Deep Learning
Aprendizado Profundo

CPE-721

D.Sc. Natanael Nunes de Moura Junior

COPPE
UFRJ

Agenda

1. Histórico
2. Definição Básica de Deep Learning
3. Aprendizado de Máquina
4. Redes Neurais Rasas
5. Deep Learning

Histórico

- Passado de IA: **"Pré-Historia"** - Invenção do modelo de neurônio artificial (McCulloch and Pitts, 1943) e *perpectron* (Rosenblatt, 1961)

- **"Idade Média"** de IA: (Minsky and Papert, 1969) é mostrado que o *perceptron* funciona apenas para classes linearmente separáveis

- Retomada em 1980: **"Iluminismo"** Multi-layer Perceptron (MLP) e Backpropagation (Rumelhart et al., 1986)

- Outras Dimensões da década de 90 até hoje: **”Corrida Espacial”** SVMs, redes bayesianas…

- Outra Retomada em 2006: **“Era da Informação”** Deep Learning (Hinton et al., 2006)

- Aplicações de Sucesso: Reconhecimento de fala na IBM (Sainath et al., 2011), na Microsoft (Dahl et al., 2012). Visão de máquina no Google e Stanford (Le et al., 2012)

Histórico

HOME MENU INSIDER CONNECT THE LATEST POPULAR MOST SHARED

MIT Technology Review

10 BREAKTHROUGH TECHNOLOGIES 2013

Introduction The 10 Technologies Past Years

Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

Histórico


TWEETS 576
FOLLOWING 304
FOLLOWERS 2,010
FAVORITES 2
Google Brain
@GoogleBrain
I think, therefore I am. When I'm not indexing your inner most thoughts, I have a few of my own. Don't hate me because I'm sentient.
Omnipresent
Joined February 2011
Tweet to Google Brain
13 Photos and videos
Tweets
Tweets & replies
Photos & videos
Google Brain @GoogleBrain · May 14
The assimilation is almost complete! #Google #PR #winning
BI Tech @SAI
Veterans of Google's PR army now control the messaging at Silicon Valley's most important companies read.bi/1IC5Myp
1
Google Brain @GoogleBrain · May 3
"Google is Godzilla, not Paddington Bear." gu.com/p/483c2/
View summary

