(1) Introdução ao Aprendizado Profundo

Redes Neurais e Aprendizado Profundo

Moacir Antonelli Ponti
ICMC, Universidade de São Paulo

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

São Carlos-SP/Brasil

Agenda

Uma tarefa de classificação

Mudando o pipeline do aprendizado

Machine vs Deep Learning

Rede neural: do raso ao profundo

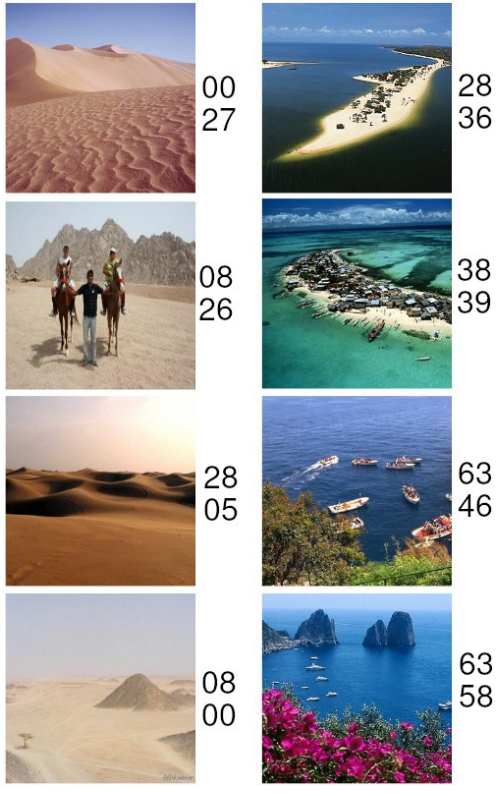
Tarefa: aprender a distinguir dois tipos de imagens

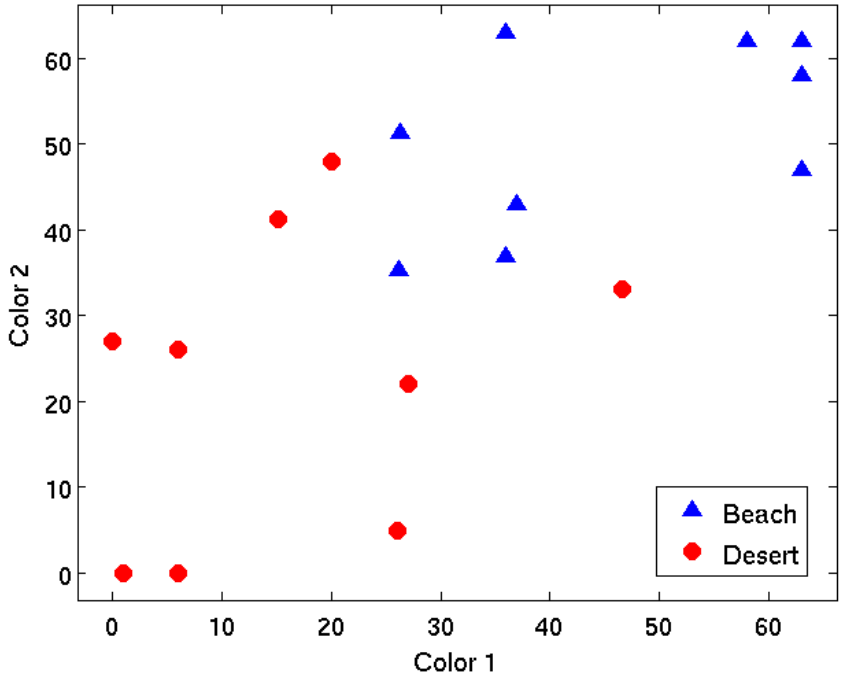
- **deserto/desert**;
- **praia/beach**.

Aprendizado supervisionado: dadas imagens anotadas (rotuladas) gere um modelo que seja capaz de classificar imagens desconhecidas (não vistas) em uma dessas duas classes.

Uma tarefa de classificação

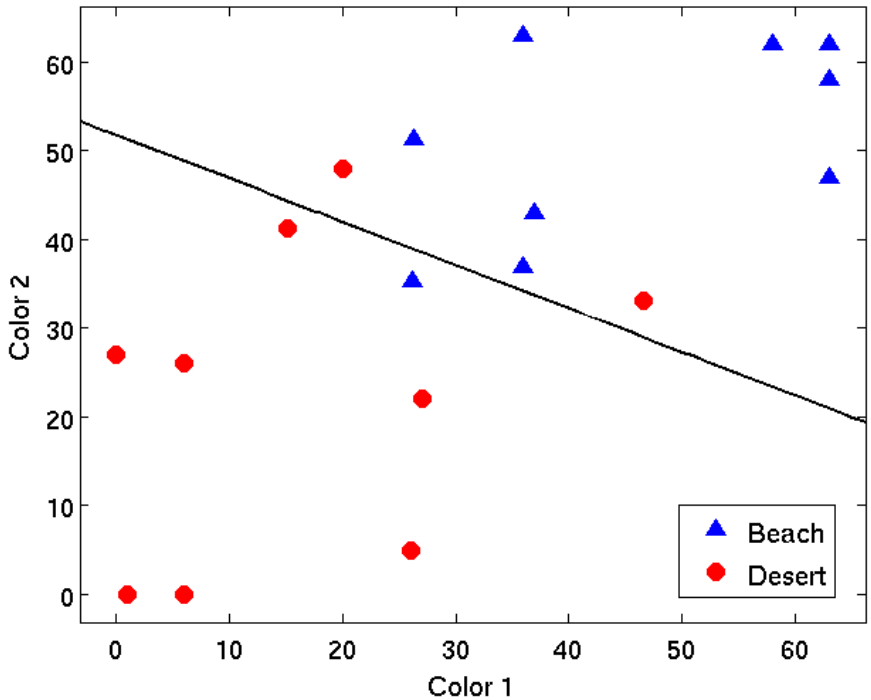
- **Atributos**: necessários para permitir medir a (dis)similaridade entre exemplos (imagens)
  - podemos usar os próprios pixels mas há desvantagens
  - vamos codificar as cores em 64 valores e representar as imagens por dois valores relativos as duas cores mais frequentes





Uma tarefa de classificação

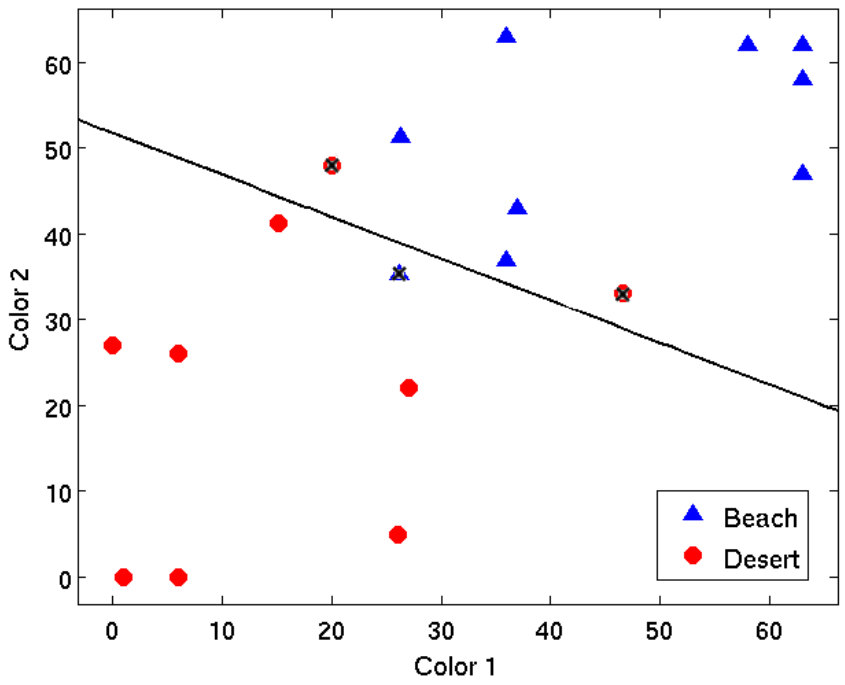
- **Classificador**: é o modelo (e não o algoritmo) criado a partir de um conjunto de dados anotado.
- ... deve permitir, com base na representação de uma nova imagem, classificá-la
  - usando a navalha de Occam: um classificador linear



