



Introdução ao aprendizado de máquina 6

Aula 6- Introdução a deep learning

- Dificuldades de se treinar redes neurais profundas
- Gradientes desaparecendo ou explodindo
- Técnicas para se estimar redes neurais profundas
- Redes Neurais convolucionais (CNN)
- Interpretação das convoluções
- Outras aplicações de CNN



Dificuldades de se treinar redes neurais profundas

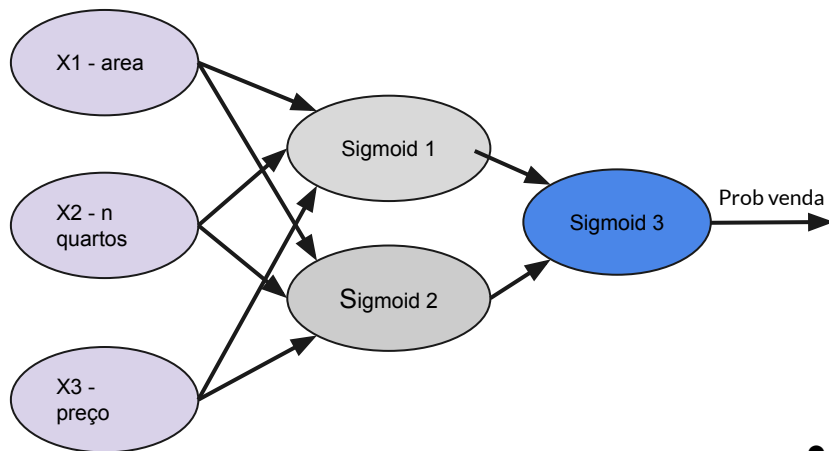
Relembrando redes neurais



Camada de entrada

Camada escondida

Camada de saída



$$a_1 = \Omega (\theta_{10} + \theta_{11} x_1 + \theta_{12} x_2 + \theta_{13} x_3)$$

$$a_2 = \Omega (\theta_{20} + \theta_{21} x_1 + \theta_{22} x_2 + \theta_{23} x_3)$$

$$a_3 = \Omega (\theta_{30} + \theta_{31} a_1 + \theta_{32} a_2)$$

$$\hat{y} = a_3$$

- Precisamos iniciar os parâmetros aleatoriamente para quebrar simetria!!

Relembrando a derivada



$$a_1 = \Omega(\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2)$$

$$a_2 = \Omega(\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2)$$

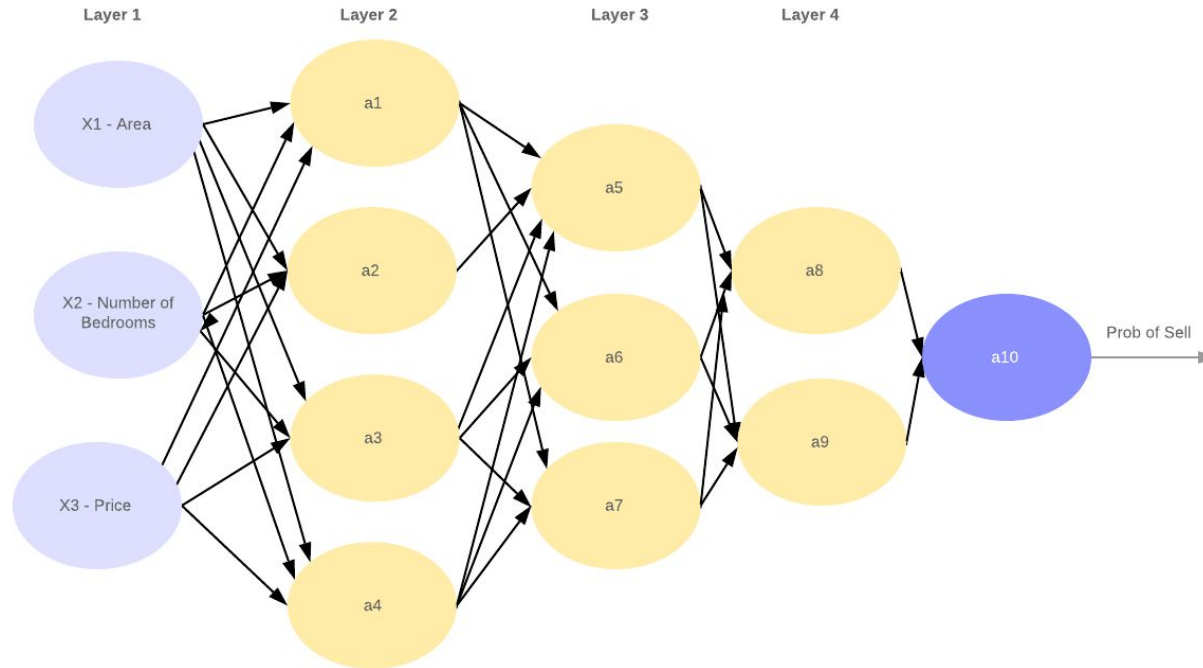
$$a_3 = \Omega(\theta_{30} + \theta_{31}a_1 + \theta_{32}a_2)$$