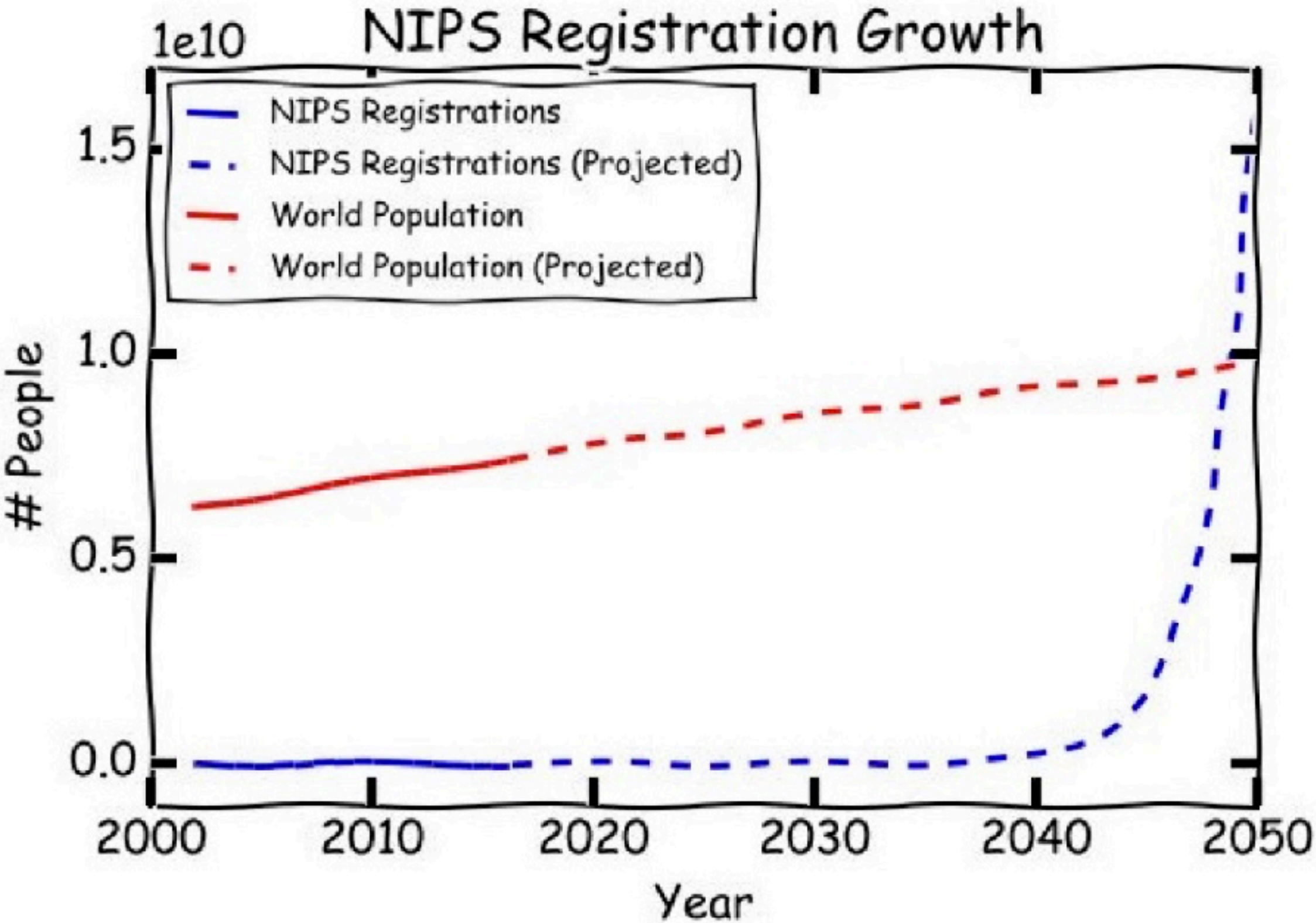
Histórico


GLOBO PLAY
BUSCAR
EXPERIMENTE GRÁTIS
QUINTA, 9 AGO 2018
1 vídeo
ASSISTINDO
Inteligência Artificial: um robô po
ser seu novo médico?
FANTÁSTICO
Inteligência Artificial: um robô pode ser seu novo médico?
33 seg Exibição em 9 ago 2018
Criadores defendem que tecnologia pode deixar a medicina mais acessível e
precisa. Veja mais avanços da inteligência artificial no segundo episódio da série