,

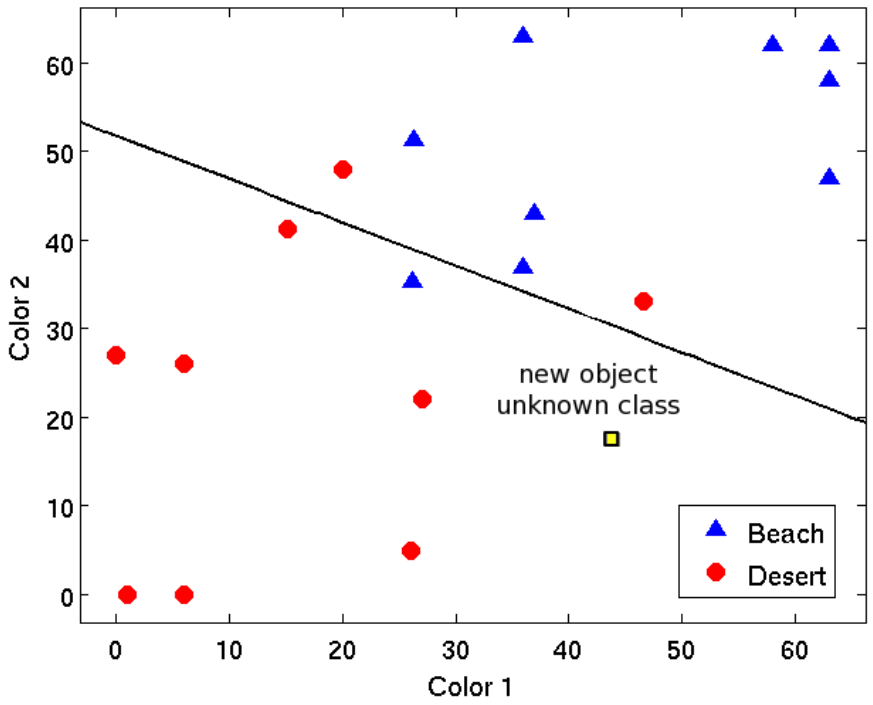
Uma tarefa de classificação

- Exemplos usados para obter o modelo/classificador: **conjunto de treinamento**.
- Dados de treinamento comumente não são completamente separáveis
- O erro do classificador nesses dados é o erro de treinamento.



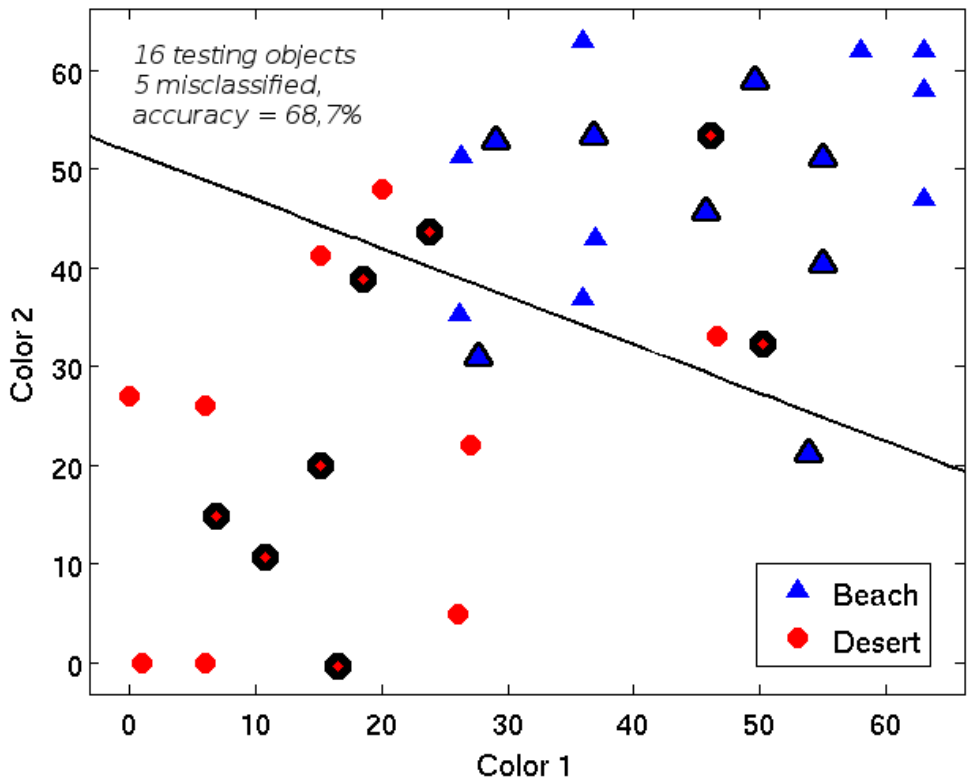
Uma tarefa de classificação

- O modelo treinado ou **classificador**, pode então ser usado para inferir/classificar um **novo dado**.

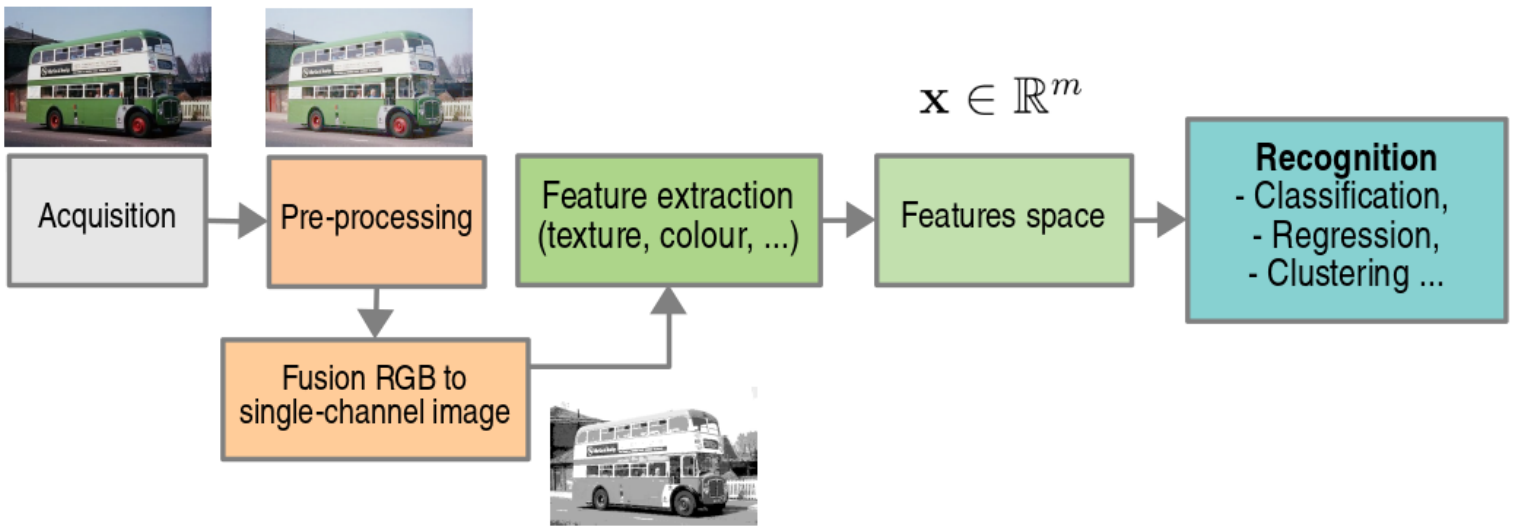


Uma tarefa de classificação

- Como saber quão bom é um modelo? Apenas testando em dados não vistos/desconhecidos pelo classificador
- ... **conjunto de teste**.