$$\hat{y} = a_3$$

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

$$J(\theta) = -\sum (y_i \log(\hat{y}) + (1 - y_i) \log(1 - \hat{y}))$$

Amplificando e reduzindo os sinais



Obstáculos em se treinar redes neurais



1. Gradientes desaparecendo ou explodindo dificultam o treino das camadas iniciais
2. Dados insuficientes para tantos parâmetros
3. Pode ser muito demorado
4. Modelo com milhões de parâmetros pode levar ao sobreajuste



Gradientes desapareciendo, gradientes explotando

Gradientes desaparecendo e explodindo



Problemas com gradientes:

- A. Gradiente desaparecendo: quando o signal obtido no erro retorna às camadas iniciais, o gradiente pode estar muito pequeno e essas camadas não aprendem
- B. Gradientes explodindo: as vezes o oposto acontece, gradientes ficam muito grandes e não se chega ao parâmetro ótimo
- C. Aprendizado instável: camadas diferentes aprendem com velocidades diferentes

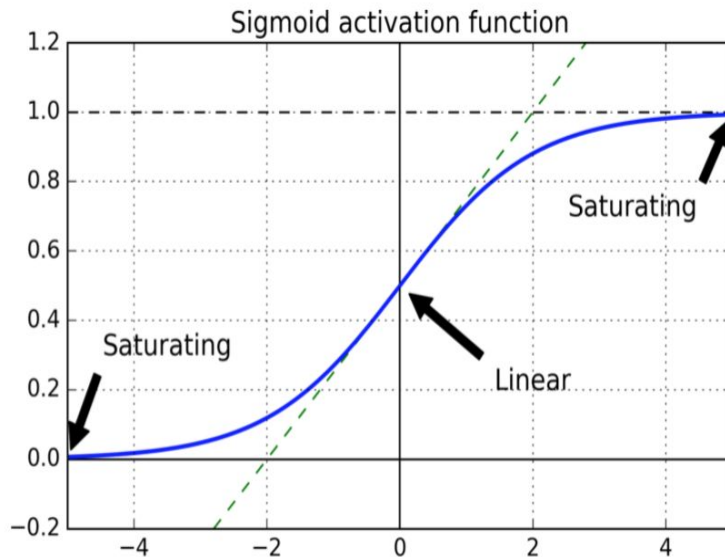
Entendendo o problema

[Glorot and Bengio \(2010\)](#) em um paper seminar explicam as principais causas do problema:

1. Sigmoid não é uma boa função de ativação
2. Especialmente quando combinada com inicialização normal(0,1)

Possíveis soluções:

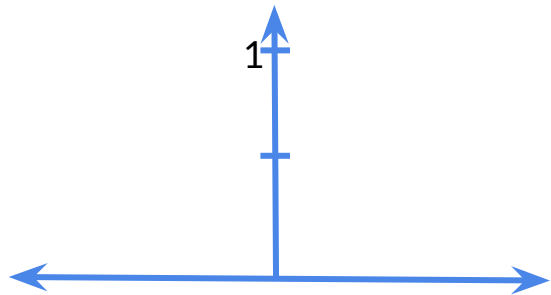
3. Método de iniciação Xavier garante que a variância do gradiente é similar em ambas as direções
4. Usar funções ativação diferentes