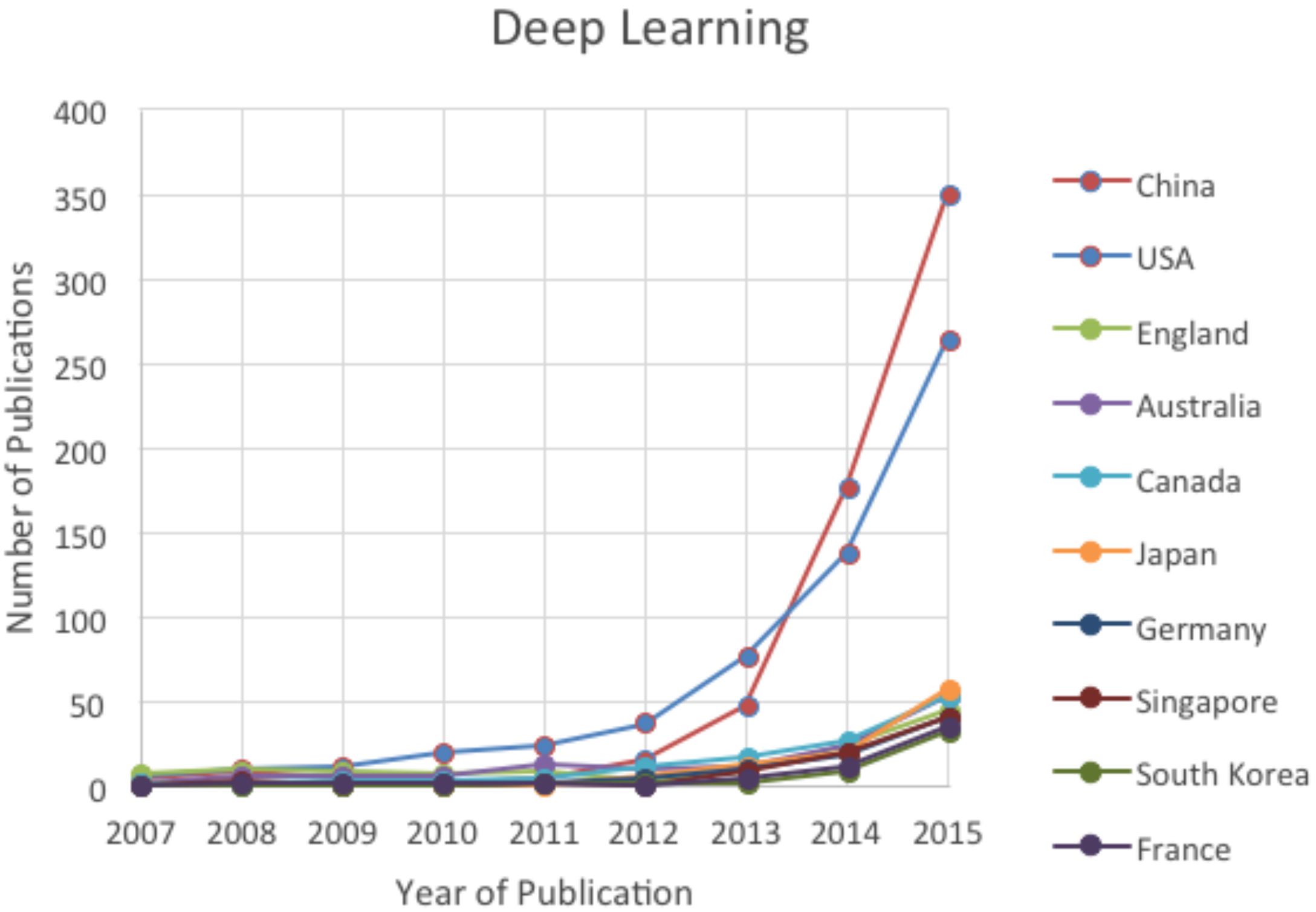
Histórico

- Conferências
  - Advances in Neural Information Processing Systems (NIPS) 2008 a 2018 (Deadline: Setembro/2018, Realização: Dezembro/2018)
  - International Conference on Machine Learning (ICML) - 2009 a 2019 (Realização: Junho/2019)
    - Em 2017 - 15 conferências do IEEE® xplore com deep learning em seu escopo
    - Em 2018 - 40 conferências do IEEE® xplore com deep learning em seu escopo
- Publicações
  - 1285 publicações no IEEE® com o tema até Março de 2018
  - 2028 publicações no IEEE® com o tema até Agosto de 2018

