

Aprendizado de Máquina

Natanael Junior^{1 2}, Hellen Lima^{1 2} e Lucas Cerqueira^{1 2}
natmourajr@lps.ufrj.br, hellenlima@poli.ufrj.br e lucas.cerqueira@poli.ufrj.br

¹Universidade Federal do Rio de Janeiro

²Laboratório de Processamento de Sinais

22/03/2017

Estrutura da Apresentação

- 1 Introdução
- 2 Redes Neurais Artificiais
- 3 Deep Learning
- 4 Aprendizado não-supervisionado
- 5 Referências Bibliográficas

Por que falar de Aprendizado de Máquina?

Definição de Aprendizado de Máquina

“Campo de pesquisa que dá aos computadores a habilidade de aprender sem serem explicitamente programados para uma determinada tarefa”

Que ótimo, mas por que eu gostaria que um computador aprenda sem ser explicitamente programado?



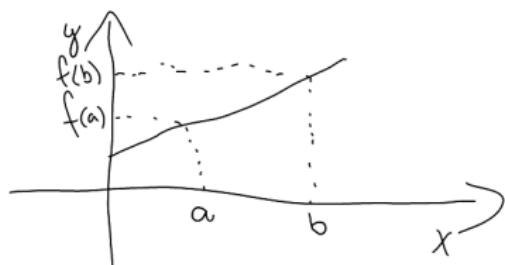
Por que falar de Aprendizado de Máquina?

O ser humano tem a capacidade de reconhecer padrões bastante complicados e lida com decisões a todo o instante

- Tudo que fazemos é baseado em decisões.
- Vida em sociedade (inteligência em colônias)

Frase de Henry Louis Mencken

“Para todo problema complexo existe sempre uma solução simples, elegante e completamente errada”



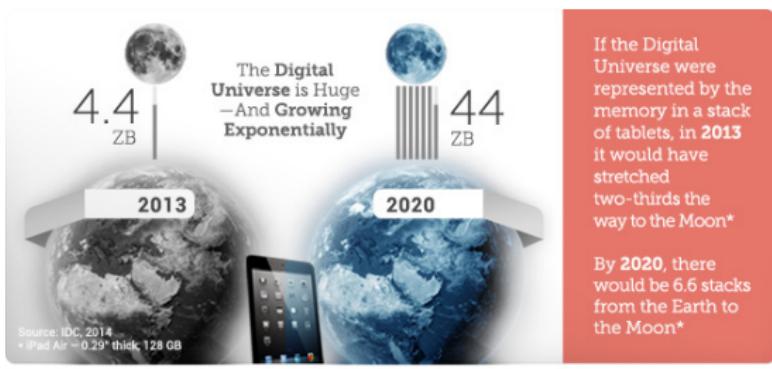
Réguas é para os fracos



Por que falar de Aprendizado de Máquina?

Todos os dias, cada **ser humano** produz uma grande quantidade de dados!!! No total, são produzidos **2,5 exabytes** de dados (em número de 2016), e isso é equivalente à

- $53 \cdot 10^7$ milhões de músicas
- $15 \cdot 10^7$ iPhones
- $25 \cdot 10^4$ Bibliotecas do Congresso Americano
- 90 anos de vídeos em HD



Por que falar de Aprendizado de Máquina?

Como analisamos esse volume dados?

- Uma quantidade enorme de dados
- Que muda muito rápido
- Em diferentes formatos
- Produzidos por fontes não confiáveis



Um pouco de história...

- **Idade Antiga:** 1943 - Neurônio Artificial [1] e 1961 - *Percetron* [2, 3].
- **Idade Média:** 1968 - É comprovado **matematicamente** que o *Percetron* só tem sucesso em separação de classes linearmente separáveis [4, 5]
- **Idade Moderna:** Década de 1980 - *MultiLayer Perceptron* (MLP) [6], **Backpropagation** [7], Teorema do Aproximador Universal [8] e a Bíblia de Redes Neurais [9].
- **Idade Contemporânea:** Década de 1990 - SVM [10], Redes Bayesianas [11]
- **Idade Contemporânea:** Metade da Década de 1990 - Aprendizado dirigido pelo conhecimento do especialista → Aprendizado dirigido pelos dados

A História breve de Aprendizado de Máquina (cont.)

- **Idade Contemporânea:** Ensemble de Classificadores (*bagging, boosted* e derivados) [12] → *Random Florest, Boosted Decision Trees*
- **Idade Contemporânea:** *Extreme Learning Machines* [13] → *Reservoir Learning* → *Echo-State Networks*
- 1997 - *IBM Deep Blue* “derrota” Garry Kasparov [14]
- 2006 - *Deep Learning* - o termo é cunhado pela primeira vez [15]
- 2010 - **Microsoft Kineck** - um aparelho utiliza **aprendizado de máquina** para fazer processamento de imagens em tempo real em **escala industrial**
- 2010 - **DeepMind** - a empresa é desenvolvida como sendo uma empresa européia de aprendizado máquina que, em 2014, será comprada pelo Google
- 2011 - **IBM Watson** derrota dois campeões de *Jeopardy!* [16] - **Processamento de Linguagem Natural**

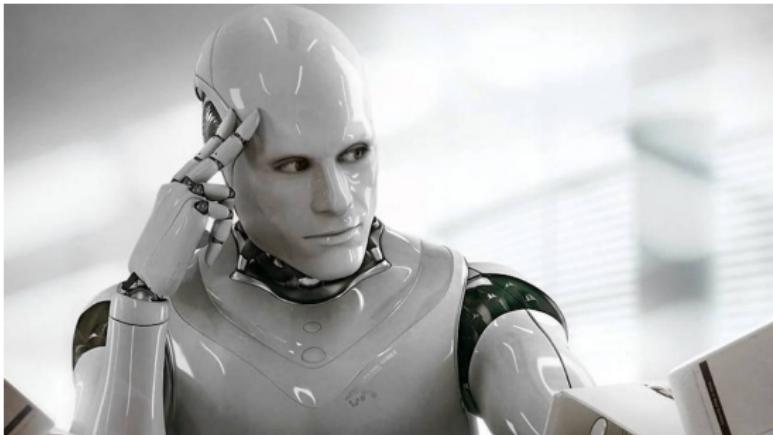
A História breve de Aprendizado de Máquina (cont.)

- 2011 - **Google Brain** - o programa é desenvolvido. Totalmente voltado para **Deep Learning**
- 2012 - **Google Brain: X Labs** - O laboratório desenvolve um algoritmo de busca autonoma para vídeos no *YouTube* e procura por ...**gatos**
- 2014 - **Facebook** - o programa **DeepFace** é desenvolvido para reconhecer e identificar automaticamente fotos da rede social
- 2015 - **Amazon** - a empresa lança um pacote de desenvolvimento para **Aprendizado de Máquina**
- 2015 - **Microsoft** - outra grande empresa lança um pacote voltado para **Aprendizado de Máquina**, desta vez com o foco em distribuição de processamento visando o ajuste de modelos
- 2016 - **Google DeepMind: AlphaGo** - um algoritmo do Google derrota o campeão mundial de Go (um jogo de tabuleiro chinês). Em comparação com o xadrez (que tem 10^{27} possíveis movimentos), o Go tem 10^{174} possíveis movimentos.

Quando usar técnicas de Aprendizado de Máquina?

Temos várias técnicas de aprendizado de máquina. Então posso sair por aí aplicando? **NÃO!**

- Existe um algoritmo, função e/ou modelo que expresse de maneira satisfatória o problema? Sim? Então, não use aprendizado de máquina
- Use aprendizado de máquina com sabedoria



Aprendizado supervisionado vs não-supervisionado

Aprendizado supervisionado

O aprendizado supervisionado se dá quando temos alvos para o processo de treinamento. O conjunto de onde os parâmetros serão extraídos é composto por dados com pares de entrada-saída. Cada par é composto por um conjunto de entradas e a saída desejada para este conjunto de entradas.

Aprendizado não-supervisionado

O aprendizado não-supervisionado se dá quando **não** temos alvos para o processo de treinamento. O conjunto de onde os parâmetros serão extraídos é composto apenas pelos dados de entrada.

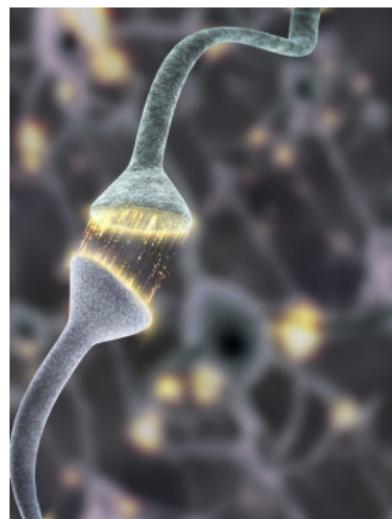
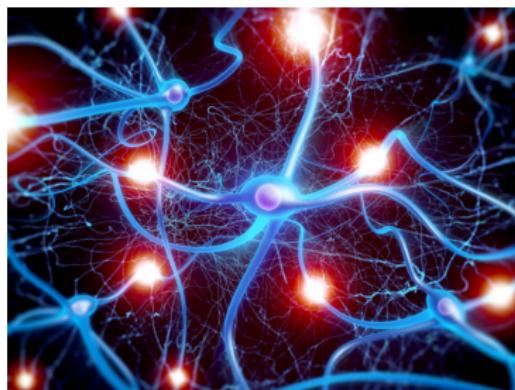
Aprendizado semi-supervisionado

O aprendizado semi-supervisionado se dá quando temos alguns eventos com **alvos** e outros **não**.

O que são Redes Neurais Artificiais?

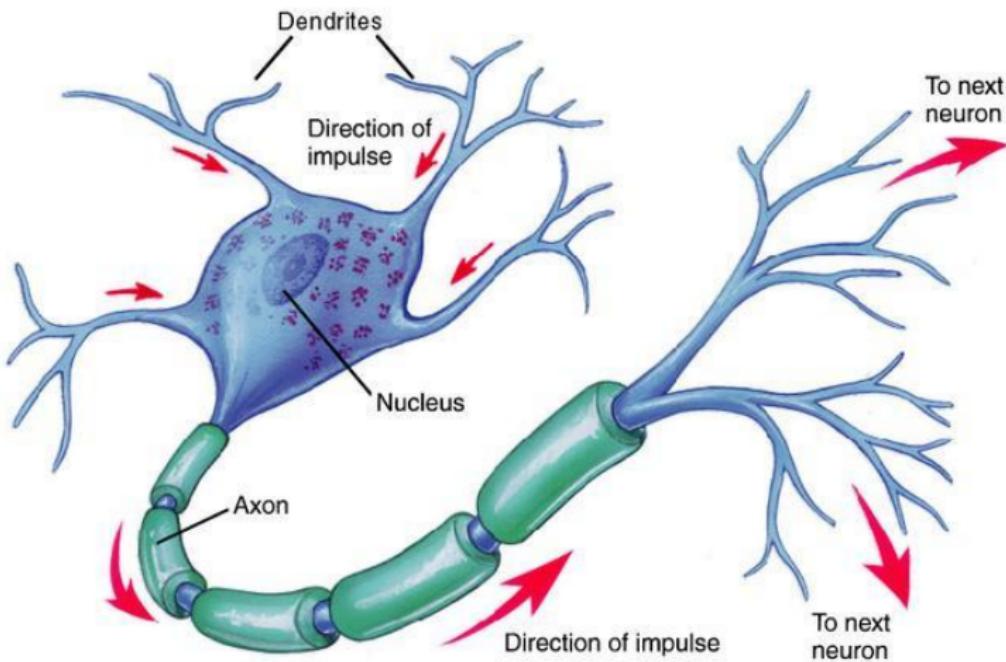
Definição de Redes Neurais Artificiais

Redes Neurais Artificiais são modelos computacionais baseados em uma quantidade de conexões de **unidades simples de processamento** (chamadas de neurônios). Inicialmente foram inspiradas no comportamento de redes neurais biológicas.



