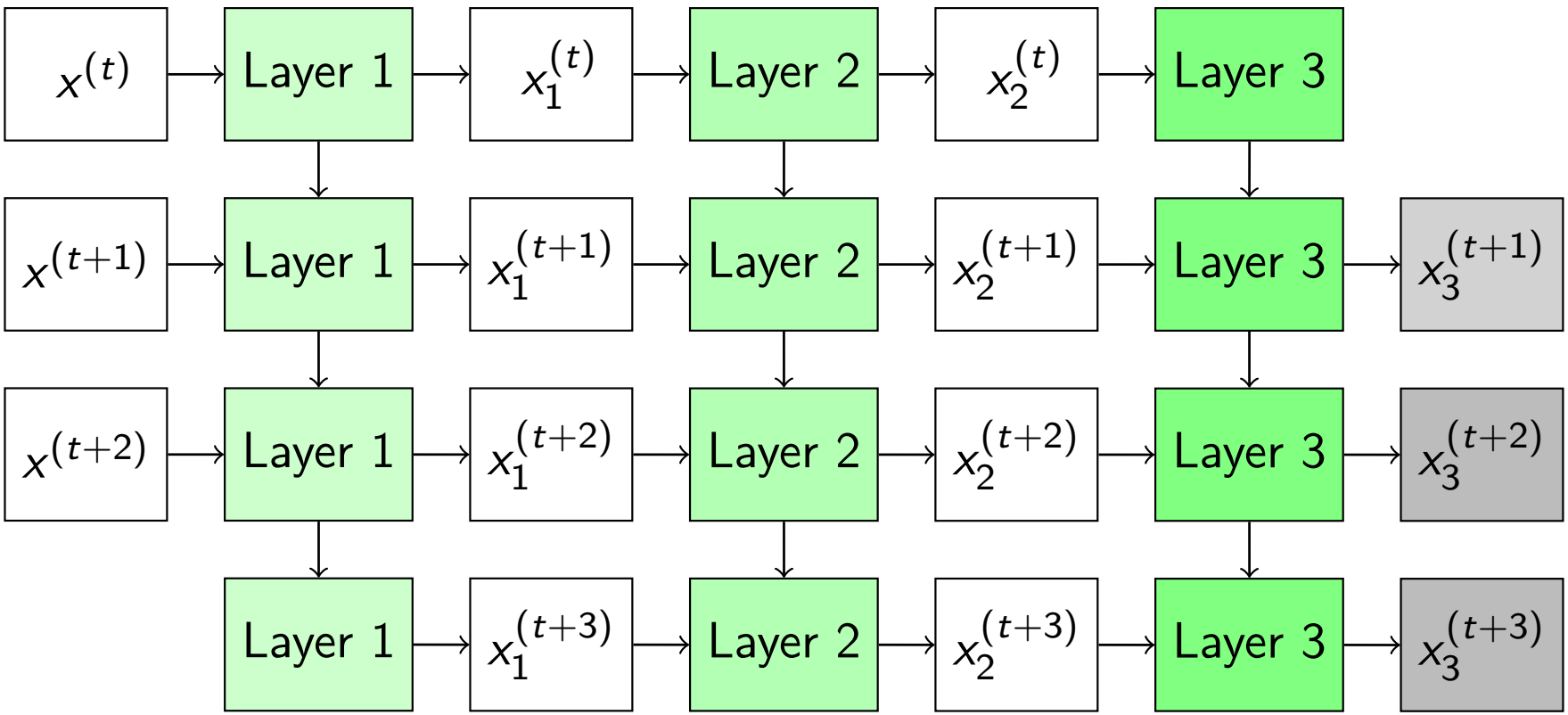
Configurações de sequências

- Entrada sequencial, uma saída e.g. um texto é dado como entrada e a saída é sua análise de sentimentos: conteúdo positivo ou negativo.



Configurações de sequências

- Entrada sequencial, saída sequencial e.g. tradução automática de sentenças entre diferentes linguagens (que pode ou não ter um atraso)