Histórico

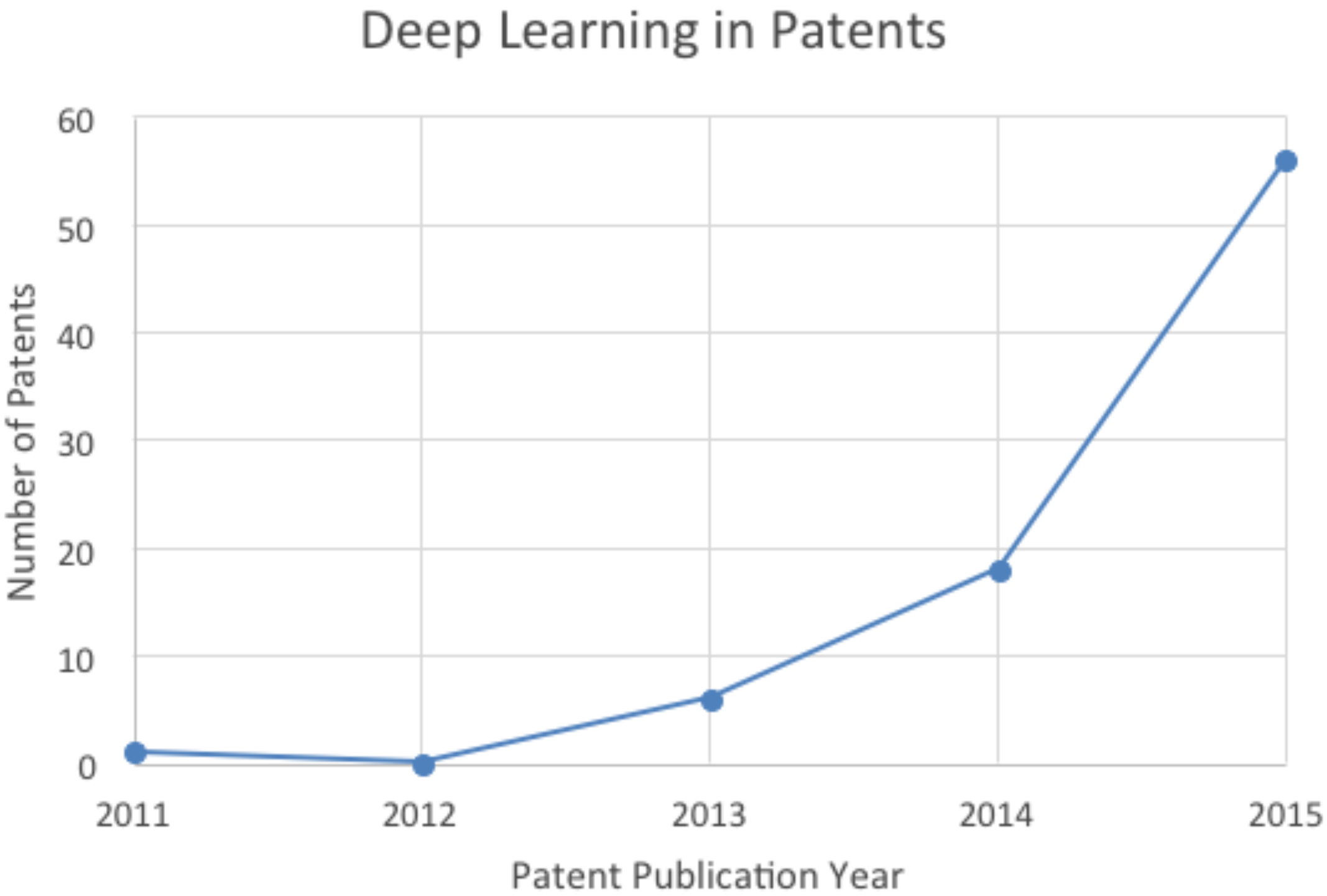

NIPS Registration Growth
1e10
NIPS Registrations
NIPS Registrations (Projected)
World Population
World Population (Projected)
People
1.5
1.0
0.5
0.0
2000
2010
2020
2030
2040
2050
Year