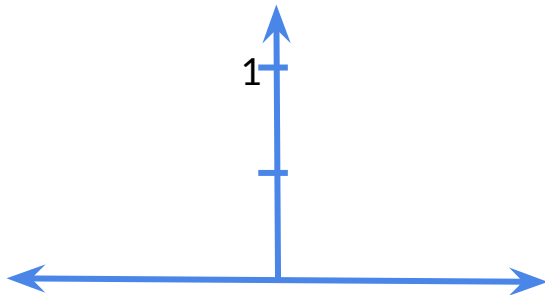
Outras funções de ativações



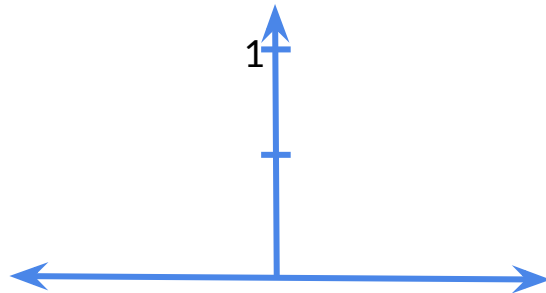
Sigmoid



Relu



Leaky Relu





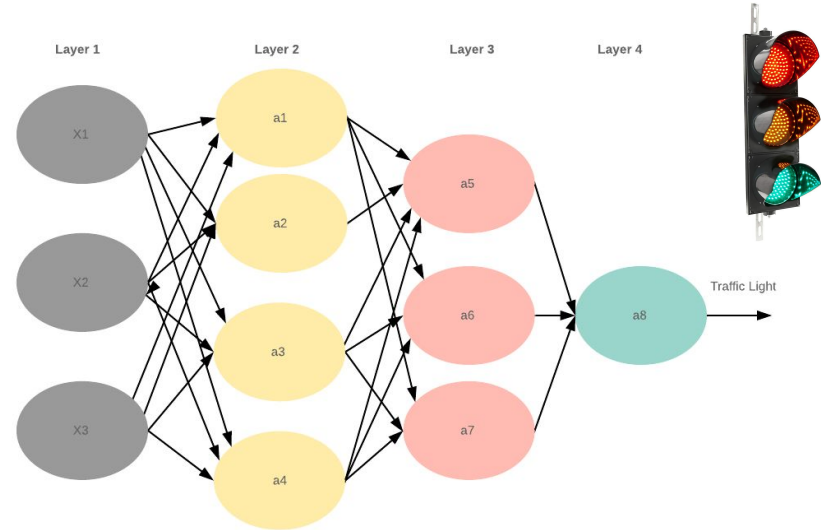
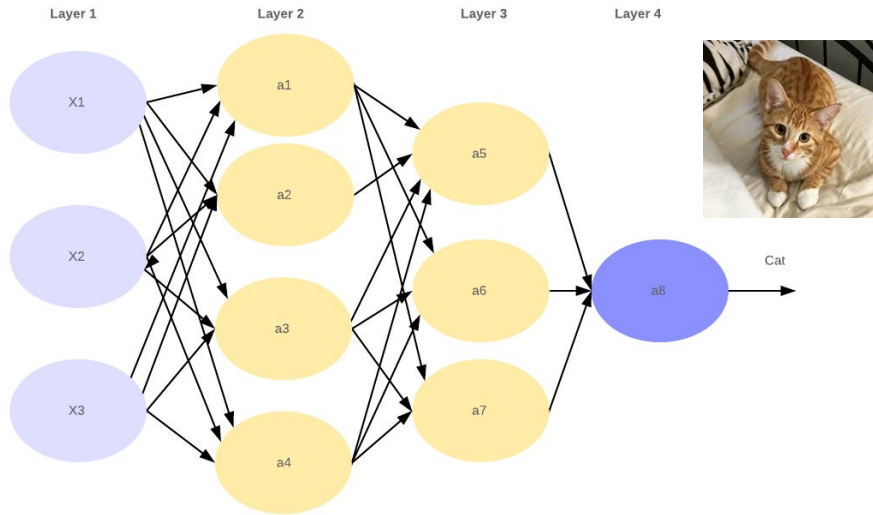
Outras técnicas de deep learning

Outras técnicas de deep learning



- A. Cortar os gradientes (ex $\text{abs}(\text{gradients}) \leq 1$)
- B. Normalizar a ativação de cada camada
- C. Transferir o aprendizado
- D. Dropout (abandono)

Transferir o aprendizado



Dropout (abandono)