Agenda

Machine vs Deep Learning

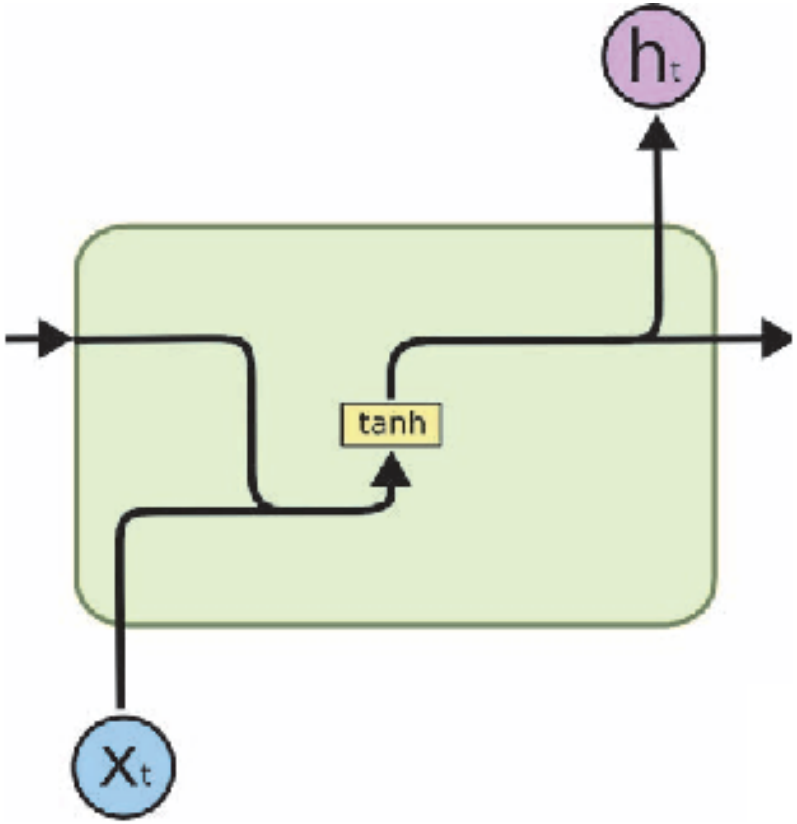
Dados estruturados e independentes: Perceptron

Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

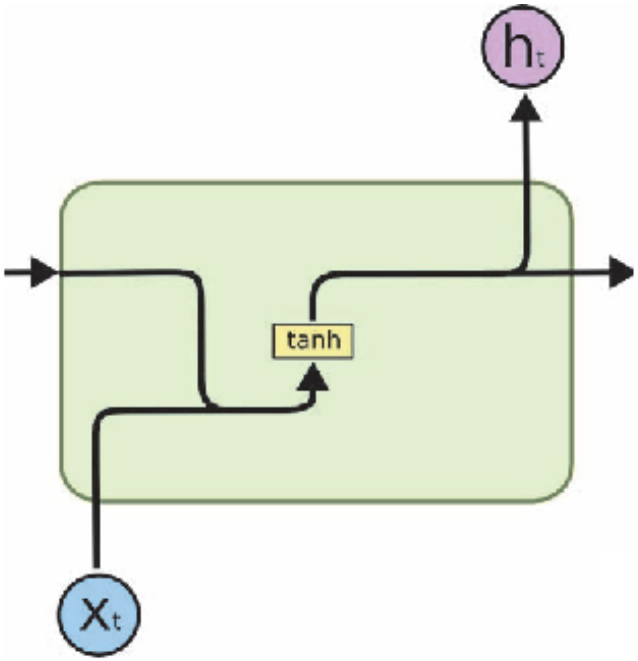
Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

Convergência e aprendizado

Aprende um tipo de "memória"