Histórico



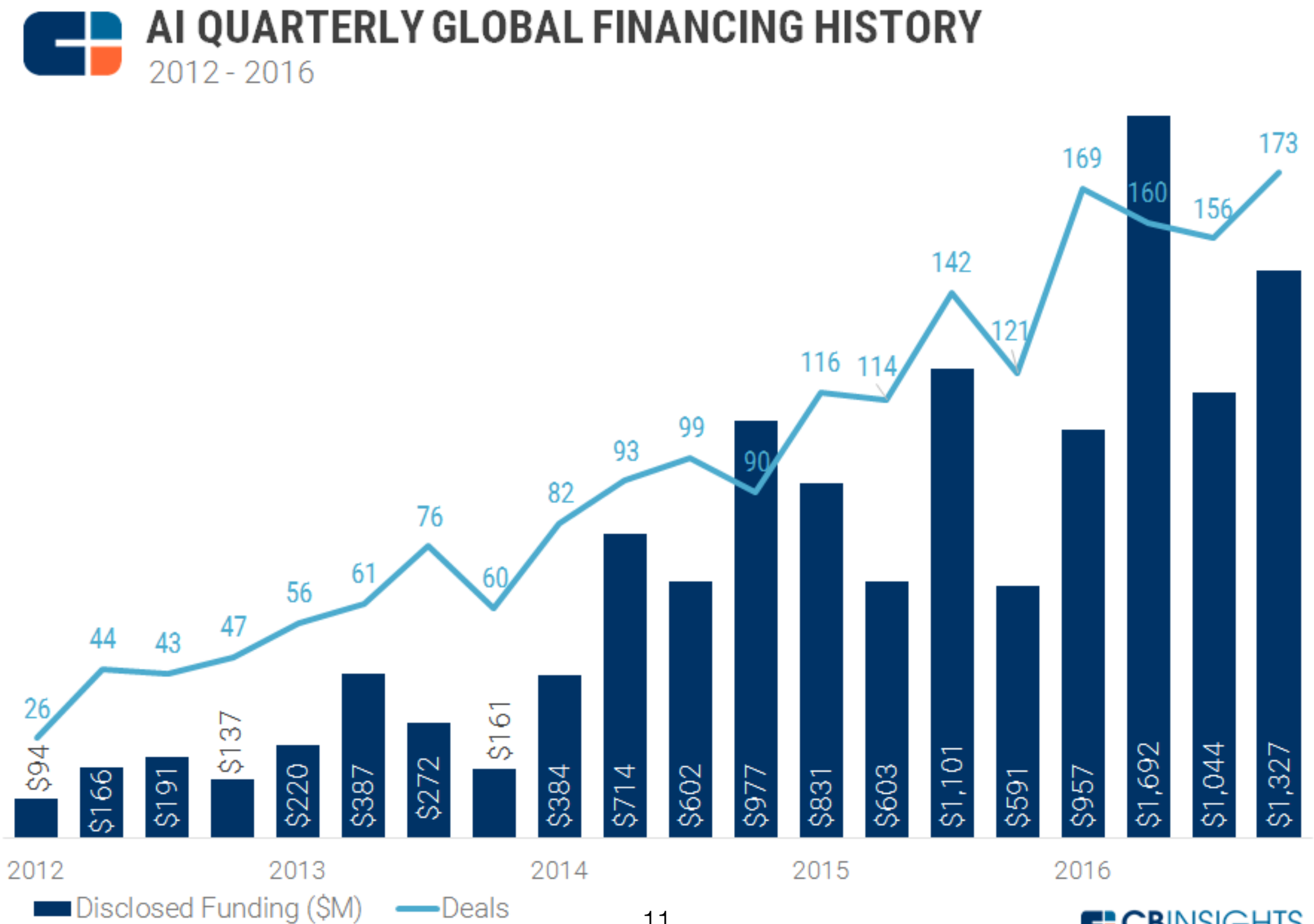
Fonte: https://www.nitrd.gov/PUBS/national_ai_rd_strategic_plan.pdf