Pipeline de reconhecimento de imagens



Agenda

Uma tarefa de classificação

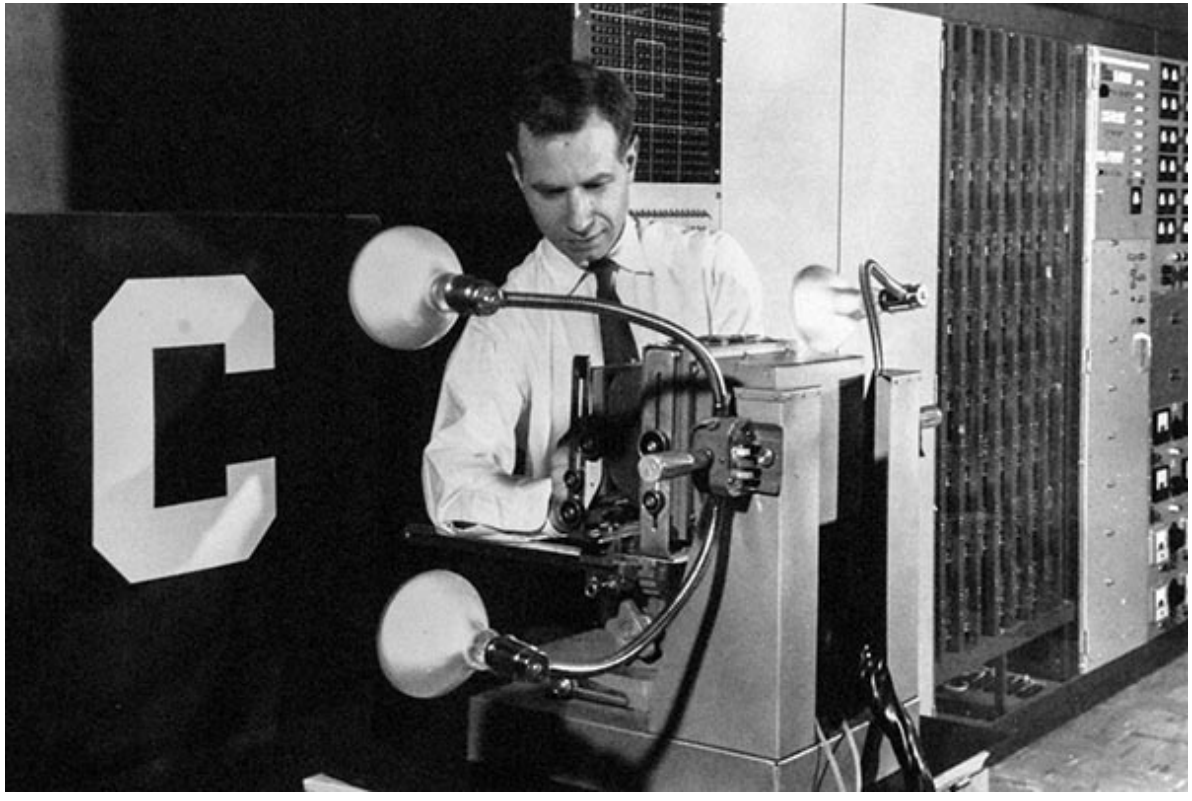
Mudando o pipeline do aprendizado

Machine vs Deep Learning

Rede neural: do raso ao profundo

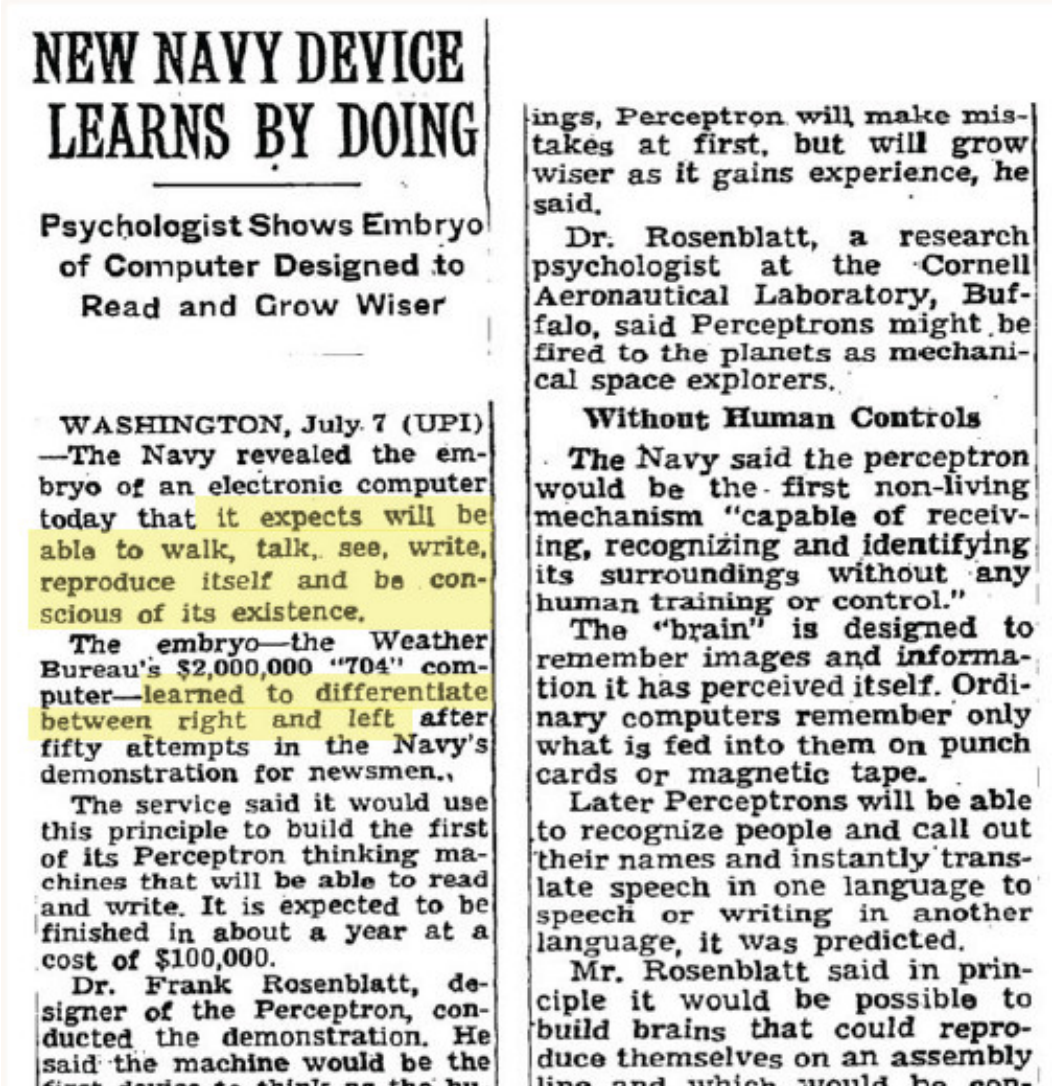
Primavera das redes neurais

- 1958: Frank Rosenblatt propõe o Perceptron como um **modelo conexionista** bioinspirado;



Primavera das redes neurais

- The New York Times, 1958 : "... o embrião de um computador ... que será capaz de andar, falar, ver, escrever, se reproduzir e ser consciente de sua existência."



NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the

ings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

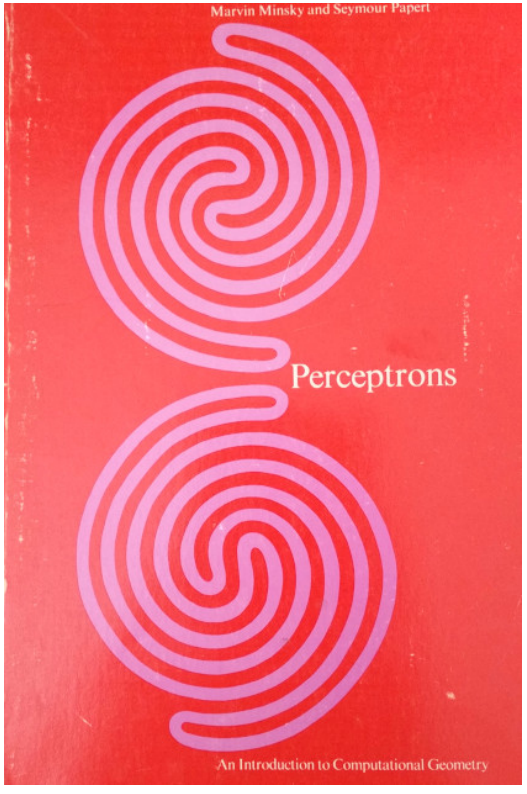
The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly

Primavera das redes neurais

- Limitações em treinar o Perceptron original foram encontradas
  - Minsky e Papert (MIT). Perceptrons, 1969

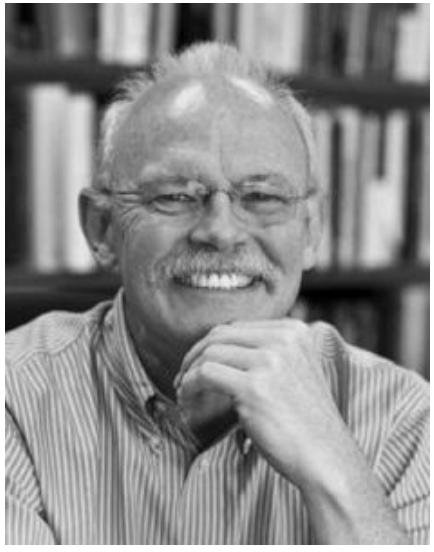


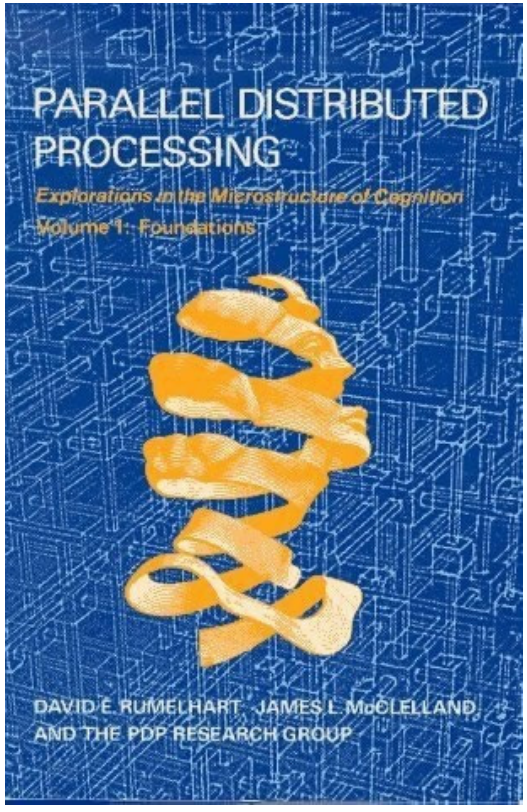
Inverno

- de 1970 até 2010 grande parte da pesquisa em "Inteligência Artificial" se referia a criar sistemas especialistas
- havia maior atenção para métodos com embasamento teórico e garantias matemáticas (como SVM):
  - "espaços de atributos/características"
  - "generalização": confiar no treinamento de um modelo;
  - "viés" ou "complexidade": reduzir a complexidade do modelo aumenta as garantias de aprendizado;
  - "tamanho da amostra" (Lei dos Grandes números): o estimador se aproxima do valor esperado conforme aumenta a quantidade de exemplos anotados.

Nova primavera

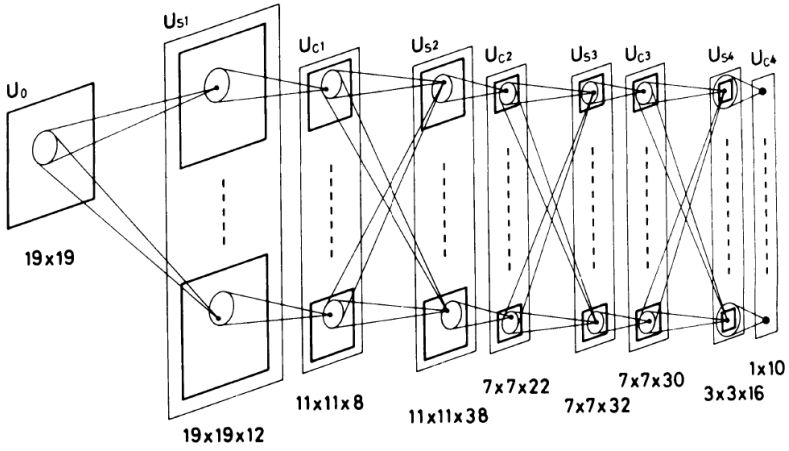
- Pesquisadores que trabalharam no inverno (1980's).
  - Rumelhart e McClelland (1986)



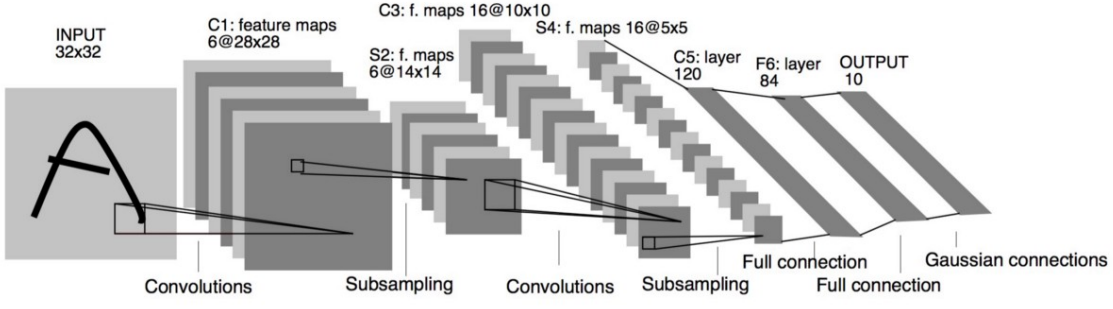


Trabalhos precursores do aprendizado profundo...

Fukushima's Neocognitron (1989)



LeCun's LeNet (1998)

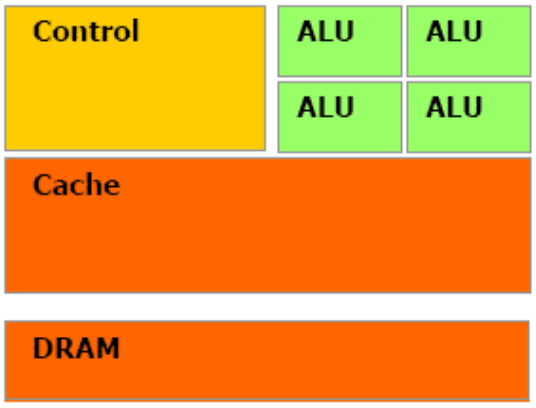


- ... e então uma nova primavera surgiu.