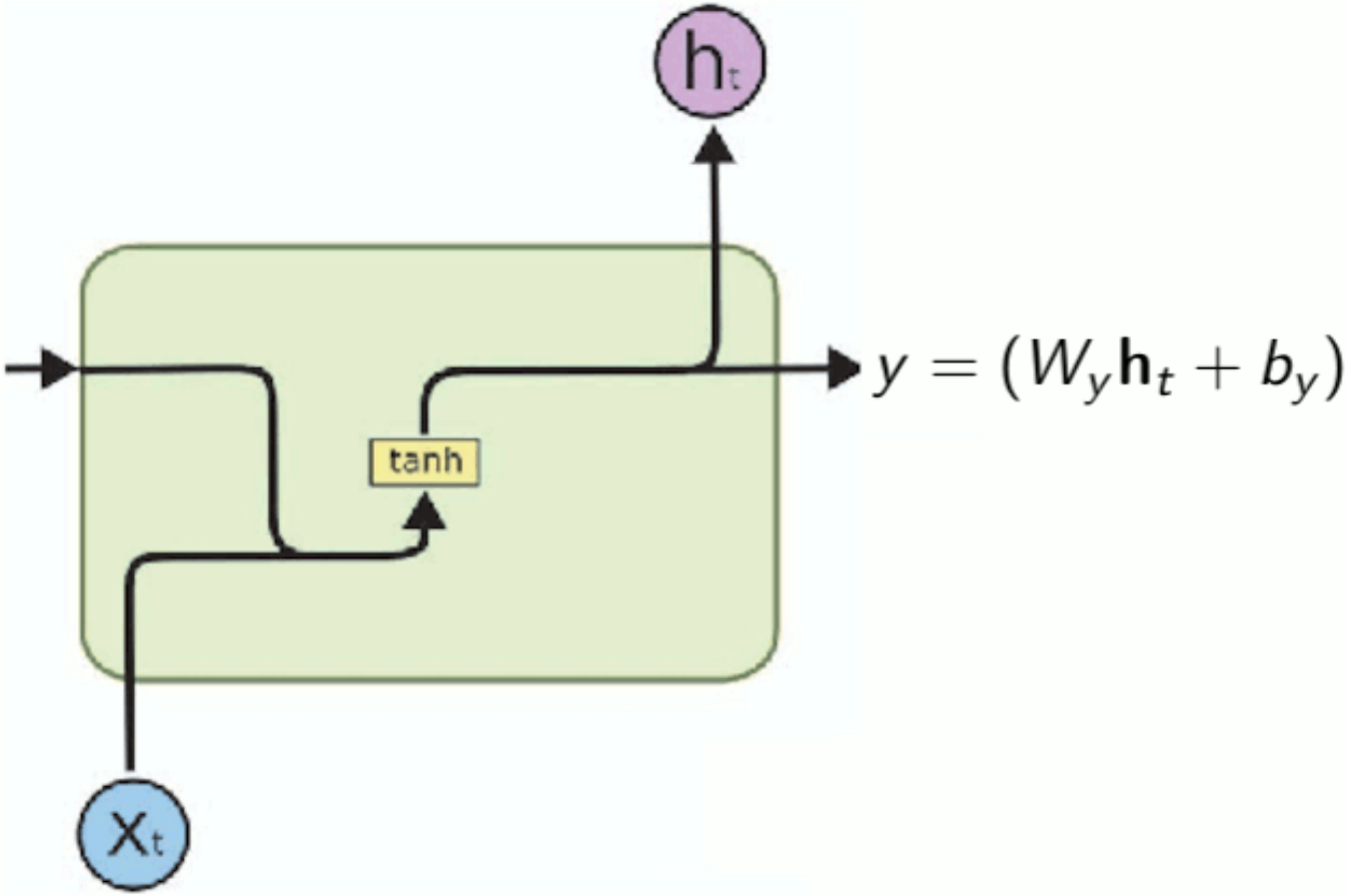
Componentes são combinações lineares



$$h_t = \tanh\left(W_h h_{t-1} + W_x x_t + b_h\right)$$
$$y = \left(W_y h_t + b_y\right)$$

Saída recorrente (sumário) e saída da rede

$$\mathbf{h}_t = \tanh\left(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + b_h\right)$$