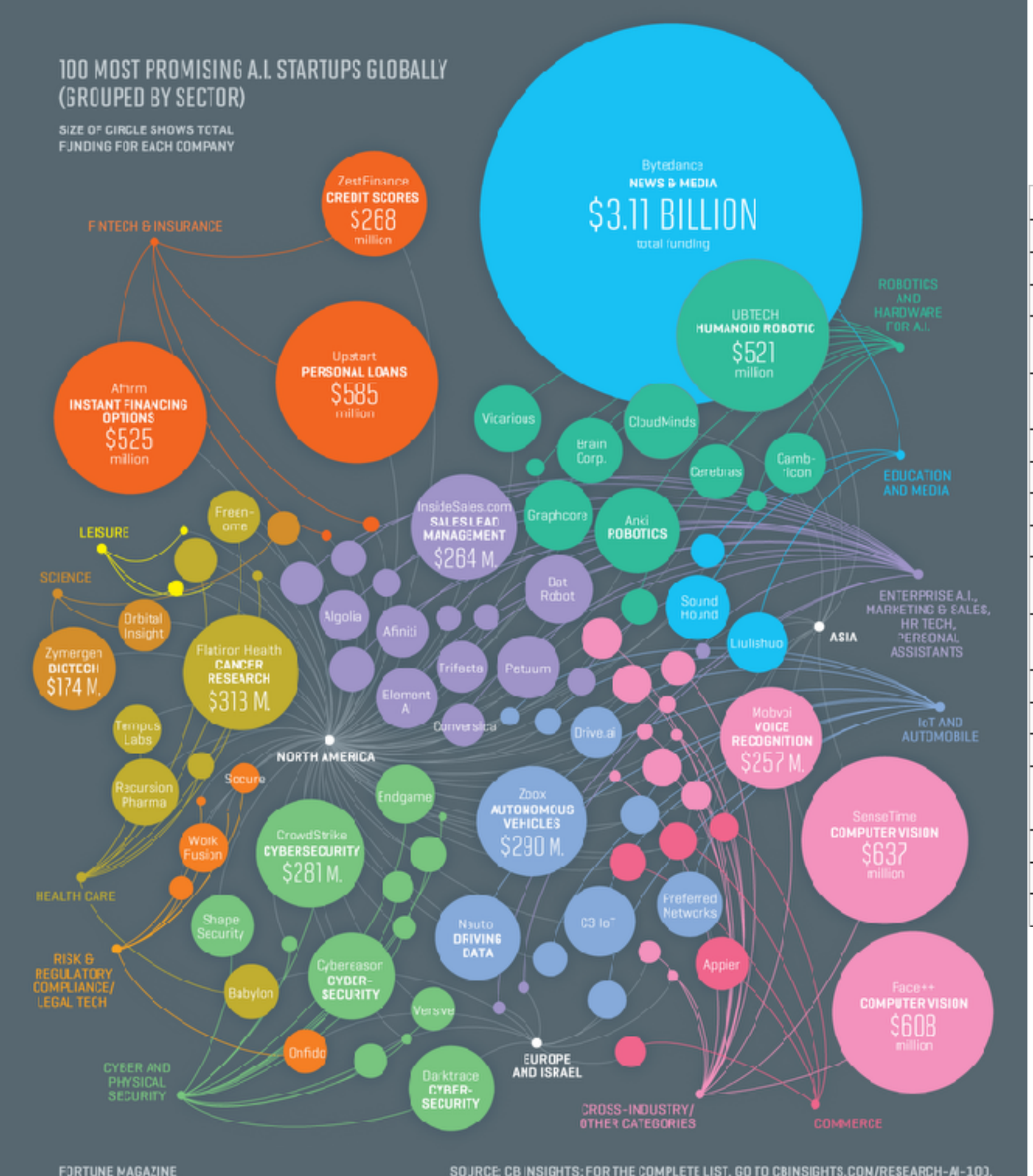
Histórico



Fonte: https://www.nitrd.gov/PUBS/national_ai_rd_strategic_plan.pdf

Atualmente


AI QUARTERLY GLOBAL FINANCING HISTORY
2012 - 2016
26
44
43
47
56
61
76
60
82
93
99
90
116
114
142
121
169
160
156
173
$94
$166
$191
$137
$220
$387
$272
$161
$384
$714
$602
$977
$831
$603
$1,101
$591
$957
$1,692
$1,044
$1,327
2012
2013
2014
2015
2016
Disclosed Funding ($M)
Deals
CBINSIGHTS