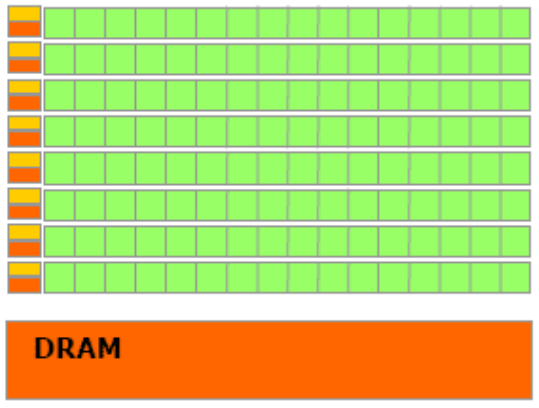
Razão 1: disponibilidade de dados anotados

ImageNet Challenge: $\sim$ 1.4 milhões de imagens, 1000 classes.

Razão 2: poder de processamento gráfico paralelo



CPU



GPU

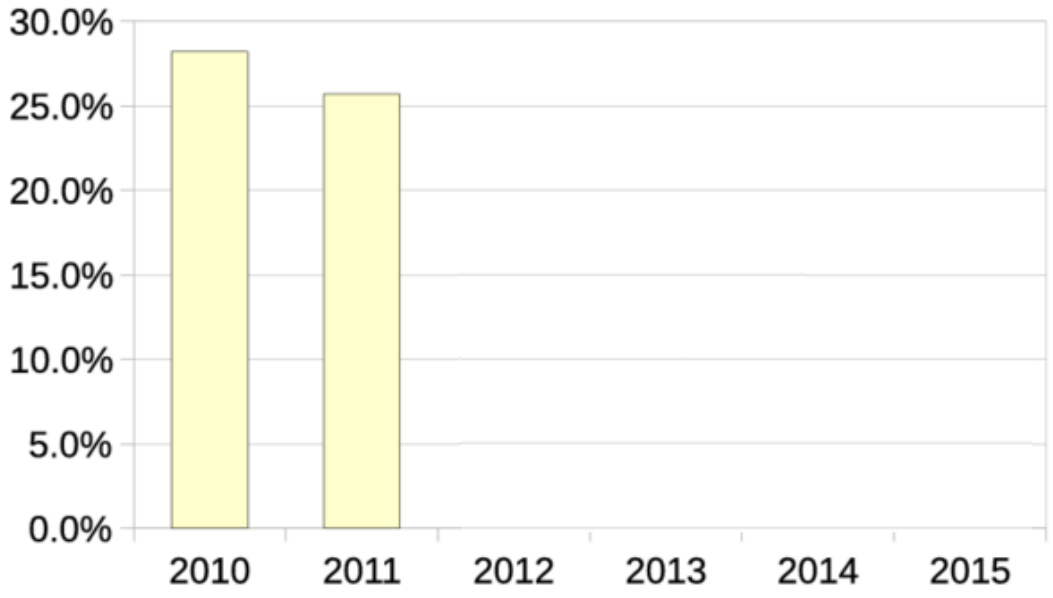
CPU

Baixa densidade de processamento
Controle complexo
Maior tamanho de cache
Baixa tolerância a latência

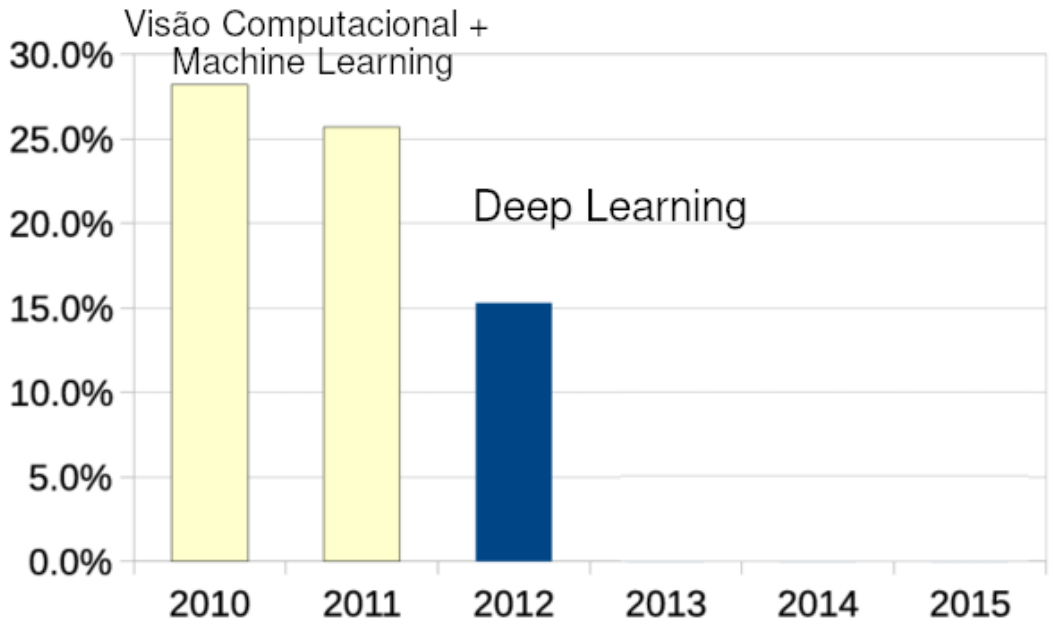
GPU

Alta densidade de processamento
Controle simples
Cache pequeno
Alta tolerância a latência

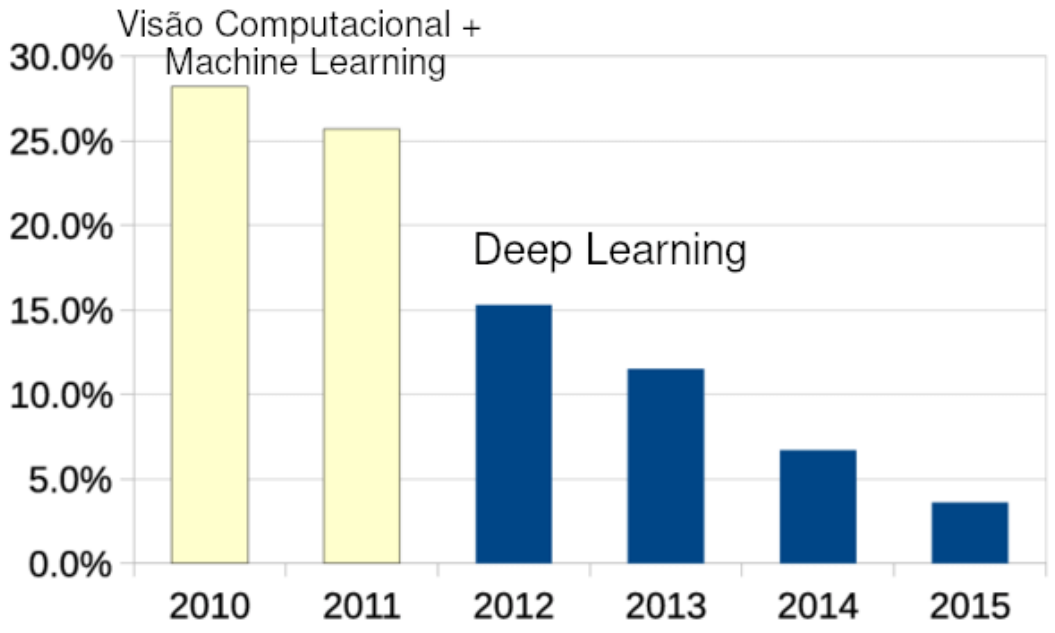
Resultados do desafio (erro de classificação top-5)



Resultados do desafio (erro de classificação top-5)