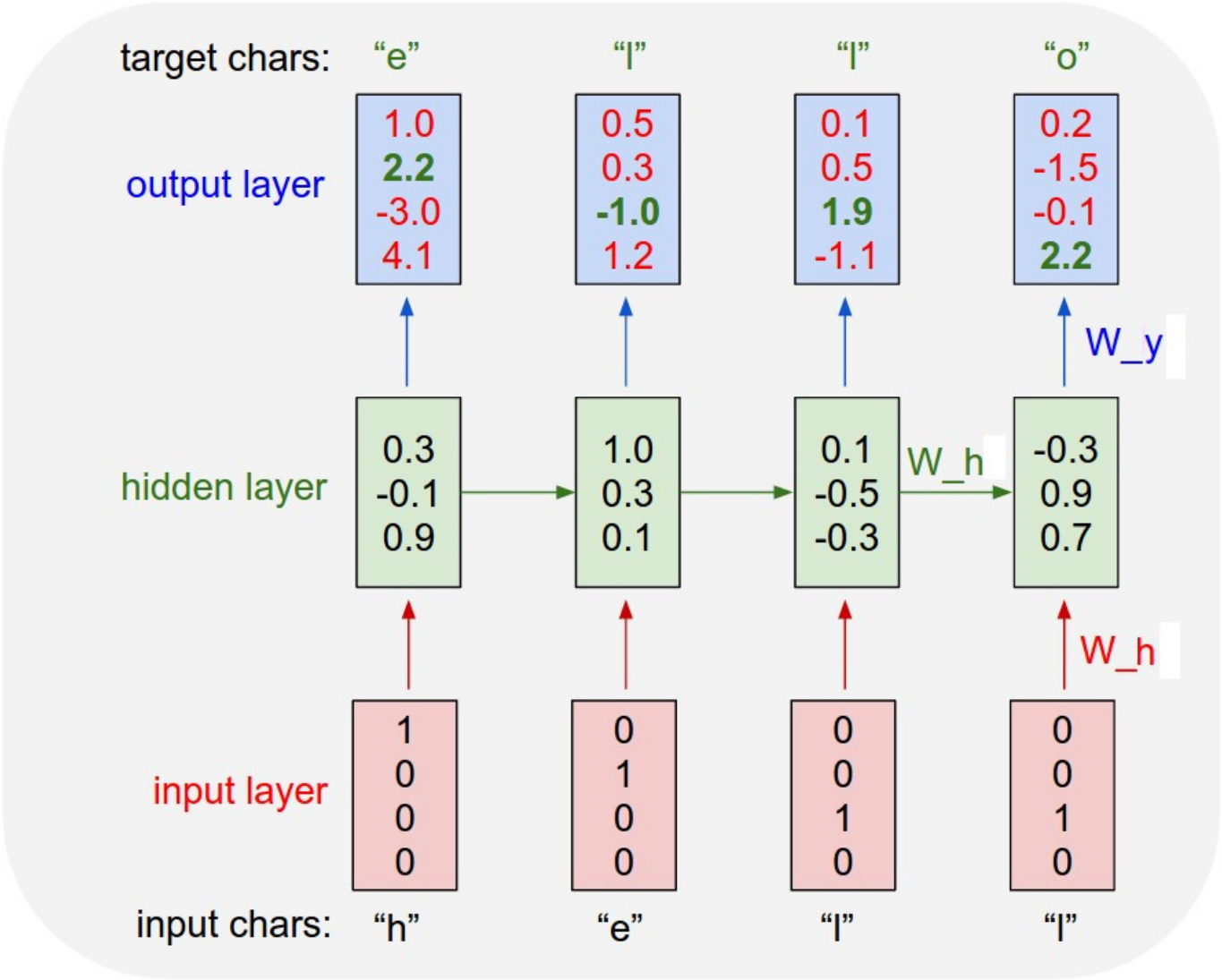
Exemplo: predizer próximo caracter

Definimos uma codificação one-hot para os caracteres:

- `h` $= [1, 0, 0, 0]$
- `e` $= [0, 1, 0, 0]$
- `l` $= [0, 0, 1, 0]$
- `o` $= [0, 0, 0, 1]$

`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

Exemplo: predizer próximo caracter



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Agenda

Machine vs Deep Learning

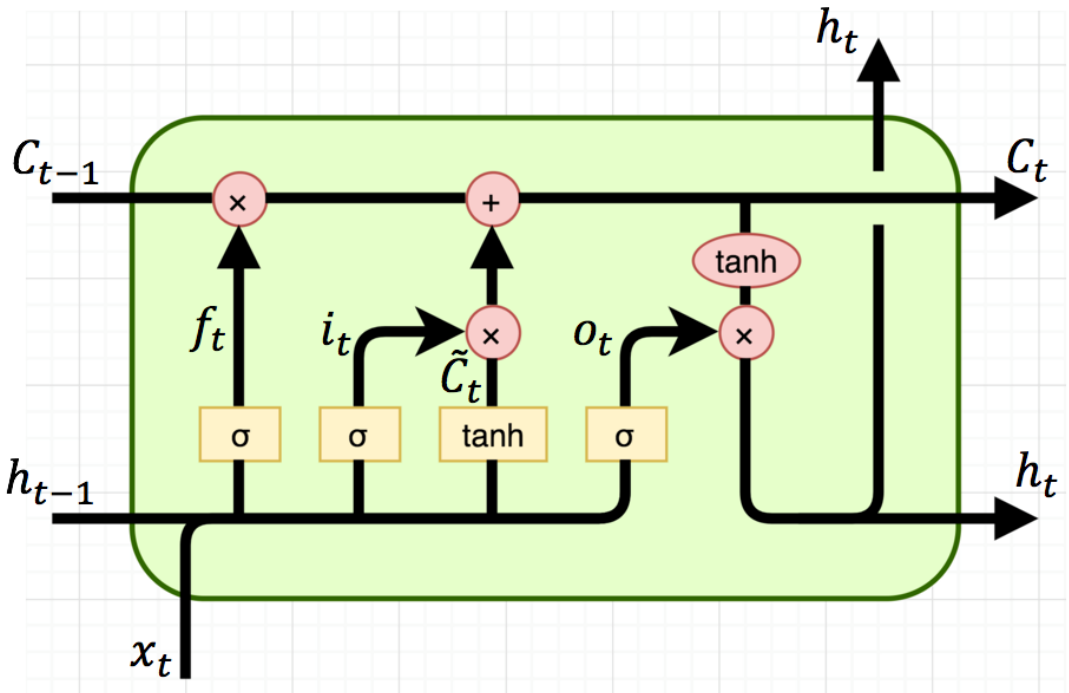
Dados estruturados e independentes: Perceptron

Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

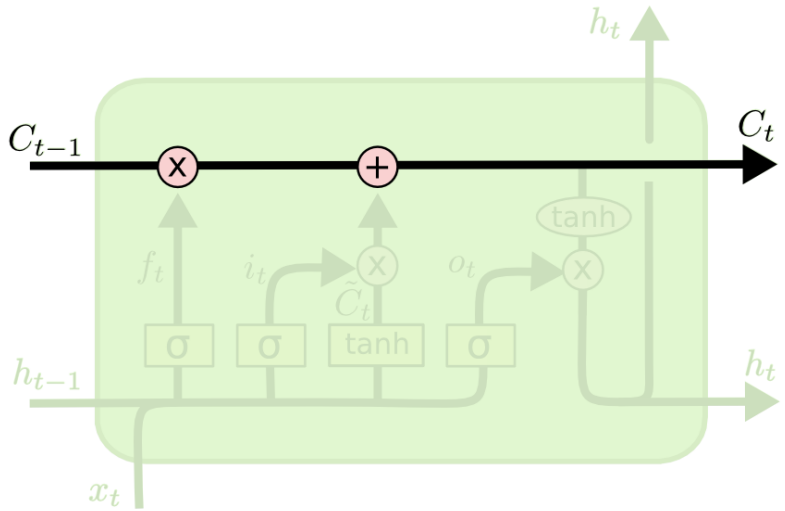
Convergência e aprendizado

Long Short Term Memory Unit (LSTM)



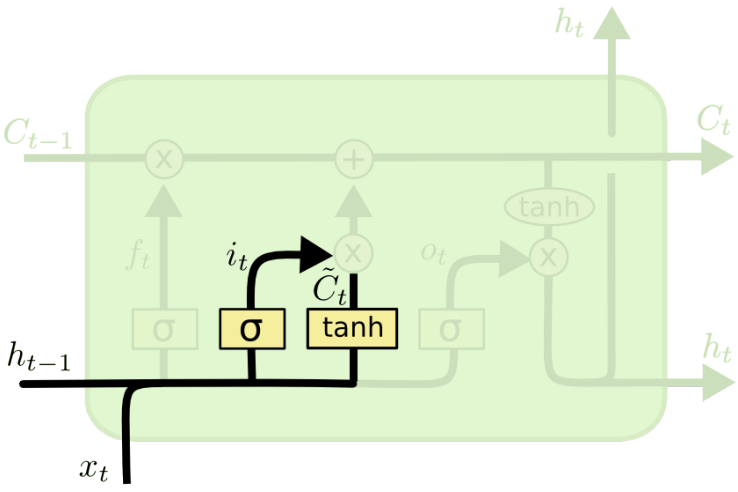
Adaptados de `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

LSTM network: Cell state



- Responsável pela memória longa da unidade, adiciona contribuições para além da iteração anterior.
- Esse estado pode ser modificado por 3 portões/**gates**

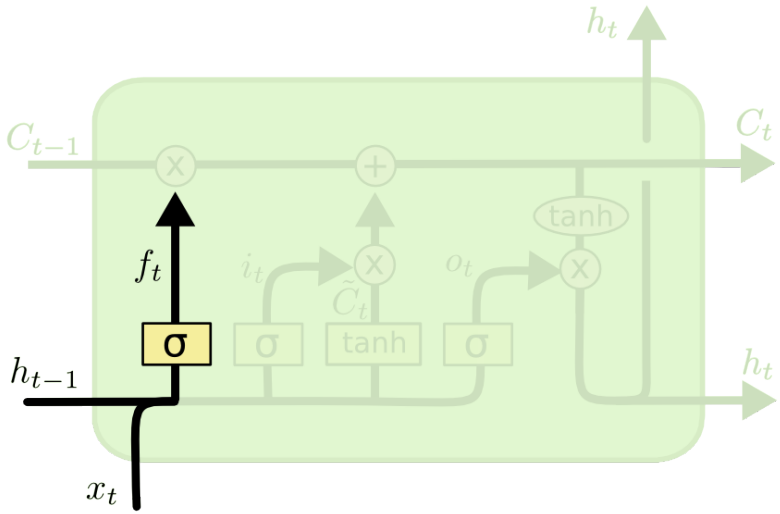
LSTM network: input / update gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- primeiro, combina o sumário anterior h_{t-1} e a entrada x_t
- então, aprende um filtro $\tilde{C}_t$ que indica quais partes devem ser mantidas na "memória longa", sendo somado a C_{t-1}