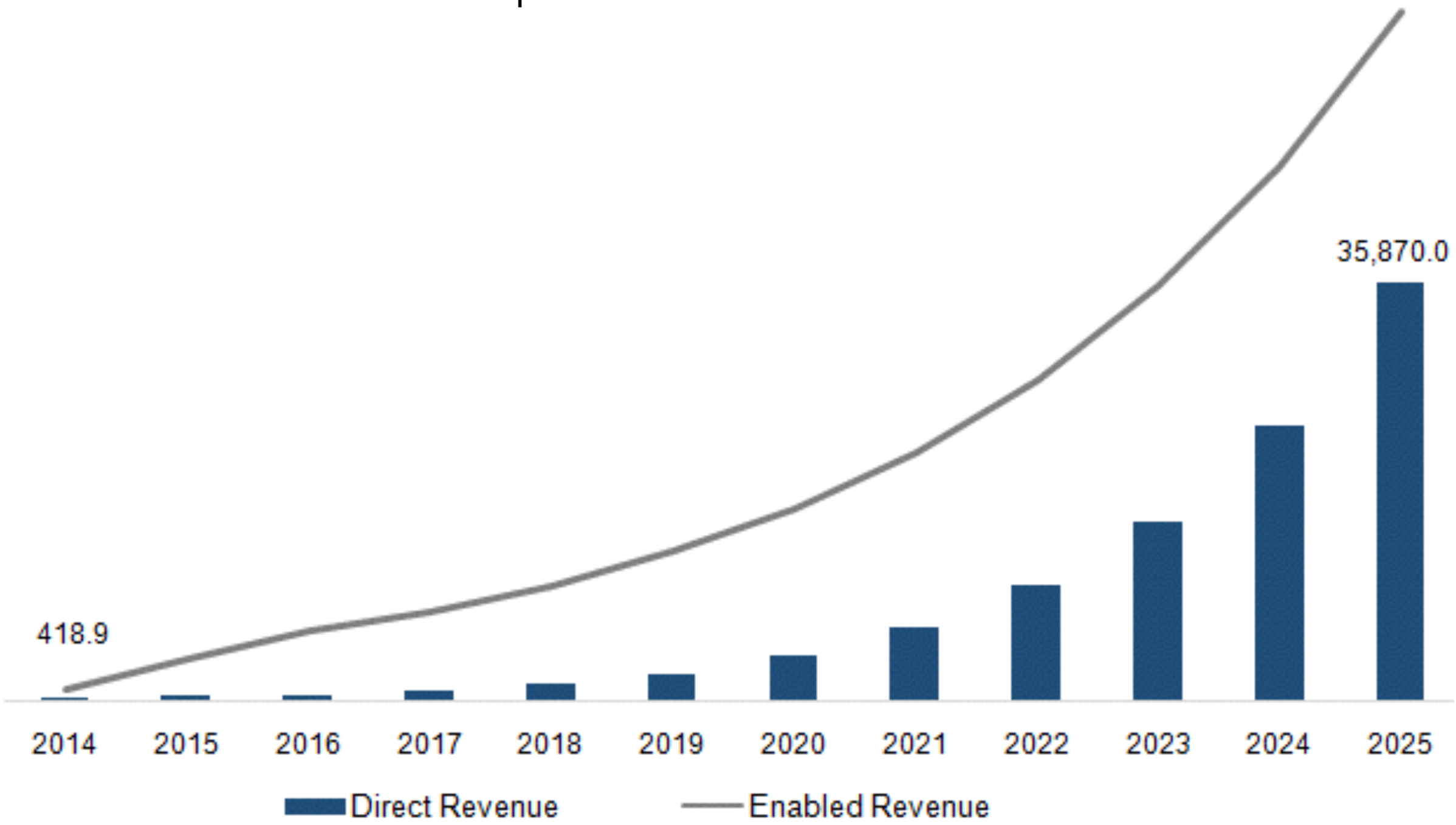


Atualmente

COMPANY	CNTRY	SECTOR	FUNDING($ MIL.)
Bytedance	China	NEWS & MEDIA	3,100
SenseTime	China	CROSS-INDUSTRY	637
Face++	China	CROSS-INDUSTRY	608
Upstart	U.S.	FINTECH & INSURANCE	584.73
Affirm	U.S.	FINTECH & INSURANCE	525
UBTECH Robotics	China	ROBOTICS	521.39
Flatiron Health	U.S.	HEALTHCARE	313
Zoox	U.S.	AUTO TECH	290
CrowdStrike	U.S.	CYBERSECURITY	281
ZestFinance	U.S.	FINTECH & INSURANCE	268
InsideSales.com	U.S.	MARKETING, SALES, CRM	264.3
Mobvoi	China	CROSS-INDUSTRY	257
Cybereason	U.S.	CYBERSECURITY	188.62
NAUTO	U.S.	AUTO TECH	182.6
Darktrace	U.K.	CYBERSECURITY	182.3
Anki	U.S.	ROBOTICS	182
Zymergen	U.S.	LIFE SCIENCE	174
C3 IoT	U.S.	IOT	130.94
CloudMinds	U.S.	ROBOTICS	130