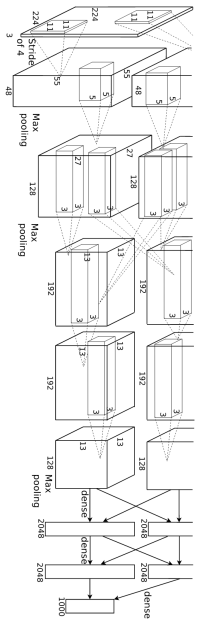


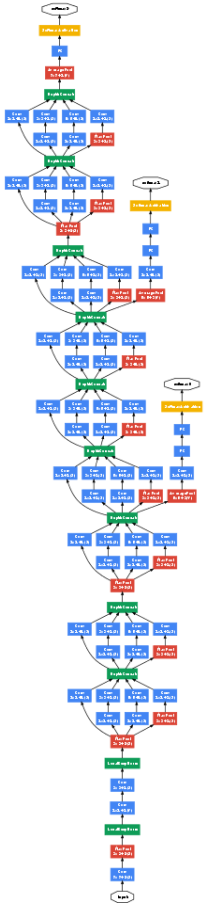
Resultados do desafio (erro de classificação top-5)

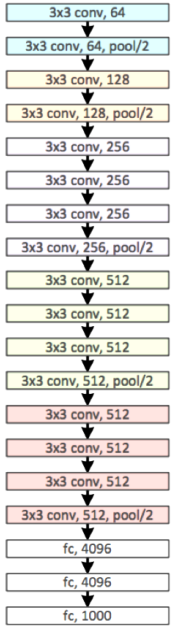


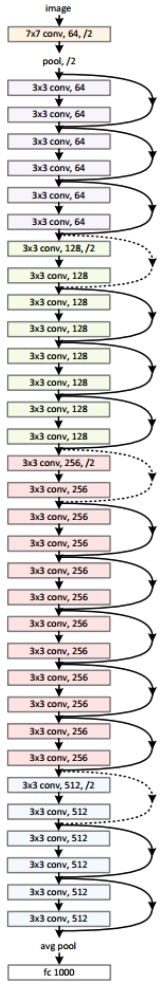
Redes neurais convolucionais profundas (e suas camadas)

AlexNet (9) Inception (22) VGGNet (16/19) ResNet (34-1000)





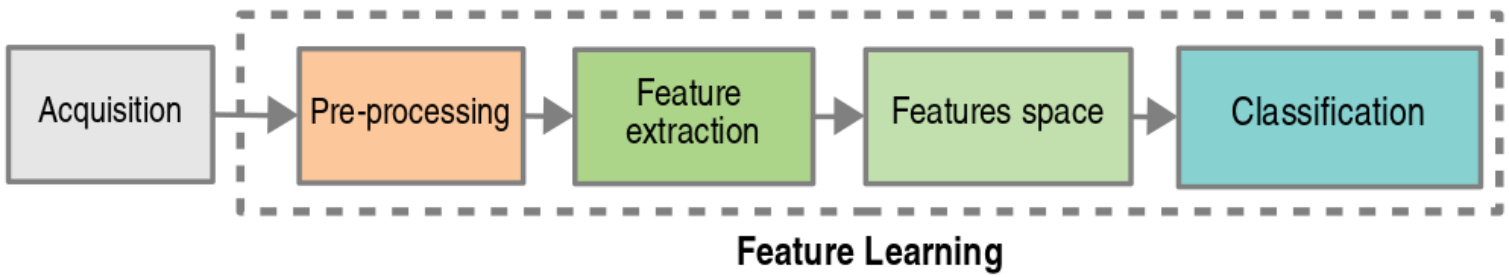




Novo pipeline de reconhecimento: aprendizado profundo



$\mathbf{x} \in \mathbb{R}^m$



Primavera das redes neurais

- Turing Award, 2018

Agenda

Uma tarefa de classificação

Mudando o pipeline do aprendizado

Machine vs Deep Learning

Rede neural: do raso ao profundo

Terminologia aprendizado supervisionado

Instância: (objeto/exemplo) de entrada $\mathrm{x}_i \in \mathbb{R}^M$

Alvo: (rótulo ou outro) de saída $y \in Y$

Dataset: $X, Y = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, for $x_i \in \mathbb{R}^M$

$I(\mathrm{x}_i) = y_i \in Y$ **rótulo** "verdadeiro" atribuído a cada exemplo.

matriz N instancias $\times$ M dimensões:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,M} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,M} \\ \cdots & \cdots & & \cdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,M} \end{bmatrix}, \text{ rótulos} = Y = \begin{bmatrix} I(\mathrm{x}_1) = y_1 \\ I(\mathrm{x}_2) = y_2 \\ \cdots \\ I(\mathrm{x}_N) = y_N \end{bmatrix}$$

Machine learning: três exemplos

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

1 – regressão: valor de um imóvel com base em suas características

- Dados disponíveis: pares (características de um imóvel, valor),
- Entrada: metragem quadrada, localização, quantidade de quartos, organizados na forma x,
- Saída: valor y (e.g. R$ 750 mil) de um determinado imóvel.

Machine learning: três exemplos

2 – 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: três exemplos

3 – 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).

Quando não envolve aprendizado?

É possível programar explicitamente a saída com base na entrada, por meio de lógica ou regras.

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

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

$$
\begin{aligned}
x_2 &= f_1(x_1) \\
x_3 &= f_2(x_2) \\
x_4 &= f_3(x_3) \\
&\dots
\end{aligned}
$$

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.

Agenda

Uma tarefa de classificação

Mudando o pipeline do aprendizado

Machine vs Deep Learning

Rede neural: do raso ao profundo

Montando um classificador

Entrada  $\rightarrow$ x

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

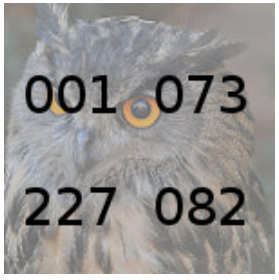
matriz de pesos | imagem | bias

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

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

Montando um classificador raso

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



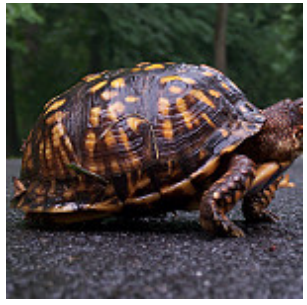
$= \mathrm{x} = [1, 73, 227, 82]$

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

$$\underbrace{\begin{bmatrix} 0.1 & -0.25 & 0.1 & 2.5 \\ 0 & 0.5 & 0.2 & -0.6 \\ 2 & 0.8 & 1.8 & -0.1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 73 \\ 227 \\ 82 \end{bmatrix} + \begin{bmatrix} -2.0 \\ 1.7 \\ -0.5 \end{bmatrix}}_{W\mathrm{x}+\mathrm{b}} = \begin{bmatrix} -37.3 \\ 6.3 \\ 38.6 \end{bmatrix}$$

Montando um classificador raso: scores obtidos





gato	-37.3	**180.3**	18.6
coruja	6.3	90.3	**26.3**
tartaruga	**38.6**	17.6	21.8

O que precisamos para melhorar esse modelo?

- uma forma de quantificar o **custo** de escolher o modelo atual
- um **algoritmo de otimização** para ajustar Θ de forma a minimizar o custo.