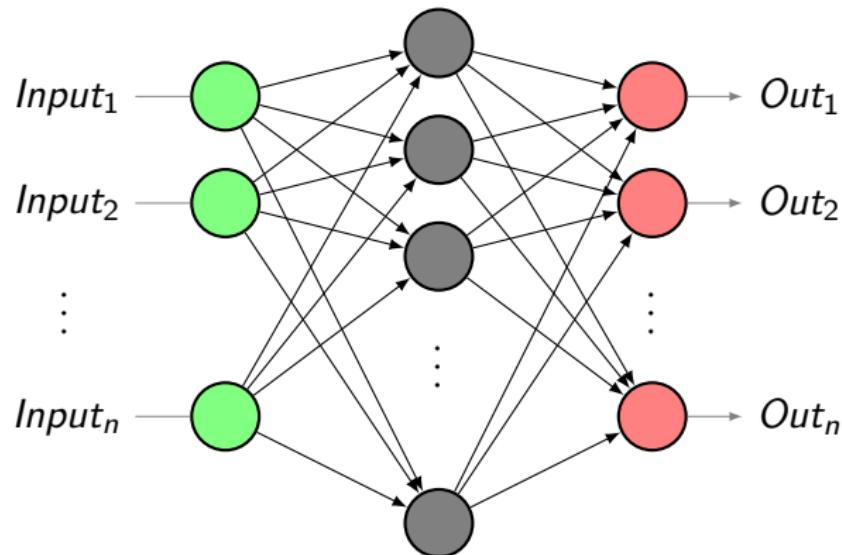
O que são Redes Neurais Artificiais?

- Neurônios Biológicos



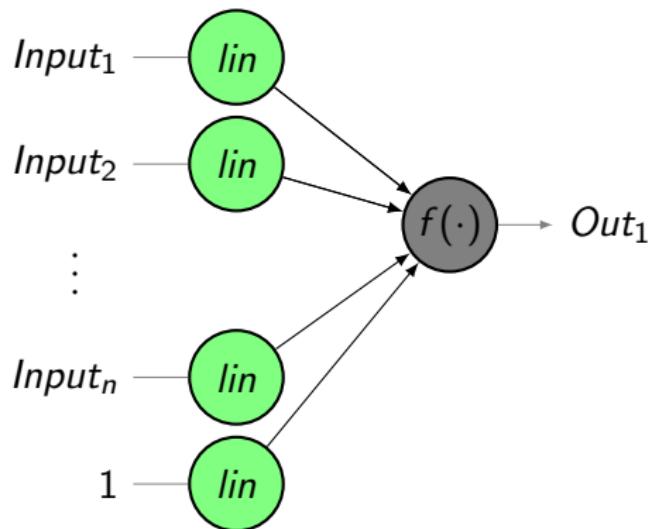
O que são Redes Neurais Artificiais?

- Redes Neurais Artificiais



O que são Redes Neurais Artificiais?

- Neurônios Artificiais



$$Out_1 = \sum_{i=1}^n f(\omega_i \cdot Input_i + \omega_{bias})$$

Modelo suficiente para separação de classes linearmente separáveis
Para classes linearmente separáveis,
mais neurônios são necessários

- Exemplo: <http://playground.tensorflow.org/>

- Treinamento → Processo de Otimização
- Erro = Custo
- Alvos → Valores desejados, Saídas → Valores estimados
- Retropropagação do erro (*backpropagation*)

$$Custo = F_{Custo}(\omega) = \frac{1}{N} \sum_{i=1}^N (Alvo_i - Out_i)^2$$

- Minimizar o custo → atualizar os pesos para obter as saídas mais próximas dos alvos
- Método mais simples: **Gradiente Descente**

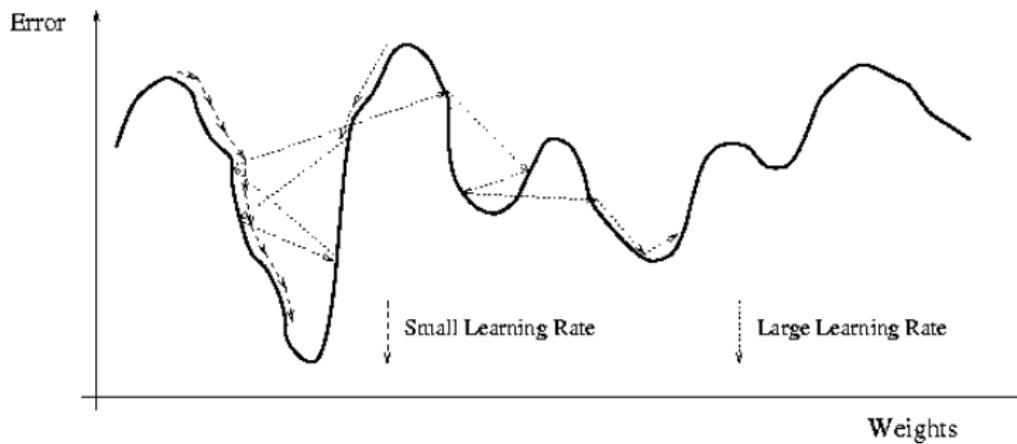
$$\omega_i(t+1) = \omega_i(t) - \eta \frac{\partial F_{Custo}}{\partial \omega_i}$$

Geralmente, um treinamento de **uma técnica de aprendizado de máquina** é feito em 3 etapas

- Etapa de Treinamento
- Etapa de Validação
- Etapa de Teste

Treinamento de Redes Neurais

Escolha do passo de aprendizado (η) → empírico, mas com **cuidado!**

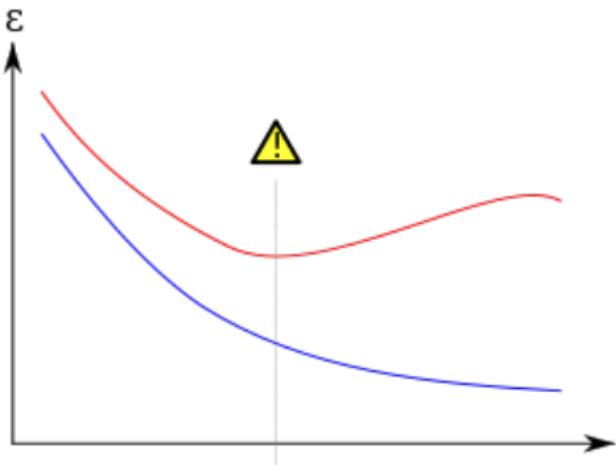
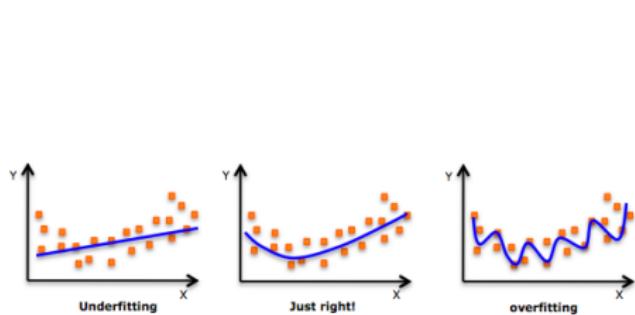


- Podemos ficar presos em um **mínimo local**.
- Para se evitar isso: diversas inicializações ou um **treinamento mais eficaz**

Treinamento de Redes Neurais

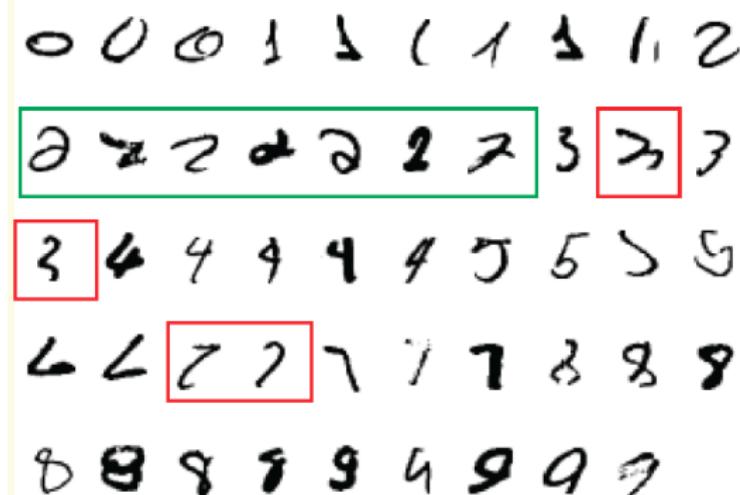
Quando temos muitos parâmetros de ajuste dos modelos, pode-se encontrar em um cenário de **overfit**.

- Que nada mais é do que uma memorização dos dados
- Com isso, temos uma perda de **generalização**



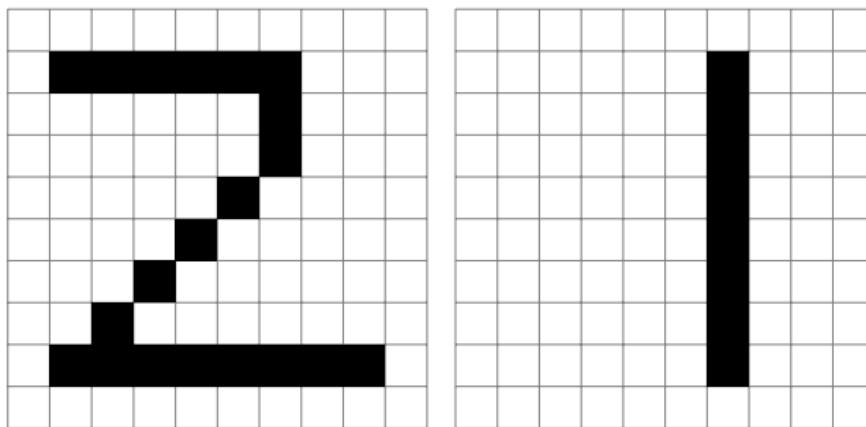
Treinamento de Redes Neurais

- Banco de Dados USPS (*United States Post Service*)
- Identificação de números escritos a mão
- Problema **Clássico** de Aprendizado de Máquina



Treinamento de Redes Neurais (cont.)

- Supondo que vamos separar “2” de “1”



- Uma maneira de **representar** dos dados é como uma matriz de pixels (**representação original** dos dados)
- Um primeiro treinamento pode ser: Somar 1 nos pixels pertencentes ao “2” e subtrair 1 dos pixels pertencentes ao “1”

Treinamento de Redes Neurais (cont.)

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0



0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0



0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	-1	0
0	0	0	0	0	0	1	-1	0	0	0
0	0	0	0	1	0	-1	0	0	0	0
0	0	0	1	0	0	-1	0	0	0	0
0	0	1	0	0	0	-1	0	0	0	0
0	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0

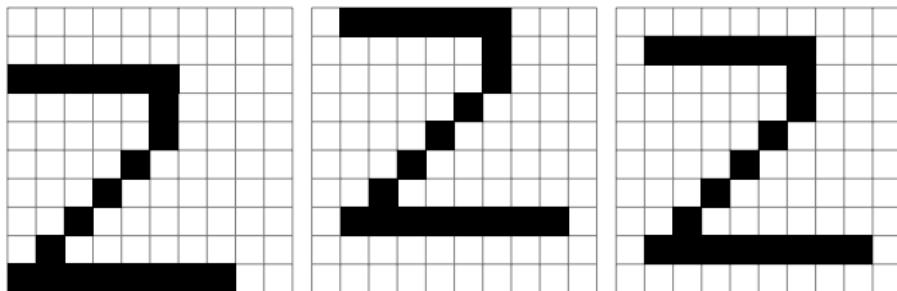
- Para fazer a classificação propriamente dita: multiplicar os valores dos pixels de um novo número pela matriz de classificação e depois somar todos os valores. Se for positivo, o número é um “2”, se não, não é.

Treinamento de Redes Neurais (cont.)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	-1	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0	0
0	0	1	0	0	0	-1	0	0	0
0	1	1	1	1	1	-1	1	1	0
0	0	0	0	0	0	0	0	0	0

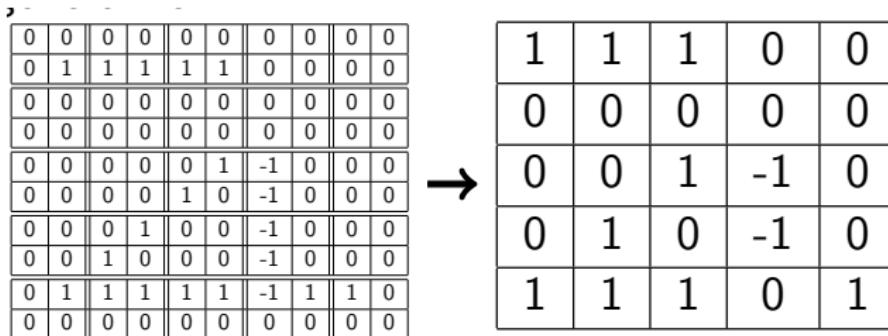
- Se novas imagens de "2" chegam, esse classificador **generaliza?**

Treinamento de Redes Neurais (cont.)



- **Generalização** é a capacidade de extender a **acurácia** da classificação a novos dados
- **Memorização dos Dados** - o classificador não generaliza

Treinamento de Redes Neurais (cont.)