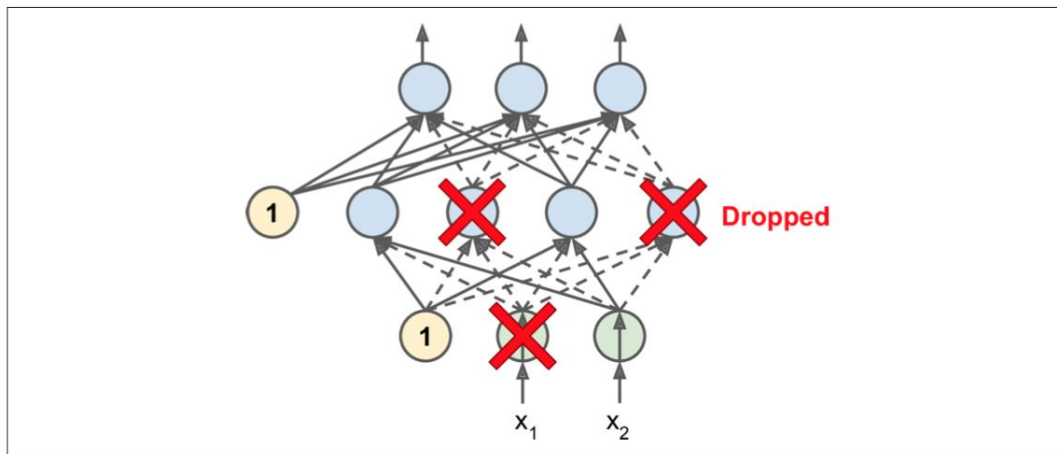
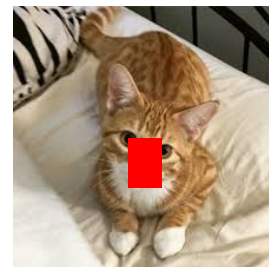


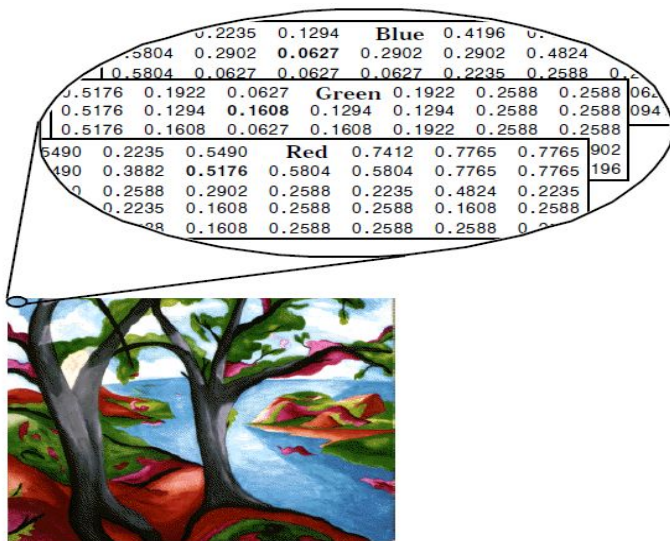
Figure 11-9. Dropout regularization





Convoluções

Problema de milhões de parâmetros



	0.2235	0.1294	Blue	0.4196	0.2902	0.4824	
0.5804	0.2902	0.0627	0.2902	0.2902	0.2902	0.4824	
0.5804	0.0627	0.0627	0.0627	0.2235	0.2588	0.2588	
0.5176	0.1922	0.0627	Green	0.1922	0.2588	0.2588	0.0627
0.5176	0.1294	0.1608	0.1294	0.1294	0.2588	0.2588	0.094
0.5176	0.1608	0.0627	0.1608	0.1922	0.2588	0.2588	
0.5490	0.2235	0.5490	Red	0.7412	0.7765	0.7765	0.902
0.490	0.3882	0.5176	0.5804	0.5804	0.7765	0.7765	0.196
0.2588	0.2902	0.2588	0.2235	0.4824	0.2235	0.2235	
0.2235	0.1608	0.2588	0.2588	0.1608	0.2588	0.2588	
0.1608	0.2588	0.2588	0.2588	0.2588	0.2588	0.2588	

O que é uma convolução



Matriz entrada

$$\begin{bmatrix} 1 & 3 & 6 & 4 \\ 2 & 2 & 0 & 4 \\ 5 & 0 & 1 & 4 \\ 4 & 4 & 1 & 3 \end{bmatrix}$$

Filtro

$$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Exemplo de convolução



$$\begin{bmatrix} 1 & 3 & 6 & 4 \\ 2 & 2 & 0 & 4 \\ 5 & 0 & 1 & 4 \\ 4 & 4 & 1 & 3 \end{bmatrix} \quad \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$