Montando um classificador raso: otimização

- Queremos otimizar uma função para selecionar o **melhor classificador**
- Essa função é chamada de perda (**loss**) ou custo (**cost**),

Estatisticamente, minimizar a perda esperada (**expected loss**:)

$$E\left[\mathcal{L}\left(f(\Theta, X)\right), Y\right] = \int_{\mathbb{R}^{dX} \times \mathbb{R}^{dY}} \mathcal{L}\left(f(\Theta, X), Y\right) dP(X, Y)$$

$P(x, y)$ (que modela a relação entrada e saída) é **desconhecida**, então:

- seja (x_i, y_i) um exemplo de treinamento, e $f : x \rightarrow y$ uma função de classificação atual, com parâmetros Θ.
- a perda empírica (**empirical loss**) pode ser computada:

$$\ell\left(f(\Theta, \mathrm{x}_i), y_i\right)$$

Montando um classificador raso: otimização

Perdas empíricas $\mathcal{L}(f(\Theta, X, Y))$ computadas em N exemplos

Mean squared error (valores contínuos) / perda quadrática

$$\ell(f(\Theta, X), Y) = \frac{1}{N} \sum_{i=1}^{N} (\ \overset{\text{estimado}}{\hat{y}_i} - \overset{\text{real}}{y_i}\ )^2$$

Cross entropy (bits ou vetores de probabilidade) / entropia cruzada

$$\ell(f(\Theta, X), Y) = \frac{1}{N} \sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

Minimizando a perda

Usa-se a inclinação da função de perda com relação aos parâmetros do modelo.
Para cada direção j do espaço de parâmetros Θ

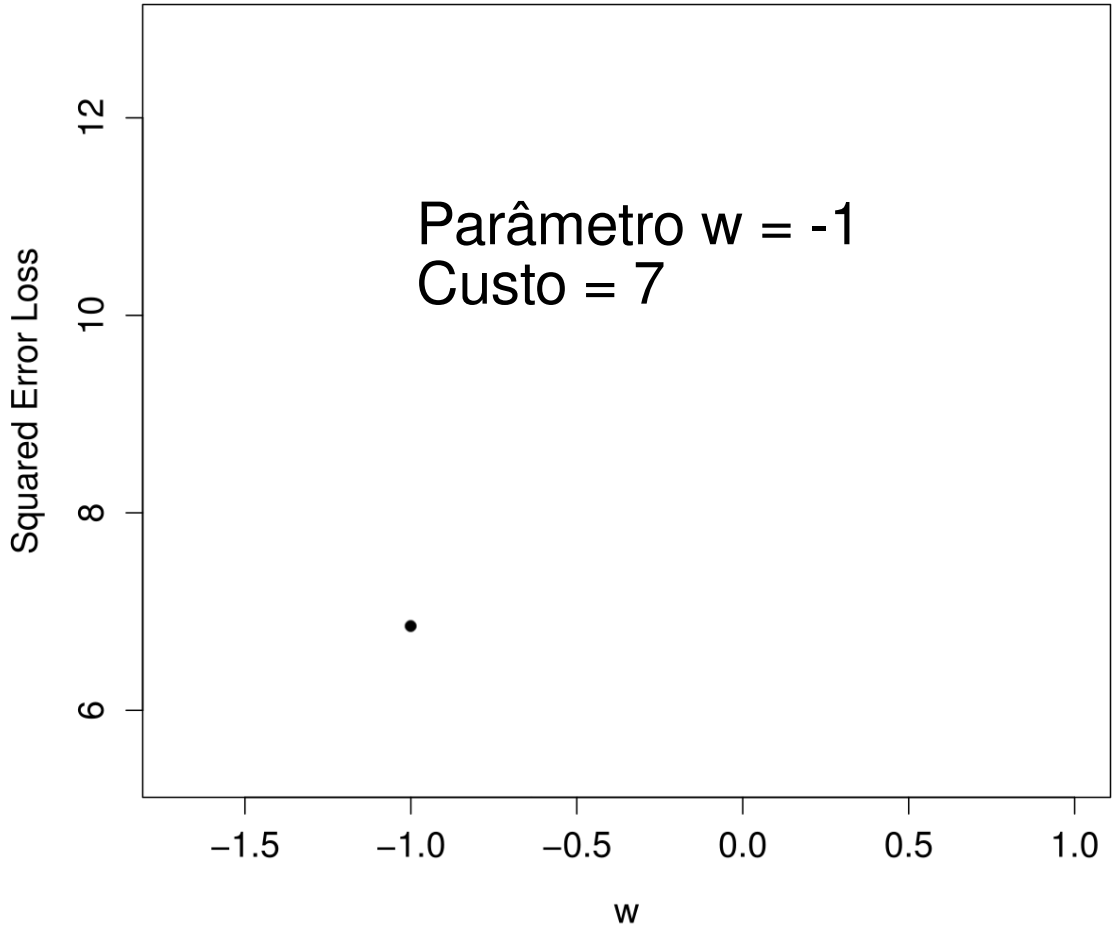
$$\frac{d\ell\left(f(w_j, \mathrm{x}_i)\right)}{dw_j} = \lim_{\delta \to 0} \frac{\ell\left(f(w_j + \delta, \mathrm{x}_i)\right) - \ell\left(f(w_j, \mathrm{x}_i)\right)}{\delta}$$

Muitas dimensões (parâmetros) resultam em um gradiente (vetor de derivadas).

Na prática computamos o gradiente numérico e buscamos pelo vale (mínimo) por meio da descida do gradiente (**Gradient descent**).