The diagram illustrates a transformation from one set of data to another. On the left, there is a 10x5 matrix filled with binary values (0s and 1s). On the right, there is a second 10x5 matrix, also filled with binary values. A large black arrow points from the left matrix to the right matrix, indicating a mapping or transformation process.

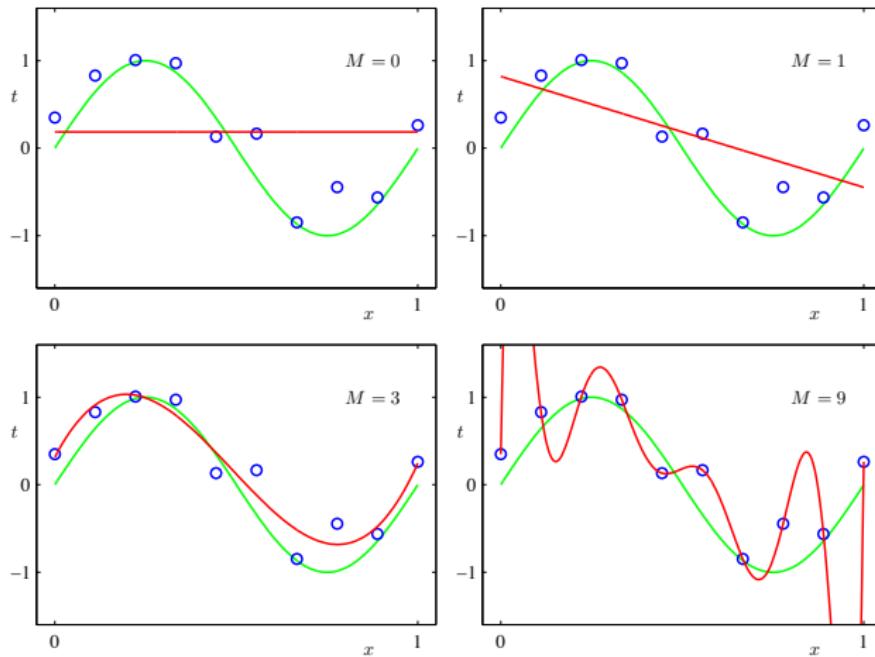
0	0	0	0	0
0	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	1
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	1	1	1
0	0	0	0	0

1	1	1	0	0
0	0	0	0	0
0	0	1	-1	0
0	1	0	-1	0
1	1	1	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

- Menos parametros a ajustar e o modelo consegue **generalizar** mais.

Treinamento de Redes Neurais (cont.)

- Um outro exemplo, o ajuste de um modelo polinomial a um conjunto de pontos, modelo: $f(x) = \omega_0 + \omega_1x + \omega_2x^2 + \cdots + \omega_nx^n$



Treinamento de Redes Neurais (cont.)

Modelo com **mais** parâmetros
Maior poder de mapeamento

Maior Probabilidade de
Memorizar dados

Maior Probabilidade de
Perda de Generalização

Overfit

Modelo com **menos** parâmetros
Menor poder de mapeamento

Menor Probabilidade de
Memorizar dados

Menor Probabilidade de
Perda de Generalização

Underfit



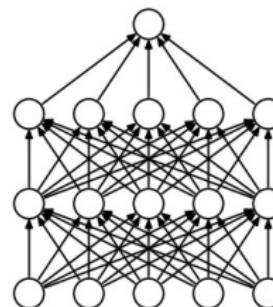
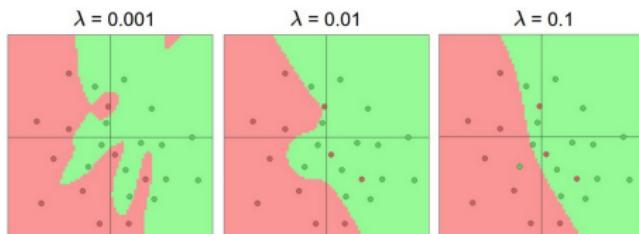
Treinamento de Redes Neurais

Podemos evitar a perda de **generalização**?

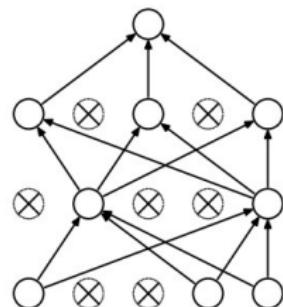
- Regularização

- $L_1: F_{Custo}(\omega) = \frac{1}{N} \sum_{i=1}^N (Alvo_i - Out_i)^2 + \lambda \sum_{\omega} \omega$
- $L_2: F_{Custo}(\omega) = \frac{1}{N} \sum_{i=1}^N (Alvo_i - Out_i)^2 + \lambda \sum_{\omega} \omega^2$

- *Dropout*



(a) Standard Neural Net



(b) After applying dropout.

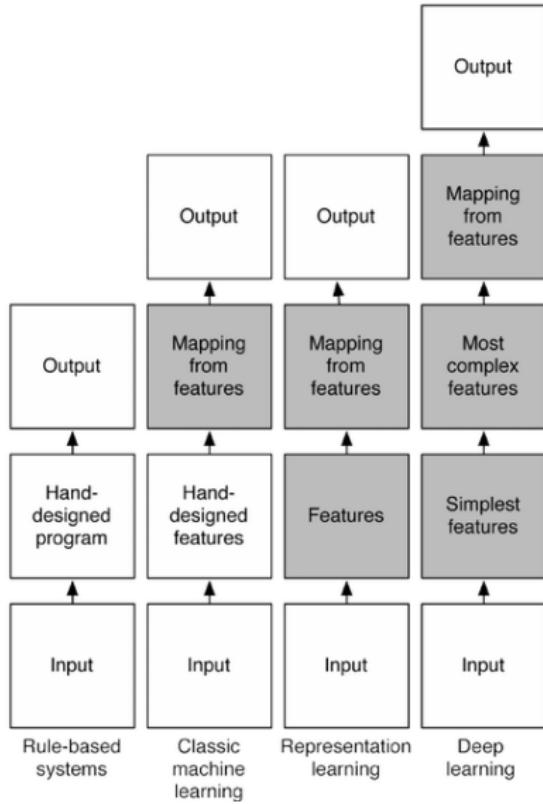
Definição de Deep Learning

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations

Deep Learning é um ramo do aprendizado de máquina com base em um conjunto de algoritmos que tentam modelar abstrações de alto nível em diferentes conjuntos de dados usando estruturas profundas com várias camadas de processamento, compostas por transformações lineares e não-lineares.

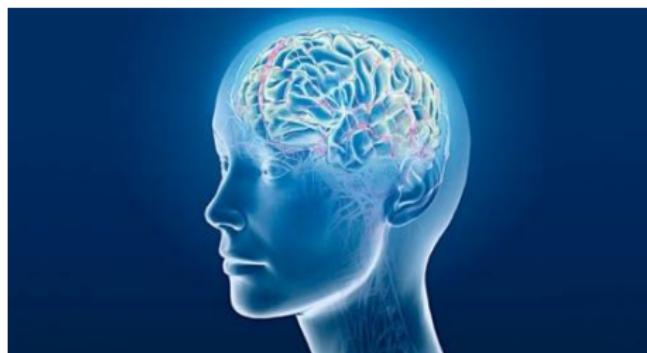
Definição de Deep Learning

- Programação baseada em Regras
- Aprendizado de Máquina Clássico
- Aprendizado de Representação
- Deep Learning



Motivação de Deep Learning

De onde mais viria...



Mas pensando bem...

- Nossos sensores são bem caracterizados: olhos, ouvidos e etc.
- Corpo humano = “sistema de aquisição de dados”
- Sensor → células especializadas em processar sinais → conexão com o cérebro
- Sensor → camadas de abstração → tomada de decisão

Motivação de Deep Learning

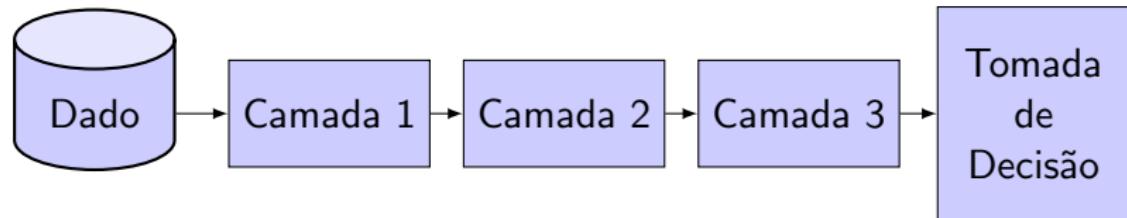
Como tomamos decisões?



Motivação de Deep Learning

Como um computador pode tomar decisões?

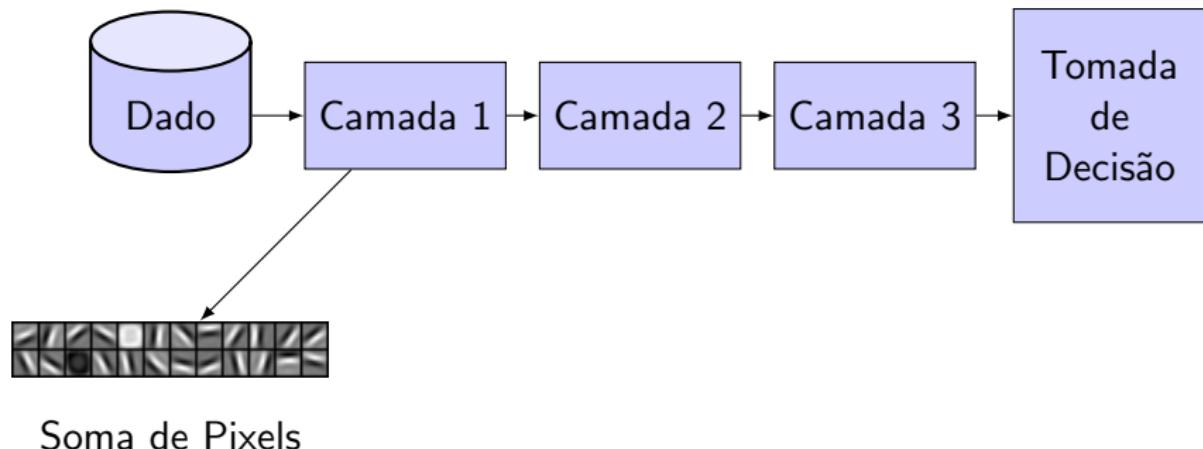
- Em [17], vemos um exemplo:



Motivação de Deep Learning

Como um computador pode tomar decisões?

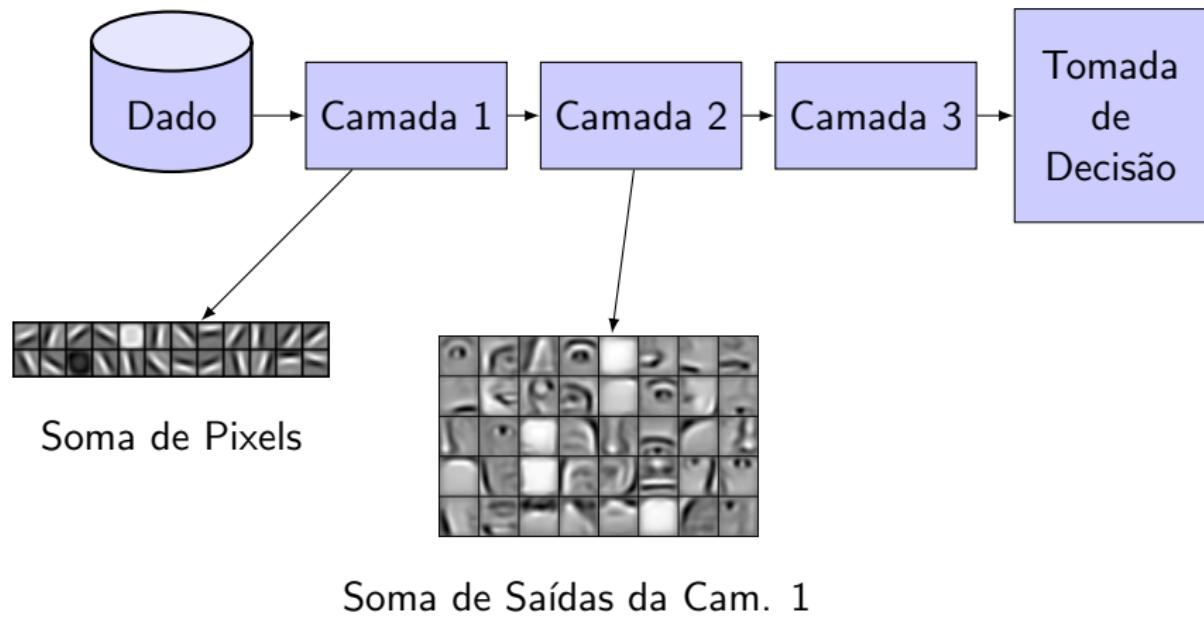
- Em [17], vemos um exemplo:



Motivação de Deep Learning

Como um computador pode tomar decisões?