Bordas (padding) e passo (stride)

Bordas (padding) e passo (stride)



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 6 & 4 & 0 \\ 0 & 2 & 2 & 0 & 4 & 0 \\ 0 & 5 & 0 & 1 & 4 & 0 \\ 0 & 4 & 4 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Dimensões do resultado



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 6 & 4 & 0 \\ 0 & 2 & 2 & 0 & 4 & 0 \\ 0 & 5 & 0 & 1 & 4 & 0 \\ 0 & 4 & 4 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Seja w = tamanho da matriz

f = tamanho da filtro

b = tamanho da borda

p = tamanho do passo

r = tamanho do resultado

$$\Rightarrow r = \frac{w - f + 2b}{p}$$



Interpretação

O que esses filtros significam?



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Outros filtros



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	



Camadas Pooling

Camadas pooling



$$\begin{bmatrix} 1 & 3 & 6 & 4 \\ 2 & 2 & 0 & 4 \\ 5 & 0 & 1 & 4 \\ 4 & 4 & 1 & 3 \end{bmatrix}$$

$[\text{max pool}]_{2 \times 2}$

$$\begin{bmatrix} 3 & 6 & 6 \\ 5 & 2 & 4 \\ 5 & 4 & 4 \end{bmatrix}$$

$[\text{avg pool}]_{2 \times 2}$

$$\begin{bmatrix} 2 & 2.75 & 3.5 \\ 2.25 & 0.75 & 2.25 \\ 3.25 & 2 & 2.25 \end{bmatrix}$$

Por que pooling funciona



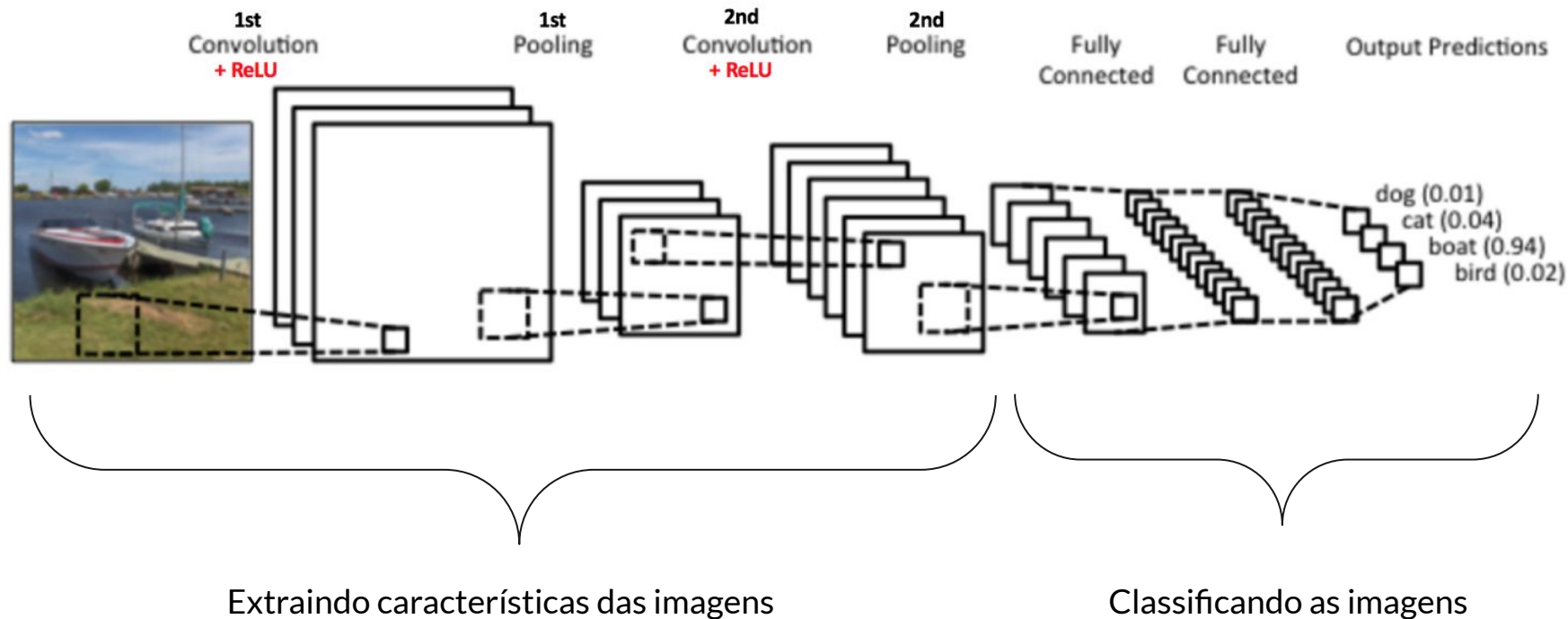
O objetivo das camadas pooling é aos poucos diminuir o tamanho da imagem representada