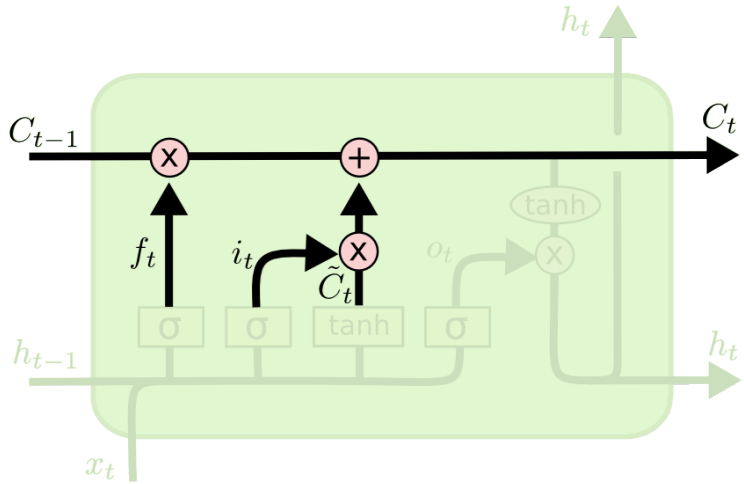
LSTM network: forget/reset gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- decide o que cancelar de C com base no sumário anterior e a entrada atual
- saída entre 0 (esquecer) e 1 (manter totalmente) para cada dimensão de C

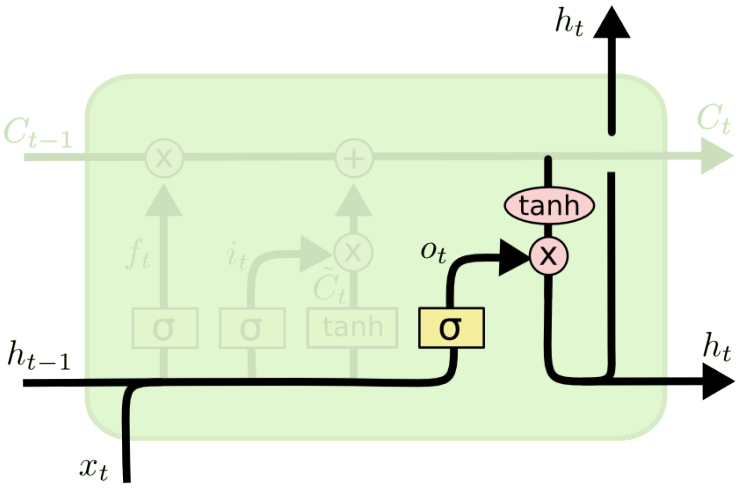
LSTM network: update Cell state



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- agora temos uma combinação entre os estados atual e anterior

LSTM network: output gate



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

- decide qual será o sumário, o qual é transformado a partir do anterior
- e ponderado de acordo com o estado de célula atual, C_t

Agenda

Machine vs Deep Learning

Dados estruturados e independentes: Perceptron

Dados espaciais: convolução
- Convolução
- Camada convolucional para redes neurais
- Pooling

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)

Convergência e aprendizado

Algumas suposições que fizemos

Dados de treinamento

- Limpos
- Representativos e bem definidos com relação à tarefa: classes, valores da regressão, etc.
- Baixa taxa de erros de rótulo
- Quantidade de dados é suficiente

Algumas suposições que fizemos

Dados de treinamento

- Limpos
- Representativos e bem definidos com relação à tarefa: classes, valores da regressão, etc.
- Baixa taxa de erros de rótulo
- Quantidade de dados é suficiente

- E se não for possível?