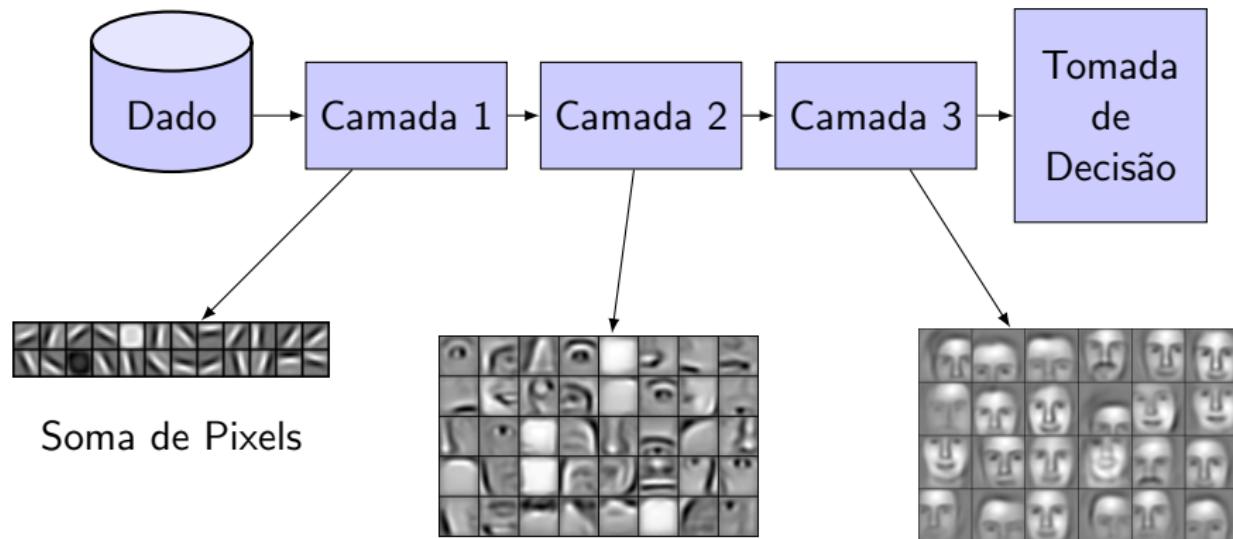
- Em [17], vemos um exemplo:



Motivação de Deep Learning

Como um computador pode tomar decisões?

- Em [17], vemos um exemplo:

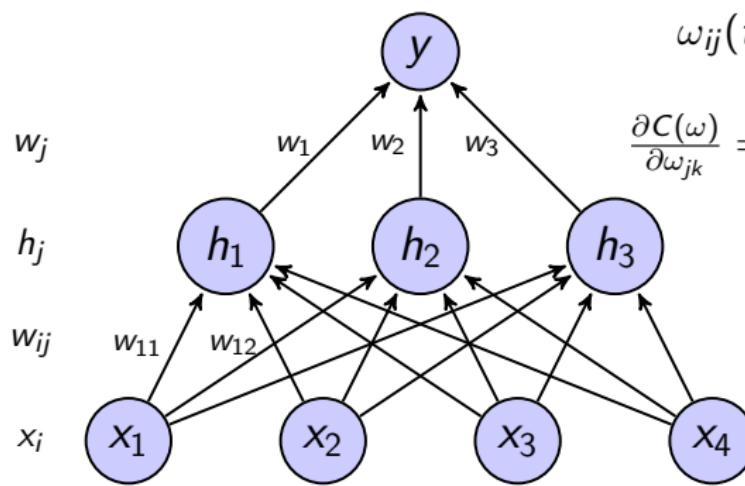


Soma de Pixels Soma de Saídas da Cam. 1 Soma de Saídas da Cam. 2

Histórico de Deep Learning

Muito bom para ser verdade...

- PROBLEMA no treinamento [18]
- Relembrando *Backpropagation*



$$\omega_{ij}(t + 1) = \omega_{ij}(t) - \eta \frac{\partial C(\omega)}{\partial \omega_{ij}}$$

$$\frac{\partial C(\omega)}{\partial \omega_{jk}} = \frac{\partial C(\omega)}{\partial in_k} \frac{\partial in_k}{\partial \omega_{jk}} = \delta_k \frac{\partial (\sum_j \omega_{jk} h_j)}{\partial \omega_{jk}}$$

$$\frac{\partial C(\omega)}{\partial \omega_{jk}} = \delta_k h_j$$

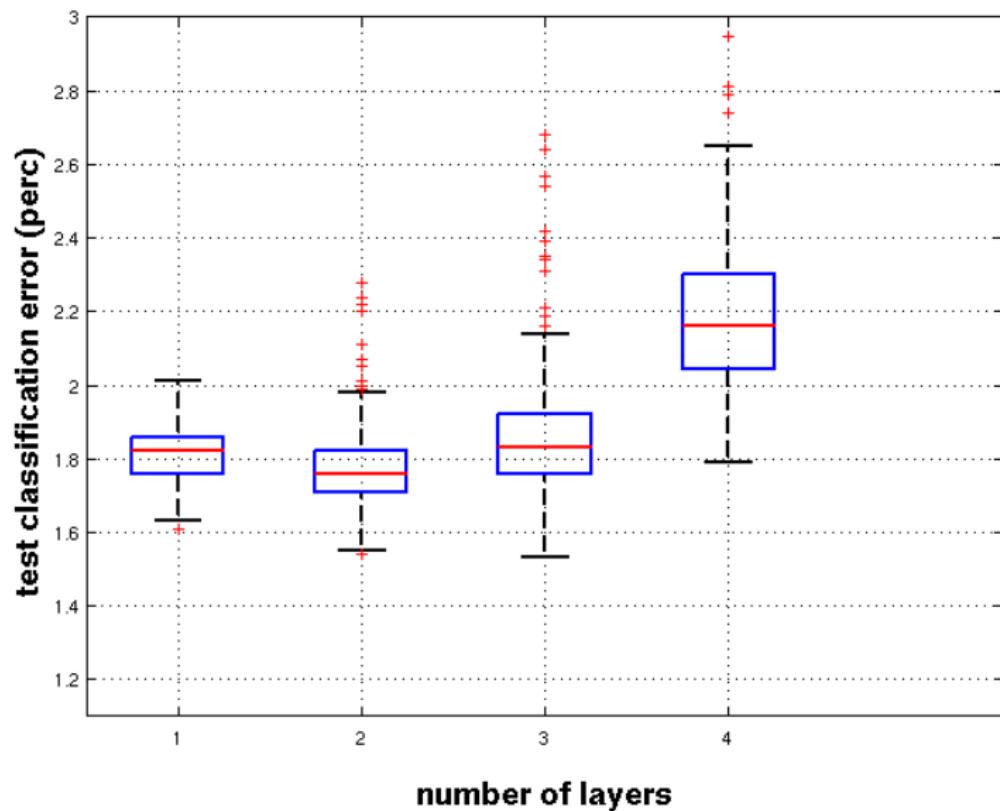
Da mesma maneira:

$$\frac{\partial C(\omega)}{\partial \omega_{ij}} = \delta_j x_i$$

Histórico de Deep Learning (cont.)

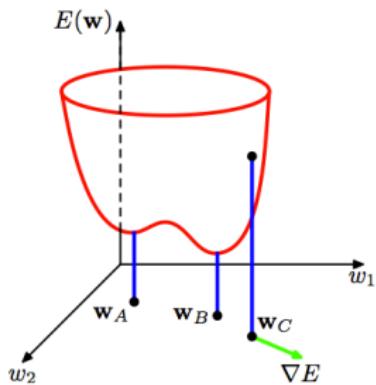
- As atualizações dos pesos seguem um formato **padrão**: $\delta_k h_j$ (**Erro Escalado * Entrada da Camada**)
- Ou seja, se o erro das camadas mais próximas à saída convergir, as camadas mais próximas a entrada não são mais treinadas.
- **Efeito Primário: Perda de generalização!!!**

Histórico de Deep Learning (cont.)



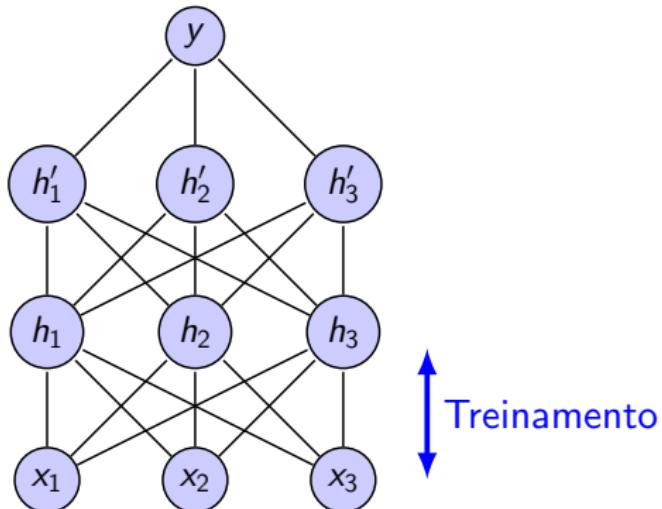
Deep Belief Networks

- Para NN com duas ou mais camadas, a função custo é não-convexa (**contínua e com, possivelmente, vários mínimos globais**)
- Ou seja, diferentes mínimos podem ser obtidos com diferentes inicializações
- Além disso, temos o problema do treinamento para multiplas camadas



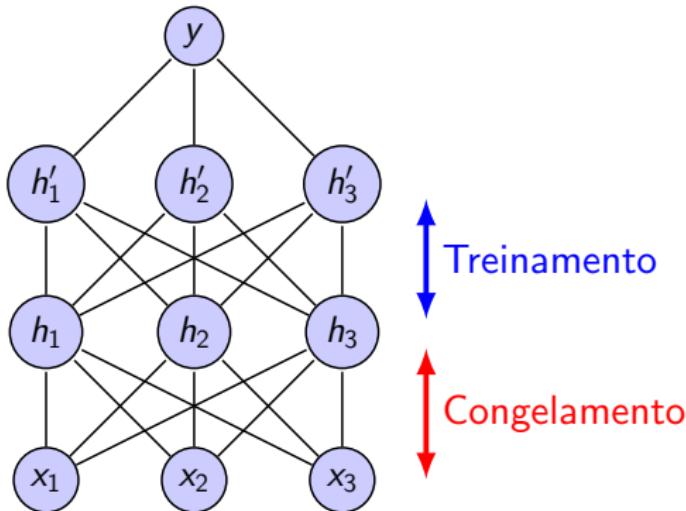
Deep Belief Networks

- Ou seja, sem um algoritmo bom para inicializar os pesos, nada de treinamento eficiente para mais de duas camadas
- Em 2006 → *Layer-Wise Pretraining* [19]
- *Layer-Wise Pretraining* é um algoritmo de treinamento não-supervisionado de **inicialização de pesos**



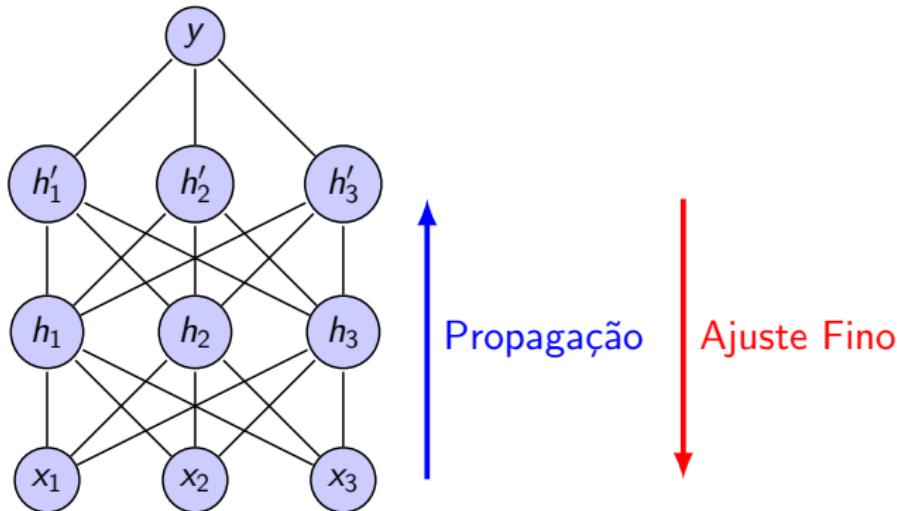
Deep Belief Networks

- Ou seja, sem um algoritmo bom para inicializar os pesos, nada de treinamento eficiente para mais de duas camadas
- Em 2006 → *Layer-Wise Pretraining* [19]
- *Layer-Wise Pretraining* é um algoritmo de treinamento não-supervisionado de **inicialização de pesos**