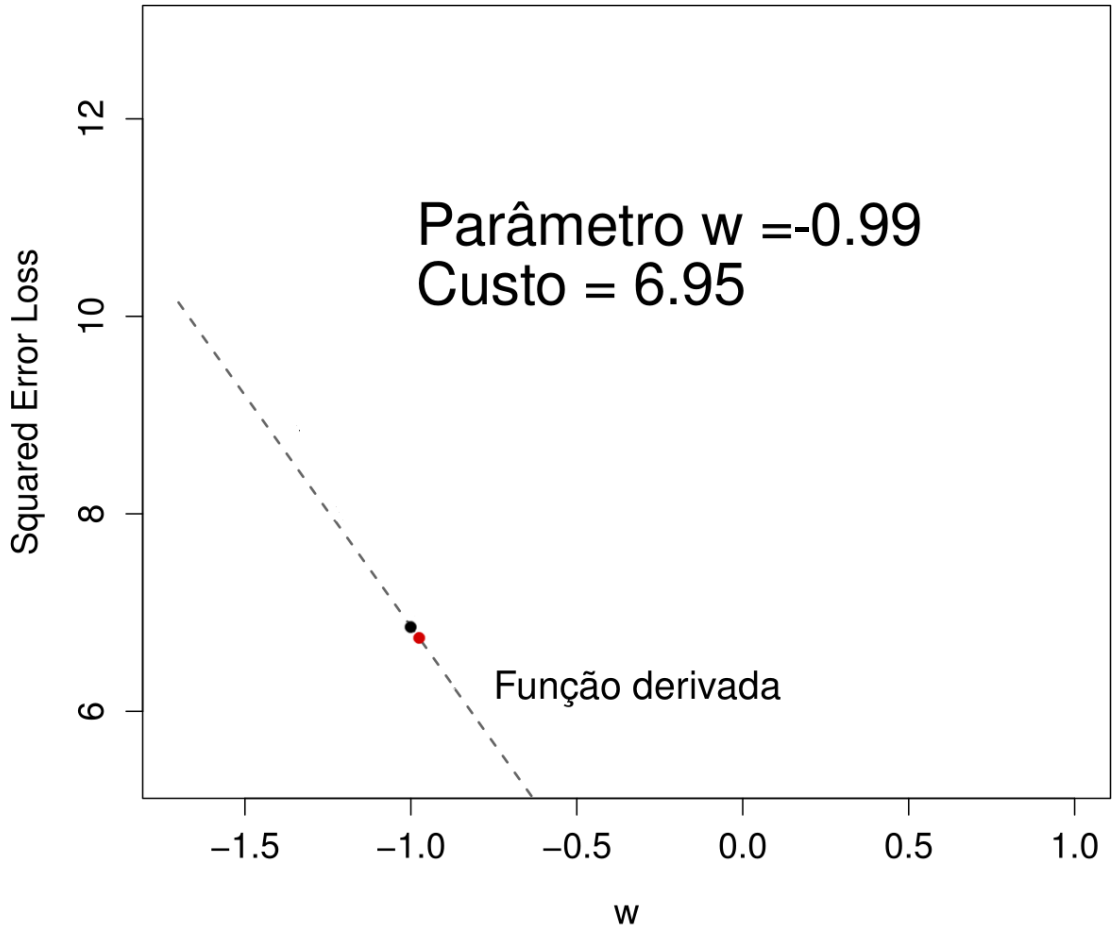
Intuição do método

Inicializamos o parâmetro com um valor arbitrário e computamos o custo



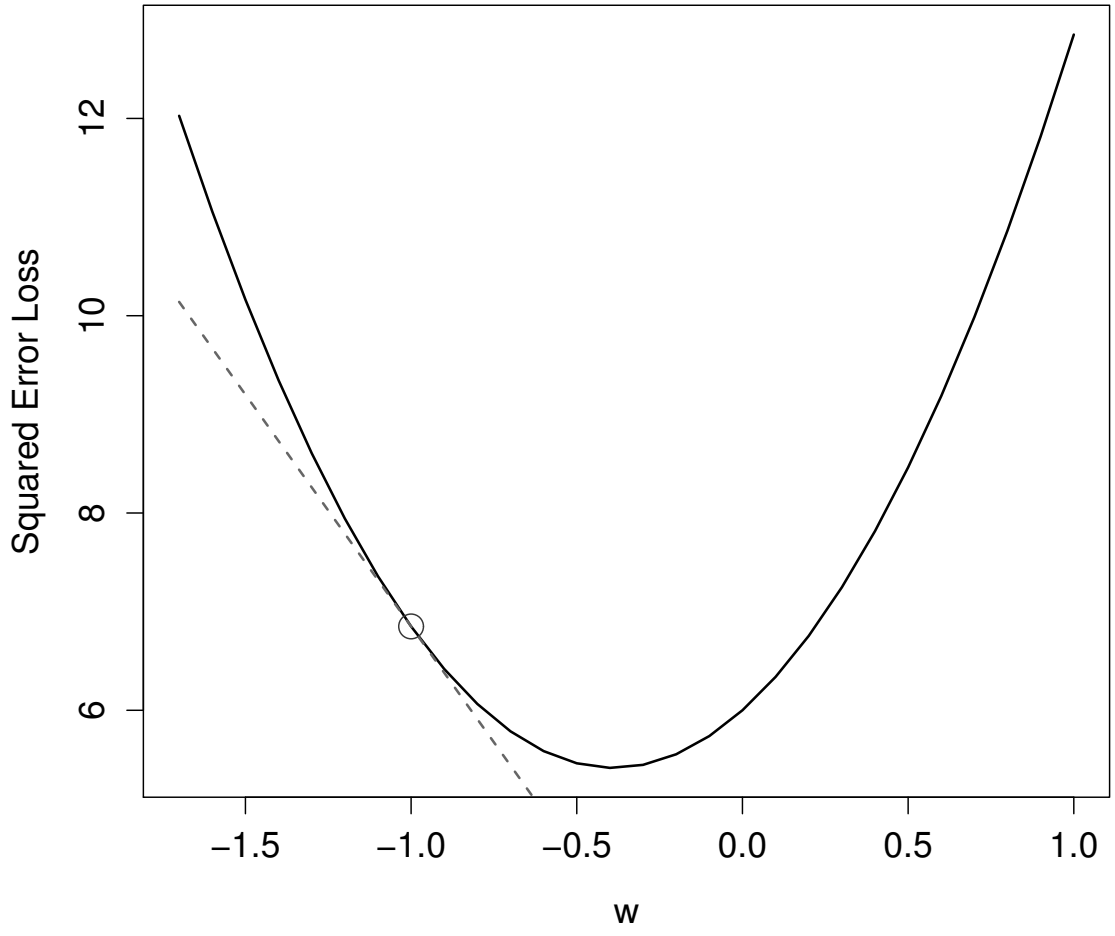
Intuição do método

A inclinação da derivada (diferença entre os custos) indica a direção que devemos seguir



Intuição do método

O que gostaríamos é que a função de custo fosse convexa, i.e. com mínimo global



Gradient descent / descida do gradiente - intuição

Dado um exemplo de treinamento ajustamos cada parâmetro do modelo

W

$$\begin{bmatrix} 0.1, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W)) = 2.31298$

$w_i + \delta$

$$\begin{bmatrix} 0.1 + 0.001, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W')) = 2.31201$

dw_i

$$\begin{bmatrix} ?, \\ , \\ , \\ , \\ , \\ ..., \\ \end{bmatrix}$$

$(f(w_i+\delta)-f(w_i))/\delta$

Gradient descent

W

$$\begin{bmatrix} 0.1, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W)) = 2.31298$

$w_i + \delta$

$$\begin{bmatrix} 0.1 + 0.001, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W')) = 2.31201$

dw_i

$$\begin{bmatrix} -0.97, \\ , \\ , \\ , \\ , \\ ..., \end{bmatrix}$$

$(f(w_i+\delta)-f(w_i))/\delta$

Gradient descent

W

$$\begin{bmatrix} 0.1, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W)) = 2.31298$

$w_i + \delta$

$$\begin{bmatrix} 0.1, \\ -0.25 + 0.001, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W')) = 2.31298$

dw_i

$$\begin{bmatrix} -0.97, \\ 0.0, \\ , \\ , \\ , \\ ..., \end{bmatrix}$$

$(f(w_i+\delta)-f(w_i))/\delta$

Gradient descent

W

$$\begin{bmatrix} 0.1, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W)) = 2.31298$

$w_i + \delta$

$$\begin{bmatrix} 0.1, \\ -0.25, \\ 0.1 + 0.001, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W')) = 2.31459$

dw_i

$$\begin{bmatrix} -0.97, \\ 0.0, \\ +1.61, \\ -, \\ -, \\ ..., \\ - \end{bmatrix}$$

$(f(w_i+\delta)-f(w_i))/\delta$

Gradient descent

W

$$\begin{bmatrix} 0.1, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W)) = 2.31298$

$w_i + \delta$

$$\begin{bmatrix} 0.1, \\ -0.25, \\ 0.1, \\ 2.5, \\ 0, \\ ..., \\ -0.1 \end{bmatrix}$$

$\ell(f(W')) = 2.08720$

dw_i

$$\begin{bmatrix} -0.93, \\ 0.0, \\ -1.61, \\ +0.02, \\ +0.5, \\ ..., \\ -3.7 \end{bmatrix}$$

$(f(w_i+\delta)-f(w_i))/\delta$

Stochastic Gradient Descent (SGD)

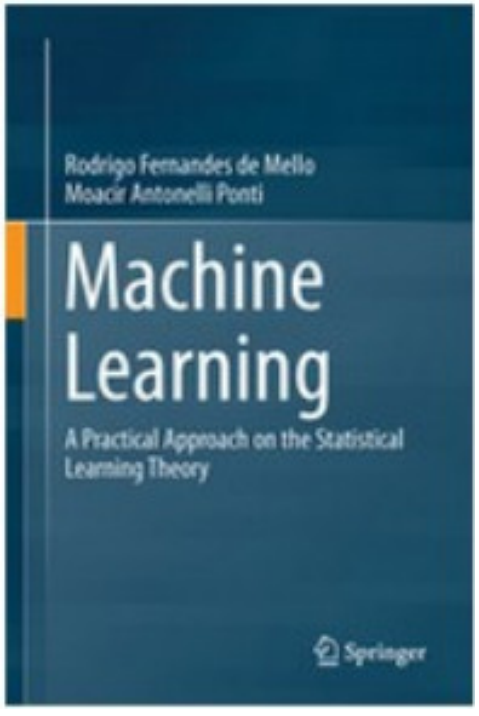
É computacionamente caro computar o gradiente para N grande

SGD:
aproximar a perda empírica usando um lote aleatório **minibatch** de instâncias: 10 até 2000+.

- mais rápido
- mais grosseiro, necessitando mais iterações

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.