Algumas suposições que fizemos

Dados de treinamento

- Limpos
- Representativos e bem definidos com relação à tarefa: classes, valores da regressão, etc.
- Baixa taxa de erros de rótulo
- Quantidade de dados é suficiente

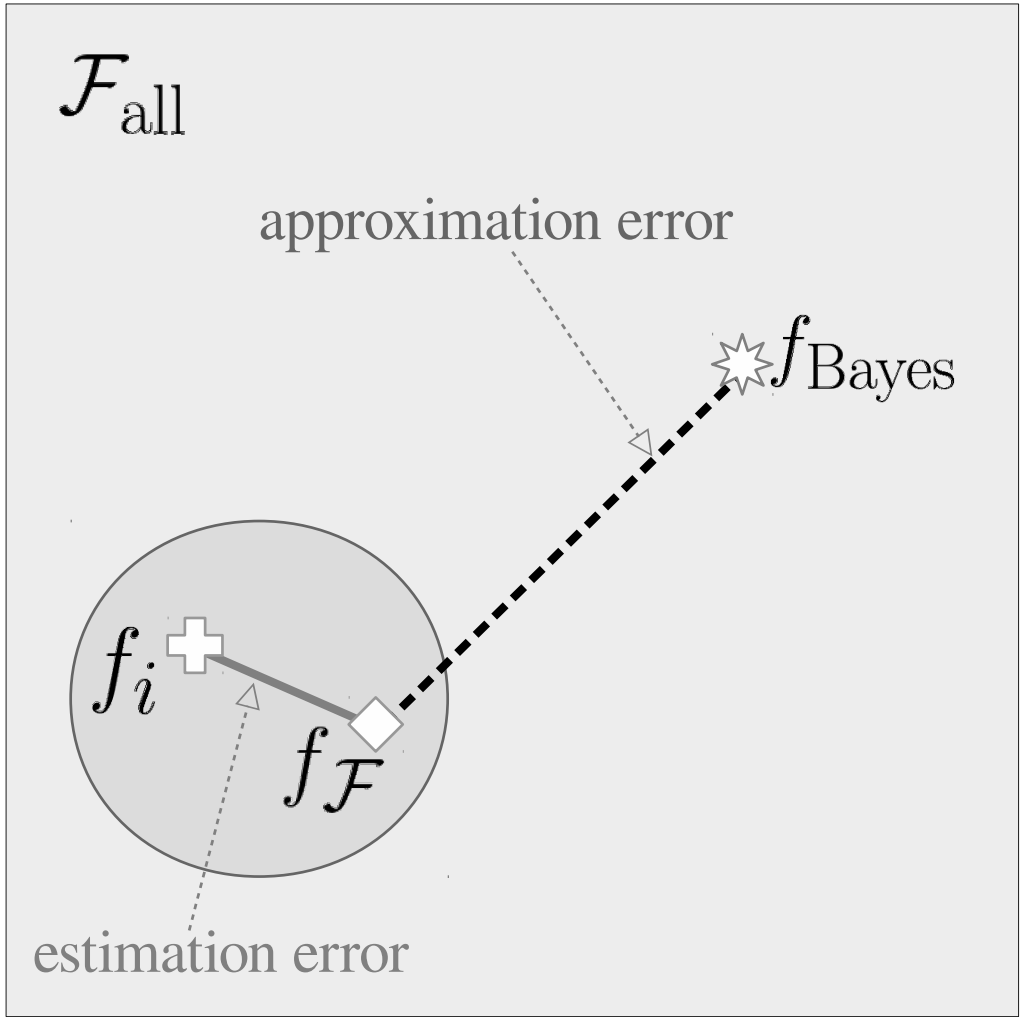
- E se não for possível?
  - Riscos: *overfitting*, baixa generalização, maior dificuldade no treinamento.

Complexidade de modelos: "viés" segundo a Teoria do Aprendizado Estatístico

- Lembrando: Aprendizado de Máquina pode ser formulado como sendo aprender os parâmetros de $f : X \rightarrow Y$
- Um algoritmo ajusta f a partir de um espaço de funções admissíveis $\mathcal{F}$:
  - "muitas" funções: mais graus de liberdade, menor garantia de convergência, possível *overfitting*;
  - "poucas" funções: menos graus de liberdade, maior garantia de convergência, possível *underfitting*.