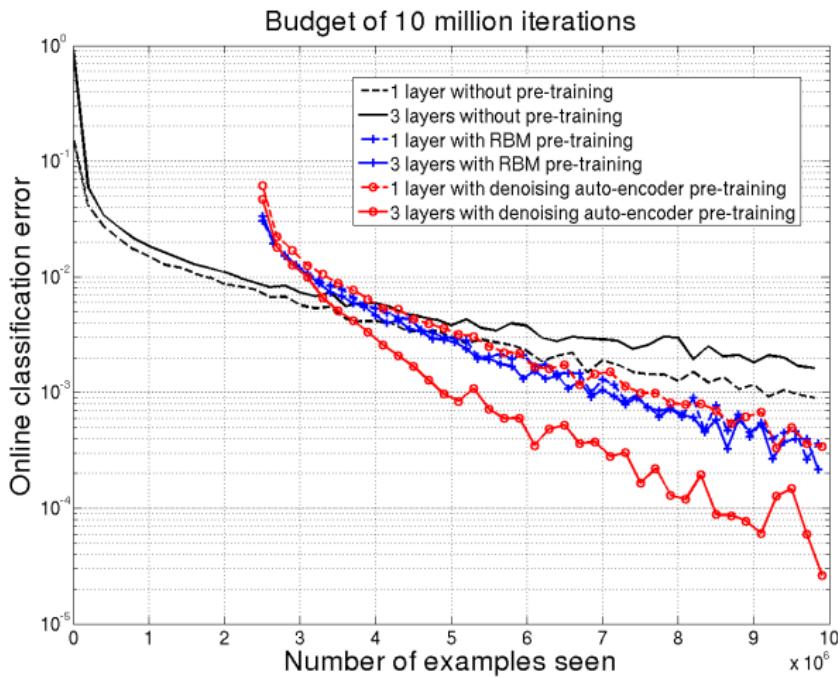
Deep Belief Networks

- Ou seja, sem um algoritmo bom para inicializar os pesos, nada de treinamento eficiente para mais de duas camadas
- Em 2006 → *Layer-Wise Pretraining* [19]
- *Layer-Wise Pretraining* é um algoritmo de treinamento não-supervisionado de **inicialização de pesos**



Deep Belief Networks

- Em [20] temos uma demonstração **experimental** de que o *Layer-Wise Pretraining* funciona



Deep Belief Networks

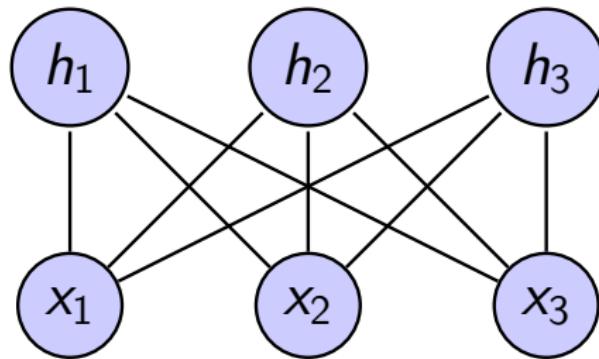
- Mudança de Paradigma: “Se você quer classificar imagens, não ensine o computador a classificar imagens. Ensine-o a enxergar” - Geoff Hinton
- Novo foco: Inicialmente, vamos nos focar em representar $P(X)$ melhor a cada uma das camadas e depois nos preocupamos em otimizar a $P(Y|X)$
- Mas como? Para DBN → RBM (*Restricted Boltzmann Machine*)

Restricted Boltzmann Machine

- Modelo baseado em energia

$$p(x|h) = \frac{1}{Z_\theta} e^{-E_\theta(x,h)}$$

- Onde $E_\theta(x, h) = -x^\top Wh - b^\top x - d^\top h$
- e $Z_\theta = \sum_{(x,h)} e^{-E_\theta(x,h)}$



Restricted Boltzmann Machine

- Podemos facilmente provar que

$$p(x|h) = \prod_i p(x_i|h)$$

- e de maneira similar:

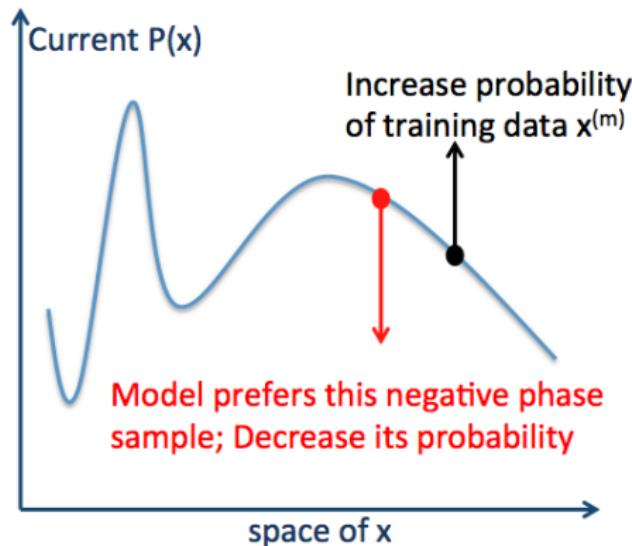
$$p(h|x) = \prod_j p(h_j|x)$$

- Ou seja, podemos realizar o cálculo de $\frac{\partial P(X)}{\partial W_{ij}}$
- E podemos observar que

$$p(h_j = 1|x) = \exp\left(x^\top W_j + d_j\right) / Z = \sigma\left(x^\top W_j + d_j\right)$$

Restricted Boltzmann Machine

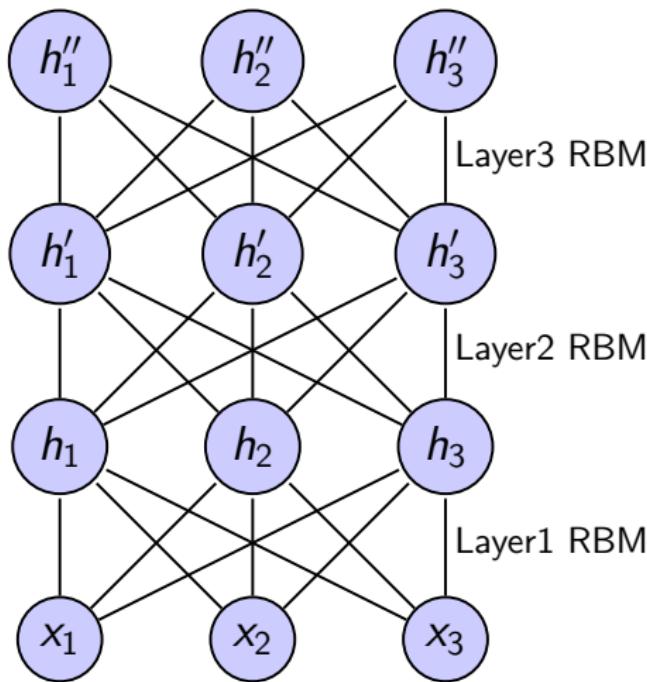
- Objetivo do treinamento: Representar muito bem $P(X)$ [21]
- Como? **Maximizando** $P(X)$ de acordo com os dados



- Esse treinamento **custa caro** computacionalmente. Solução:
Contrastive Divergence

Deep Belief Networks

Deep Belief Networks = RBM em cascata

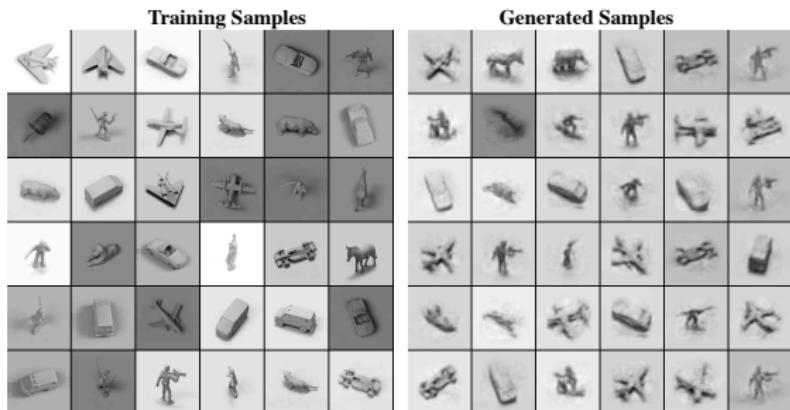
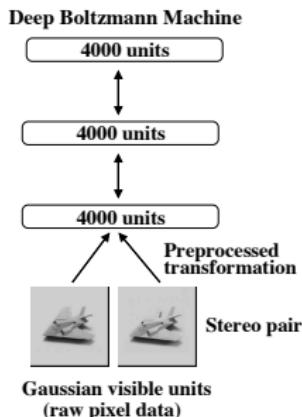


- Temos os pesos no sentido direto e no sentido reverso
- DBN = modelo não-supervisionado de estimativação de PDF $P(X)$.
- DBN = representação dos dados com máxima verossimilhança camada a camada
- DBN = modelo gerador de dados

Deep Belief Networks

Deep Belief Networks como modelo gerador de Dados

- Em [22] temos a utilização de DBM para gerar dados compatíveis com $P(X)$ treinado
- Treinamento com 20k imagens e geração de dados com dimensão de 8976 pixels



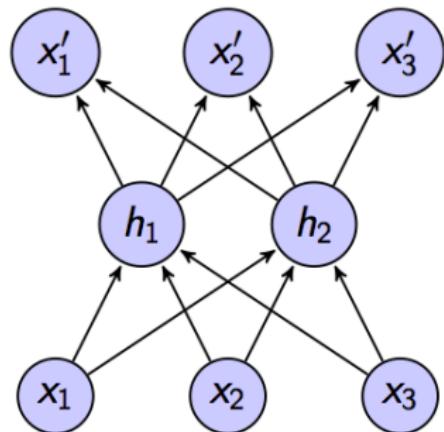
Coisas a serem lembradas até aqui...

- O algoritmo *Layer-wise pre-training* **reacendeu** o interesse por Deep Learning
- O pre-treinamento visa representar melhor os dados
- Por que RBM? Porque $p(h|x)$ é fácil de calcular, então é fácil cascatear várias RBM.
- Como o treinamento pode custar caro computacionalmente, recomenda-se a utilização da **Constrative Divergence**
- DBN formadas pelo cascamente de RBM são modelos geradores de dados

Stacked Auto-Encoders

Uma ideia antiga que se fez nova...

- Em [23], uma nova velha ideia surge que já foi vista em [24]



- Encoder

$$h = \sigma(Wx + b)$$

- Decoder

$$x' = \sigma(W'h + d)$$

- Reconstrução:

$$x'_m = \text{Decoder}(\text{Encoder}(x_m))$$

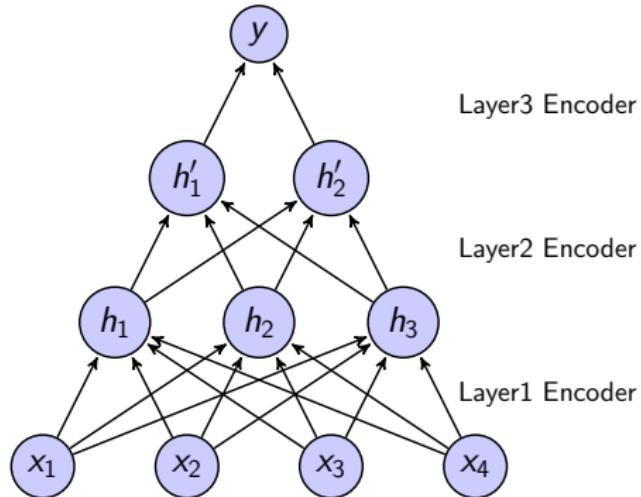
Stacked Auto-Encoders (cont.)