- Torna a representação das imagens menor e mais fácil de lidar
- Reduz o número de parâmetros e a complexidade da rede, mitigando o sobreajuste
- Torna a rede robusta a pequenas transformações, distorções e rotações na imagem
- Ajuda a chegar a uma representação equivariante da nossa imagem, de tal forma que o tamanho do objeto na foto não importa



Arquitetura de redes neurais convolucionais (CNNs)

Arquitetura das redes neurais convolucionais



Passos para se treinar redes neurais convolucionais



1. Iniciamos todos os filtros e parâmetros com valores aleatórios
2. A rede recebe a imagem no treino, passa pela propagação para frente e encontra as probabilidades de cada classe
3. Calculamos o erro total da camada de saída somando os erros de todas as classes
4. Usamos a propagação para trás para calcular o gradiente do erro com respeito a todos os **parâmetros** e aos **filtros**
5. Atualizamos os parâmetros e os filtros
6. Repetimos passos 2-5 para todas as imagens no conjunto treino



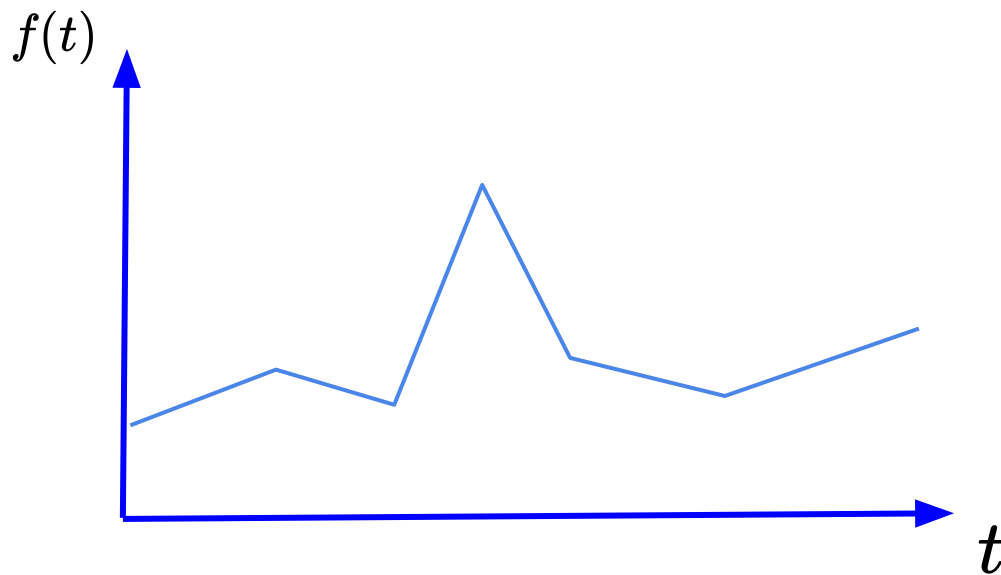
Outras aplicações

CNN podem ser usadas em mais do que reconhecimento de imagem



$$Y = \begin{bmatrix} y_{-6} \\ y_{-5} \\ y_{-4} \\ y_{-3} \\ y_{-2} \\ y_{-1} \\ y_0 \end{bmatrix} = \begin{bmatrix} 100 \\ 120 \\ 115 \\ 200 \\ 135 \\ 130 \\ 140 \end{bmatrix}$$

$$f = \left[\frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \right]$$



Por que CNNs funcionam tão bem?



- CNNs funcionam bem porque forçam localidade
- Características provavelmente são mais relacionadas outras características próximas do que com características longe
- Drasticamente reduz o número de parâmetros que temos que estimar
- Algoritmos convergem mais rapidamente sofrem menos com o sobreajuste
- Isso nos permite ter redes bem mais profundas e aprender características ainda mais complexas