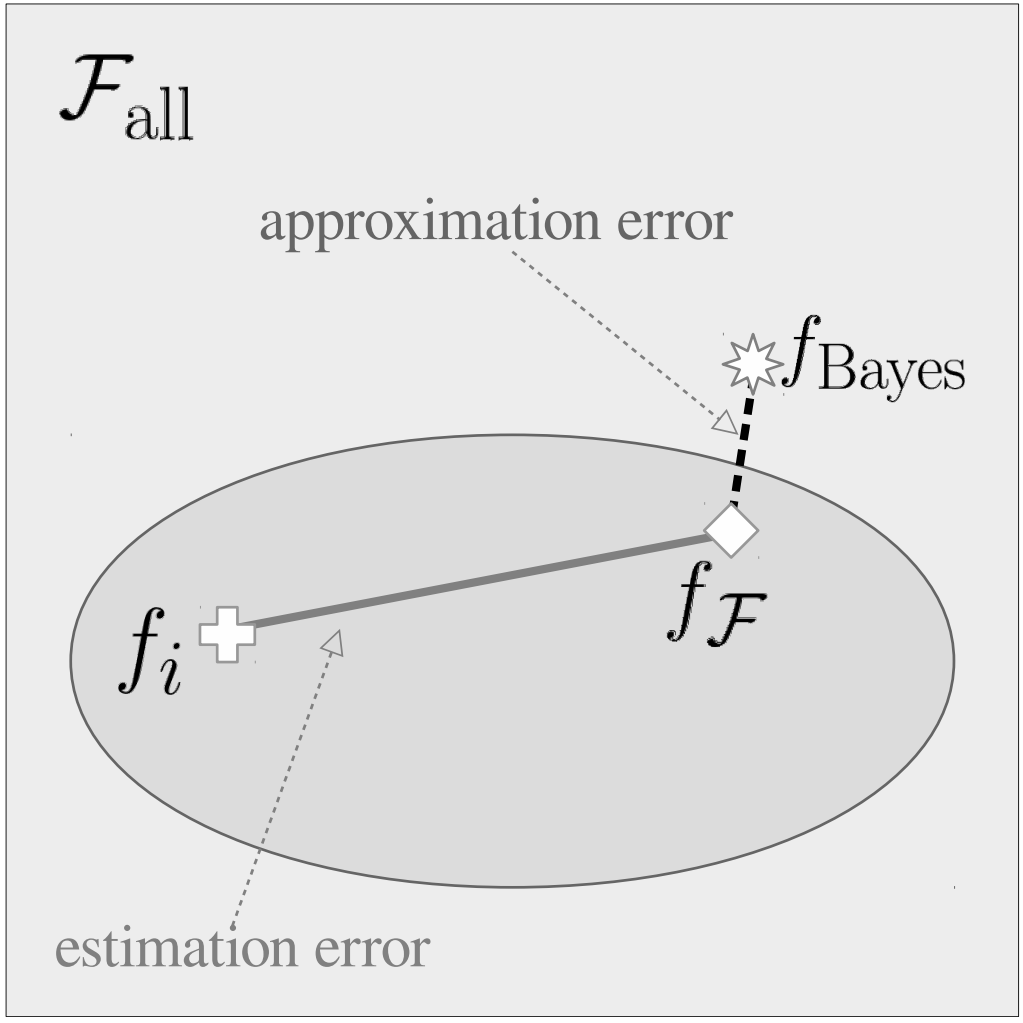
Erros quando definindo o espaço de funções admissíveis

Viés forte: espaço de funções restrito



Erros quando definindo o espaço de funções admissíveis

Viés fraco: espaço de funções amplo



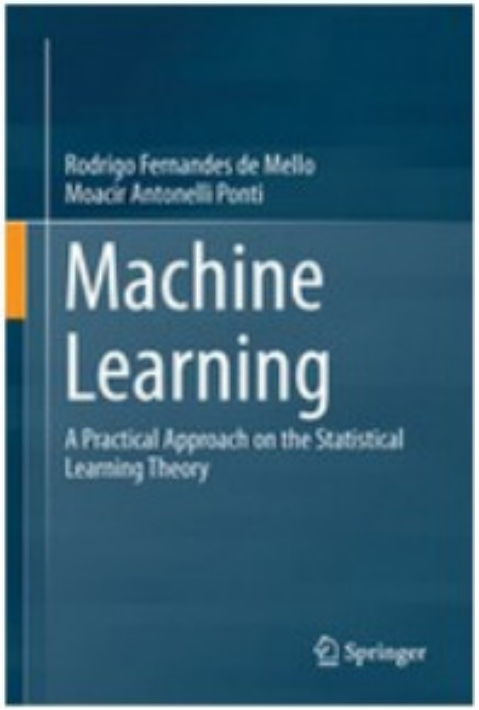
Quando o assunto é volume de dados

Nem sempre...

- é possível coletar mais
- aumentação é efetiva

Bibliografia I

Rodrigo Mello, Moacir A. Ponti. **Machine Learning: a practical approach on the statistical learning theory** Springer, 2018.



Bibliografia II

Moacir A. Ponti, Gabriel Paranhos da Costa. **Como funciona o Deep Learning**
SBC, 2017. Book chapter.
`https://arxiv.org/abs/1806.07908`

Moacir A. Ponti, Leo Ribeiro, Tiago Nazaré, Tu Bui, John Collomosse. **Everything You Wanted to Know About Deep Learning for Computer Vision but were Afraid to Ask**.
SIBGRAPI-T, 2017. Tutorial.