- Função Custo:

$$C = \sum_m \|x_m - x'_m\|$$

- Obs: Pode ser feito com um treinamento backpropagation e uma rede de duas camadas
- Pode ser implementado por: Redes Neurais Rasas (MLP) [24] ou por RBM [23] (**pouquíssimo** utilizada)

Stacked Auto-Encoders (cont.)



- Se implementado com MLP, temos um modelo determinístico

$$h = \sigma(Wx + b)$$

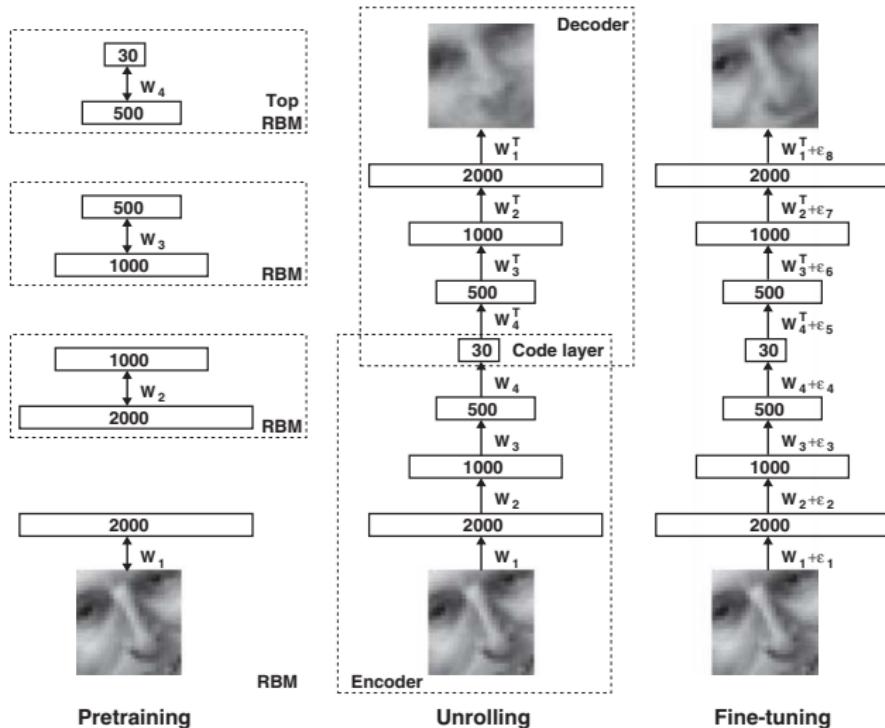
E não mais

$$p(h = 1) = \sigma(Wx + b)$$

Stacked Auto-Encoders (cont.)

- Menos custoso para treinar (não opera para estimar $P(X)$)
- Não gera um modelo gerador de dados (no caso da MLP fácil de verificar)

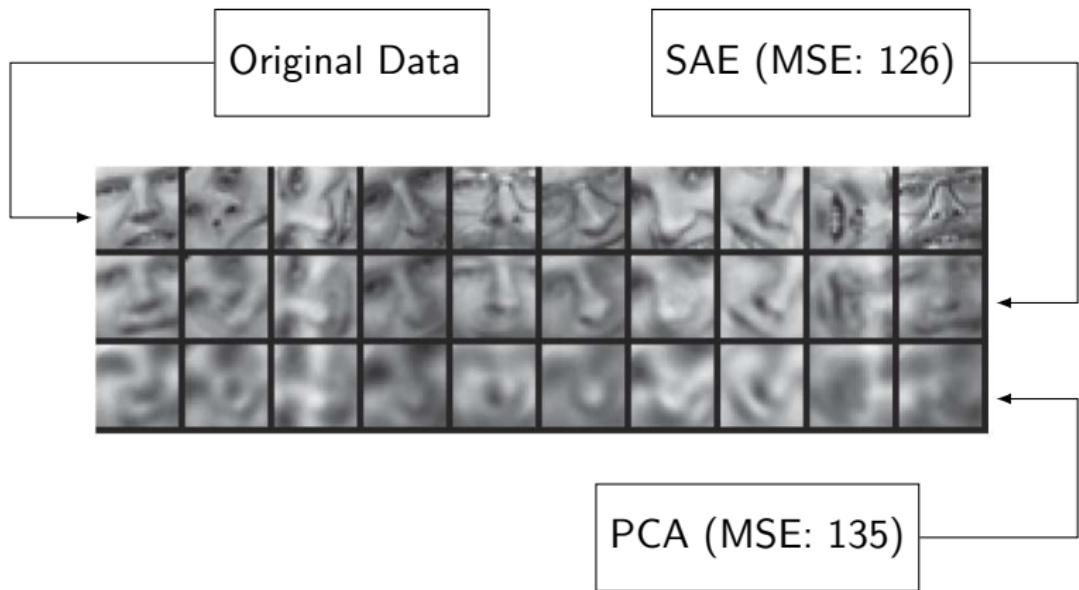
Stacked Auto-Encoders (cont.)



Stacked Auto-Encoders (cont.)

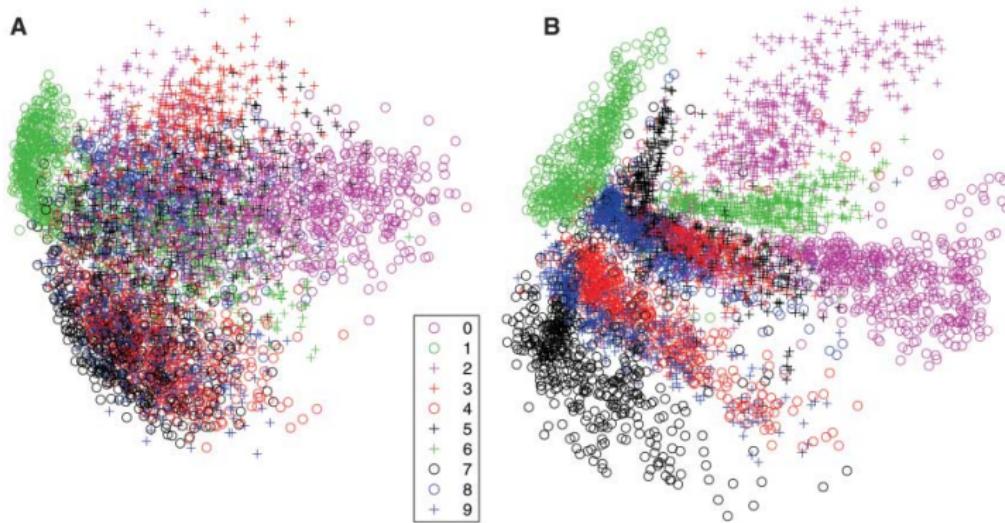


Stacked Auto-Encoders (cont.)



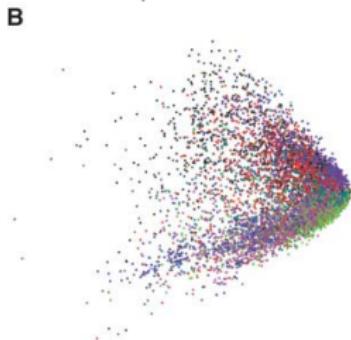
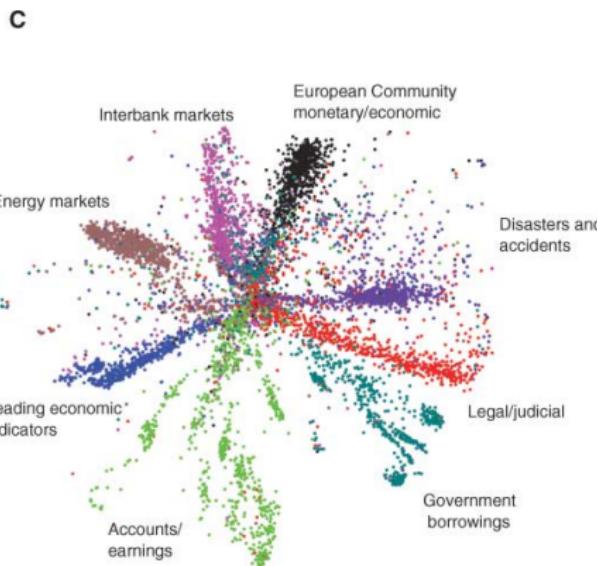
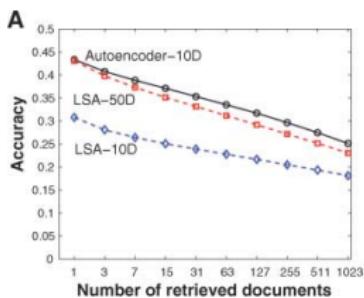
Stacked Auto-Encoders (cont.)

- Comparando PCA (2 primeiras) com SAE (1000-500-250-2)



Stacked Auto-Encoders (cont.)

- Comparando LSA (2 primeiras) [25] com SAE (2000-500-250-125-2)



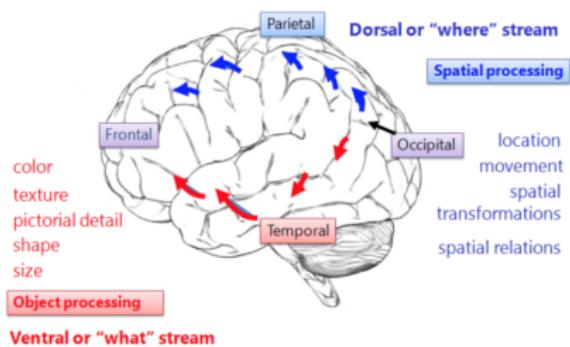
Stacked Auto-Encoders (cont.)

Variações interessantes

- *Denoising AutoEncoders*
- *Sparse AutoEncoders*

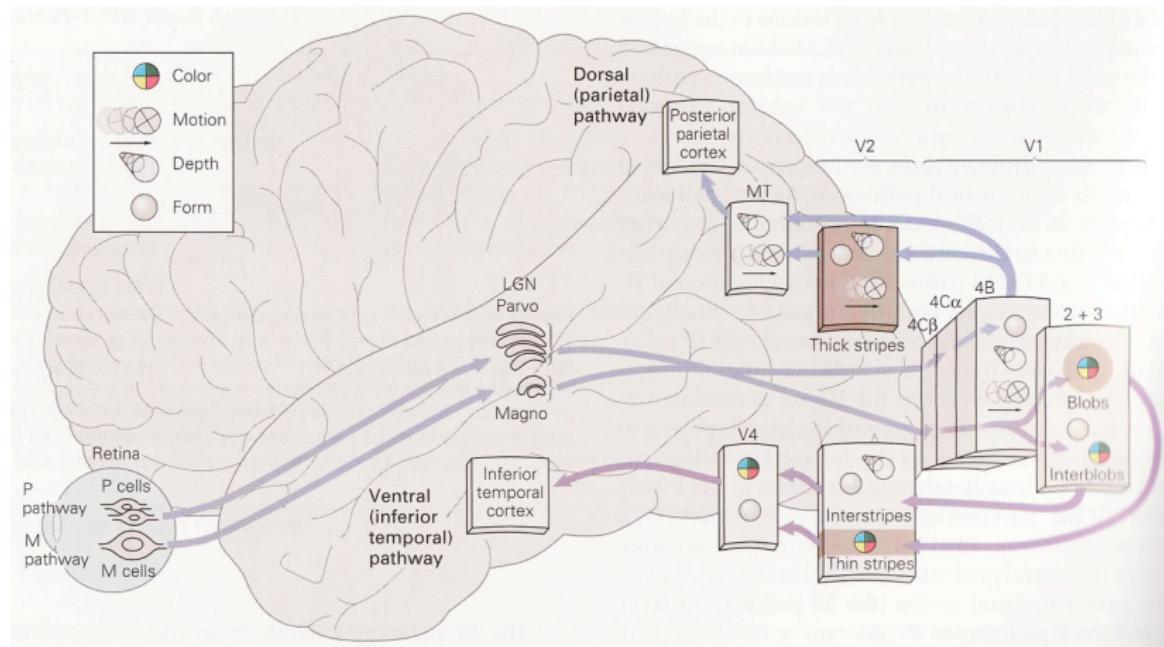
Convolutional Neural Networks - CNN, ConvNets

A proposta...



Convolutional Neural Networks - CNN, ConvNets

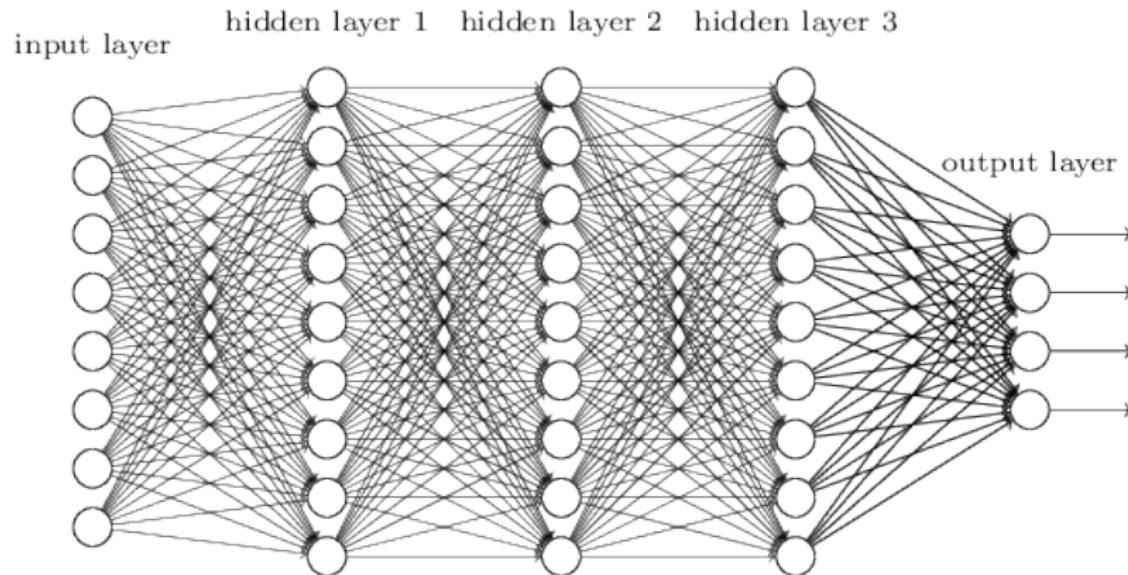
A proposta: Cada um com seu cada um...[26]



Redes Neurais Convolucionais

- Inspiradas no comportamento do Cortex Visual [27]
- É uma rede neural do tipo *Feed-Forward*
- Visa a utilização **mínima** de **pré-processamento**
- Tem por objetivo a extração de uma representação de padrões em alto nível [28]
- Explora características espaciais das entradas (+ distante na imagem → perda de estacionaridade)
- Em uma rede totalmente conectada cada uma das entradas se relacionada igualmente com a primeira camada escondida.

Convolutional Neural Networks - CNN, ConvNets



Blocos de uma CNN

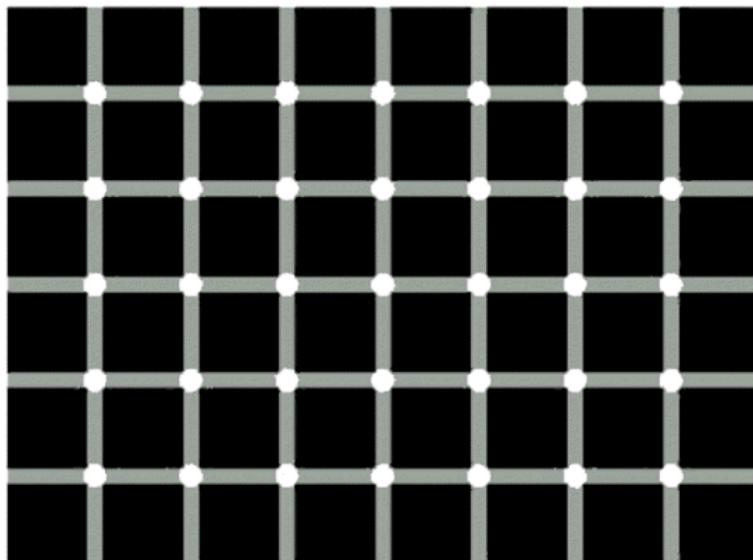
- ① Camada Convolucional - *Convolutional layer*
- ② Camada de Redução - *Pooling layer*
- ③ Camada Linear para Retificação - *ReLU layer*
- ④ Camada de Classificação - *Fully connected layer*

Convolutional Neural Networks - CNN, ConvNets

- Camada Convolucional - *Convolutional layer*
 - O que é uma convolução?

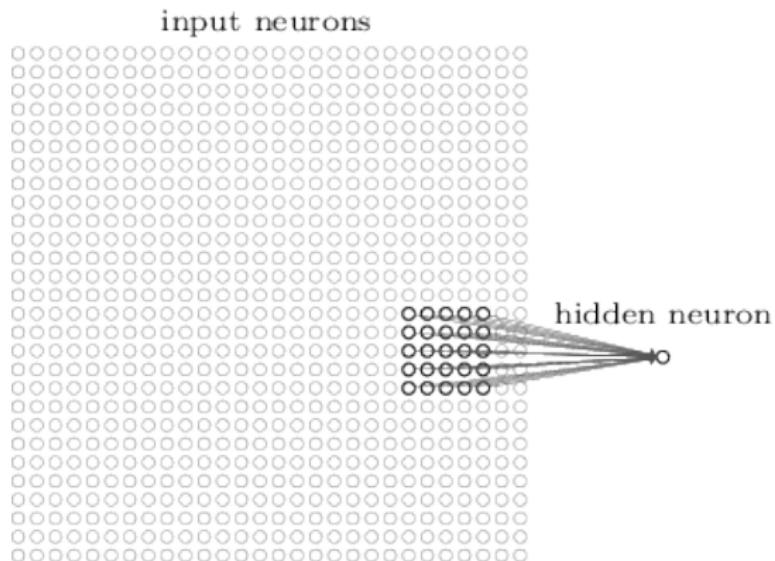
Convolutional Neural Networks - CNN, ConvNets (cont.)

- E por que isso nos ajuda? Em imagens, geralmente, informações espaciais são **importantes!!!**



Convolutional Neural Networks - CNN, ConvNets (cont.)

- Como fazer isso?



Convolutional Neural Networks - CNN, ConvNets (cont.)

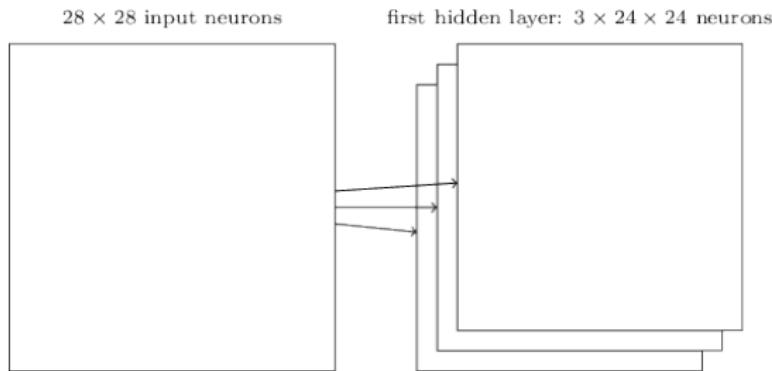
- A região acessada por cada neurônio da camada escondida é chamada *local receptive field* [29]
- Supondo in_{xy} como sendo a entrada na posição (x,y) , então a saída do neurônio pode ser descrita como:

$$out_{ij} = \sigma \left(\sum_{k=0}^{\mathcal{K}} \sum_{l=0}^{\mathcal{L}} \omega_{kl} in_{(i+k)(j+l)} \right)$$

- Onde \mathcal{K} e \mathcal{L} são os tamanhos da janela de convolução aplicados a entrada
- A primeira camada inteira compartilha o peso do neurônio e o bias (por isso essa rede tem os chamados **shared weights** e **shared bias**)
- Todos os neurônios da primeira camada escondida tem acesso a **exatamente a mesma característica** apenas deslocada
- Esse fato faz com que a primeira camada seja chamada de Mapa de Características (*Feature Map*)
- O conjunto **shared weights** e **shared bias** é chamado de filtro ou *kernel*

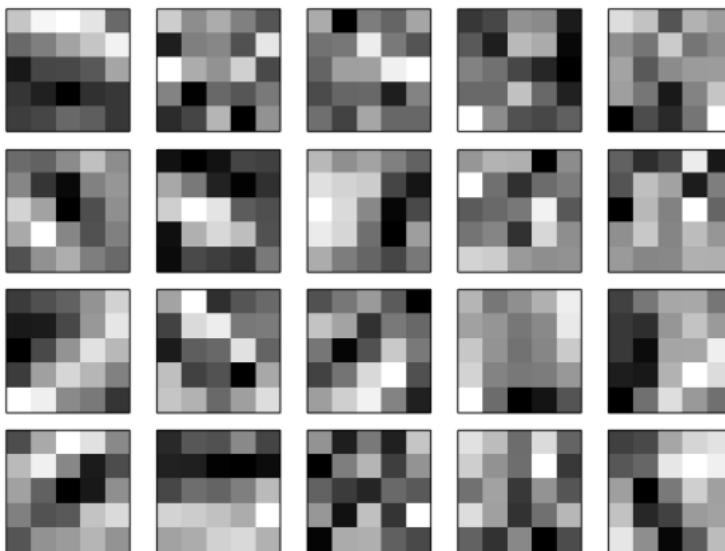
Convolutional Neural Networks - CNN, ConvNets (cont.)

- Podemos extrair quantos *Feature Maps* forem necessários, neste caso são 3 (que podem estar associados a diferentes tamanhos de filtros)



Convolutional Neural Networks - CNN, ConvNets (cont.)

- Em [30] temos 20 *Feature Maps* cada um com seu *local receptive field* de 5x5
- Cada característica aprendedida pode ser vista através da representação do seu peso



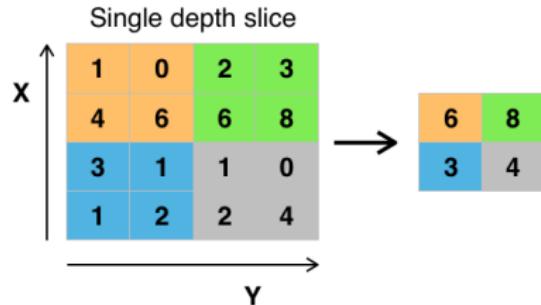
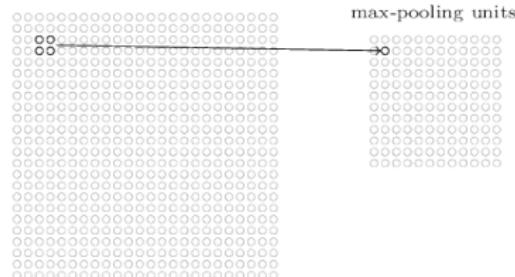
- Com essa abordagem, a quantidade de parâmetros em comparação com a MLP diminui drasticamente.
- Na MLP (para uma entrada 28×28) temos: $28 \times 28 = 784$ entradas com 30 neurônios (por exemplo) = 23.550 parâmetros
- Na CNN (para uma entrada 28×28) com cada *local receptive field* de 5×5 e 20 *Feature Maps* = 520 parâmetros
- Ou seja, uma redução de 40 vezes comparativamente.

Convolutional Neural Networks - CNN, ConvNets (cont.)

- Camada de Redução - *Pooling layer*

- Logo depois da Camada de Convolução, temos, geralmente, uma camada de Redução
- Essa camada é responsável por **simplicar a informação**
- Esta pega a informação de um *Feature Map* e cria um *Condensed Feature Maps*
- Existem outras funções de *pooling*, mas a mais comum é a *max pooling*

hidden neurons (output from feature map)



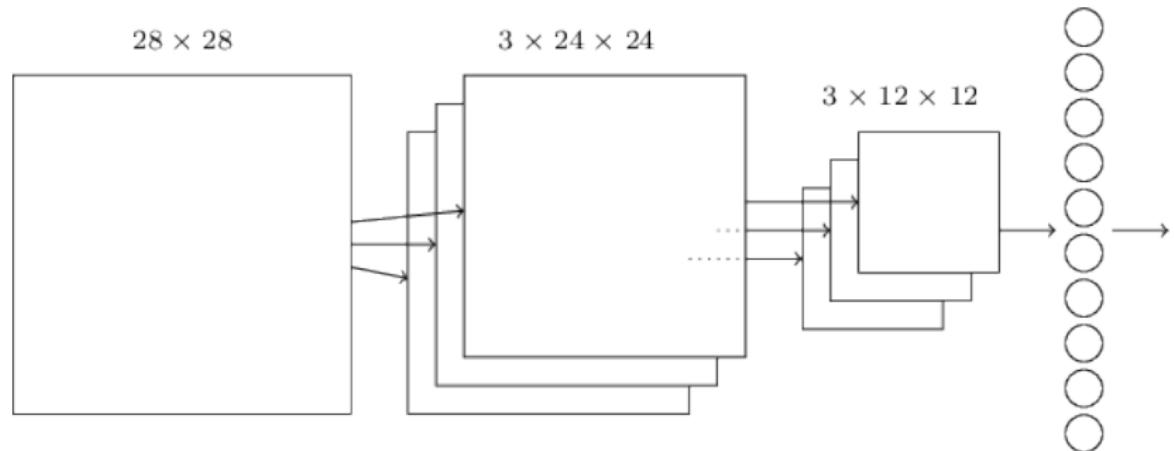
- Camada Linear para Retificação - *ReLU layer*

- É utilizada para retificar o sinal para entrada na camada de classificação (pois a mesma opera com uma função não linear limitada)
- Geralmente a função utilizada para retificação é

$$f(z) = \max(0, z)$$

Convolutional Neural Networks - CNN, ConvNets (cont.)

- Camada de Classificação - *Fully connected layer*
- É a camada para a tomada de decisões e é totalmente conectada as camadas anteriores
- Geralmente, utiliza **alvos maximamente esparsos** para a classificação



Processo de Treinamento de uma Quimera → Backpropagation

- Camada de Classificação = Backpropagation Clássico
- Camada de Retificação = Não é treinada...
- Camada de Pooling = Como não temos superposição durante a redução podemos mostrar que

$$\frac{\partial C(\omega_{ij})}{\partial \omega_{ij}} = \delta * x$$

Ou o Backpropagation Clássico

- Camada de Convolução = Backpropagation Clássico (pelo mesmo princípio)

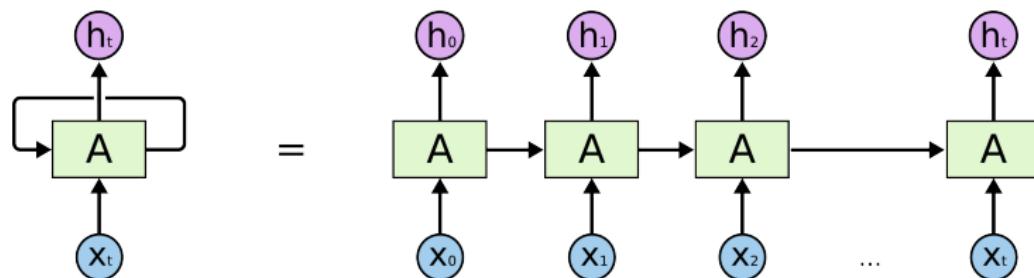
Técnicas de *Dropout*, Regularização e outras maneiras de evitar o *Overfit* devem ser utilizadas.

- Nas redes neurais vistas até aqui: Uma entrada gera uma saída e todas as entradas são **independentes** umas das outras
- Seres humanos não pensam em tudo do nada a todo segundo
- RNN se baseiam na idéia que pode haver uma **relação** entre as entradas
- Exemplo: Se você quiser prever uma sequência de letras, todo o vocabulário deve estar na sua rede.

De aorcdo com uma peqsiusa de
uma uinrvesriddae ignlse, não
ipomtra em qaul odrem as lteras de
uma plravaa etāso, a uncia csioa
iprotmatne é que a piremria e
útmlia lteras etejasm no lgaur crteo.

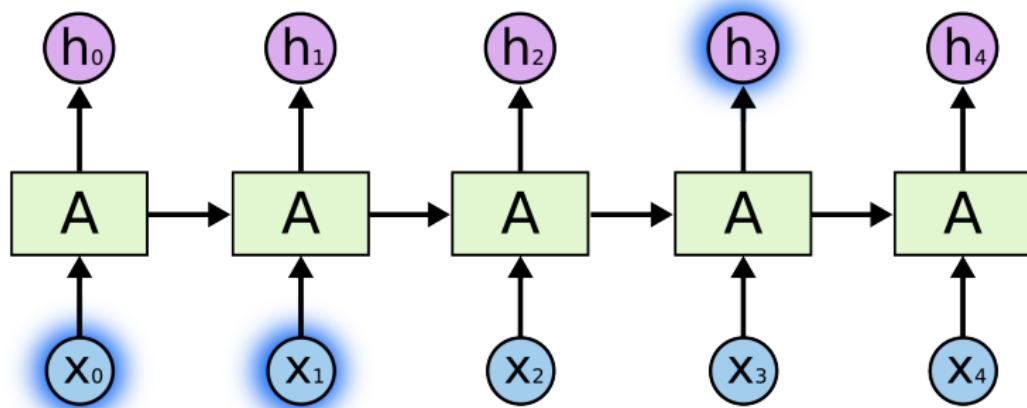
Recursive Neural Networks (cont.)

- RNN possuem *loops* internos, ou seja, uma informação anterior pode “navegar” na rede por mais tempo do que uma entrada (como acontece nas outras estruturas)



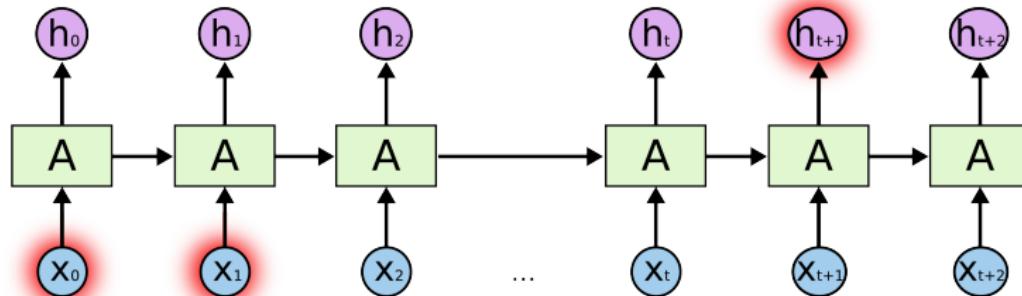
Recursive Neural Networks (cont.)

- É tudo uma questão de contexto...
- Complete a frase: As nuvens estão no ...



Recursive Neural Networks (cont.)

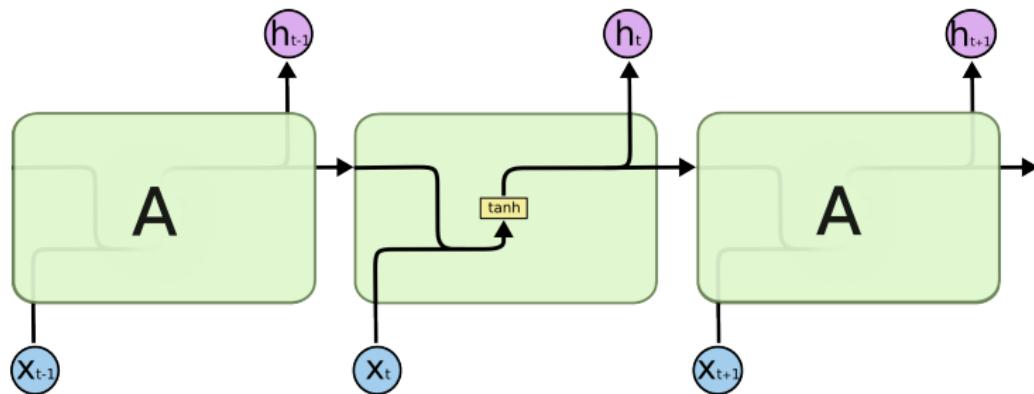
- É tudo uma questão de contexto...
- Complete a frase: Eu sou suíço e por isso, falo fluentemente ...



- Teoricamente, uma RNN poderia trabalhar facilmente com as chamadas dependências de tempo longo (*long-term dependencies*)
- Na prática, em [31] fica demonstrado que o buraco é mais embaixo.
- Embora, em [32] temos a introdução de uma nova técnica chamada LSTM (*Long Short Term Memory*)

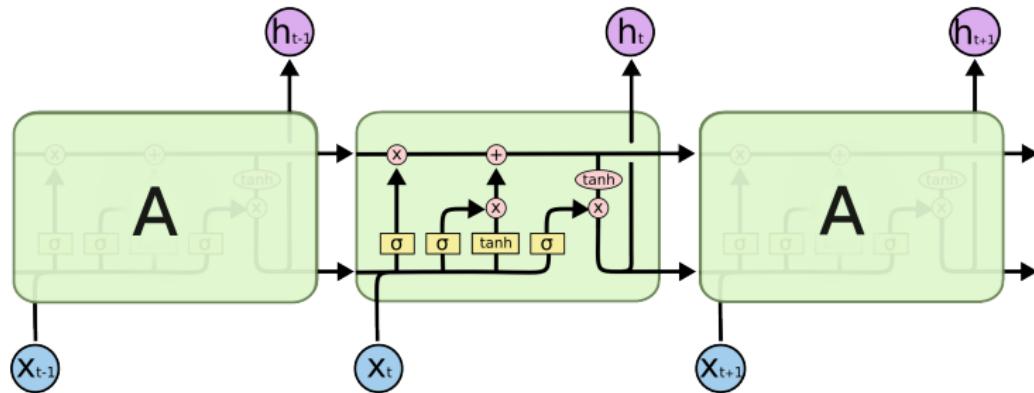
Recursive Neural Networks (cont.)

- Basicamente, quando se fala de RNN, se fala de LSTM (tamanho o sucesso da técnica)
- Por que isso? LSTM foram projetadas para não depender de memória de tempo longo.
- No modelo de uma RNN, temos:



Recursive Neural Networks (cont.)

- Já no caso de LSTM, temos:



O que é aprendizado não-supervisionado?

Definição de aprendizado não-supervisionado

Aprendizado não-supervisionado é uma das tarefas de aprendizado de máquina e é responsável por estimar uma **função que descreva a estrutura** que gerou os dados disponíveis. Fundamentalmente, os dados não possuem uma **catergorização**

Aplicações fundamentais:

- Clusterização
- Compactação de dados

O que é aprendizado não-supervisionado?

Técnicas mais famosas

- Para Clusterização:
 - **k-Means**
 - *Self-organizing Maps* - SOM
 - *Adaptive Resonance Theory* - ART
 - *Gaussian Mixture Models* - GMM
- Compactação de dados
 - *Principal Component Analysis* - PCA
 - *Independent Component Analysis* - ICA

Referências Bibliográficas

- [1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
- [2] F. Rosenblatt, "The perceptron, a perceiving and recognizing automaton: (project para)," report, Cornell Aeronautical Laboratory, 1957.
- [3] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*.
Spartan Books, 1962.
- [4] J. Mycielski, "Review: Marvin minsky and seymour papert, perceptrons, an introduction to computational geometry," *Bulletin of the American Mathematical Society*, vol. 78, pp. 12–15, 01 1972.

Referências Bibliográficas (cont.)

- [5] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1st edition ed., 1969.
- [6] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991.
- [7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1* (D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, eds.), pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.
- [8] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

Referências Bibliográficas (cont.)

- [9] S. Haykin, *Neural Networks: A Comprehensive Foundation.* MacMillan Publishing Company, 1994.
- [10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [11] J. Pearl, "Bayesian networks: A model of self-activated memory for evidential reasoning," in *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine*, pp. 329–334, Aug. 1985.
- [12] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, 2010.

Referências Bibliográficas (cont.)

- [13] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489 – 501, 2006.
- Neural Networks
Selected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN '04)
7th Brazilian Symposium on Neural Networks.
- [14] F.-H. Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*.
Princeton University Press, 2004.
- [15] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, July 2006.
- [16] D. A. Ferrucci, "Introduction to “this is watson”," *IBM Journal of Research and Development*, vol. 56, pp. 1:1–1:15, May 2012.

Referências Bibliográficas (cont.)

- [17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 609–616, ACM, 2009.
- [18] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training.", *AISTATS*, vol. 5, pp. 153–160, 2009.
- [19] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," tech. rep., Dept. IRO, Université de Montréal, 2006.

Referências Bibliográficas (cont.)

- [20] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.
- [21] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," in *Intelligence, Artificial and Statistics, 2005, Barbados*, 2005.
- [22] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 5, pp. 448–455, 2009.
- [23] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

Referências Bibliográficas (cont.)

- [24] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.
- [25] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman, "Using latent semantic analysis to improve access to textual information," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, (New York, NY, USA), pp. 281–285, ACM, 1988.
- [26] D. C. Van Essen and J. L. Gallant, "Neural mechanisms of form and motion processing in the primate visual system," *Neuron*, vol. 13, pp. 1–10, 2016/06/29 1994.
- [27] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *Journal of Physiology (London)*, 1968.

Referências Bibliográficas (cont.)

- [28] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
- [29] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *ArXiv e-prints*, 2013.
- [30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.
- [31] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Trans. Neur. Netw.*, vol. 5, pp. 157–166, Mar. 1994.
- [32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.