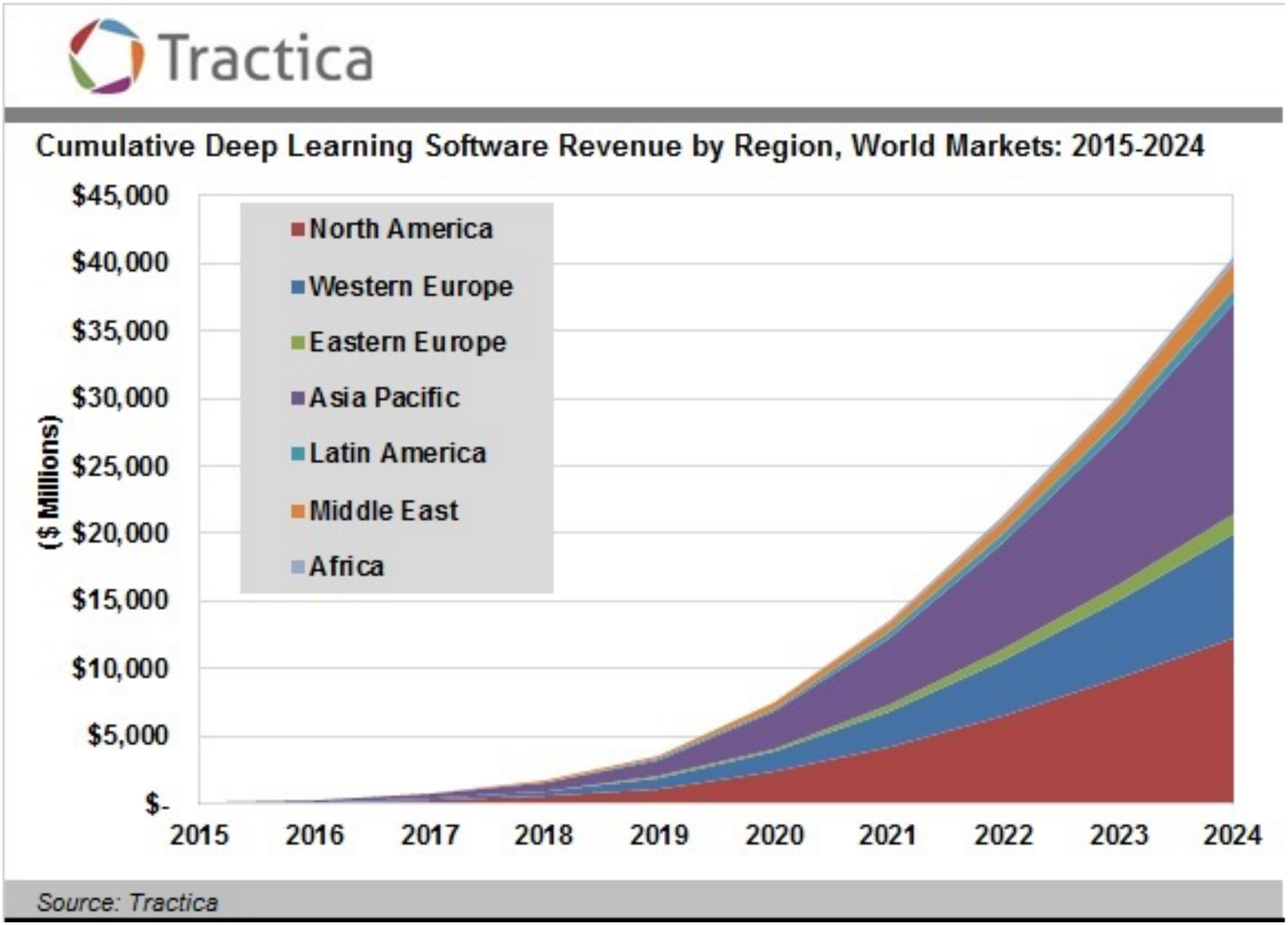
Futuro

- Previsões de 2014 para o mercado



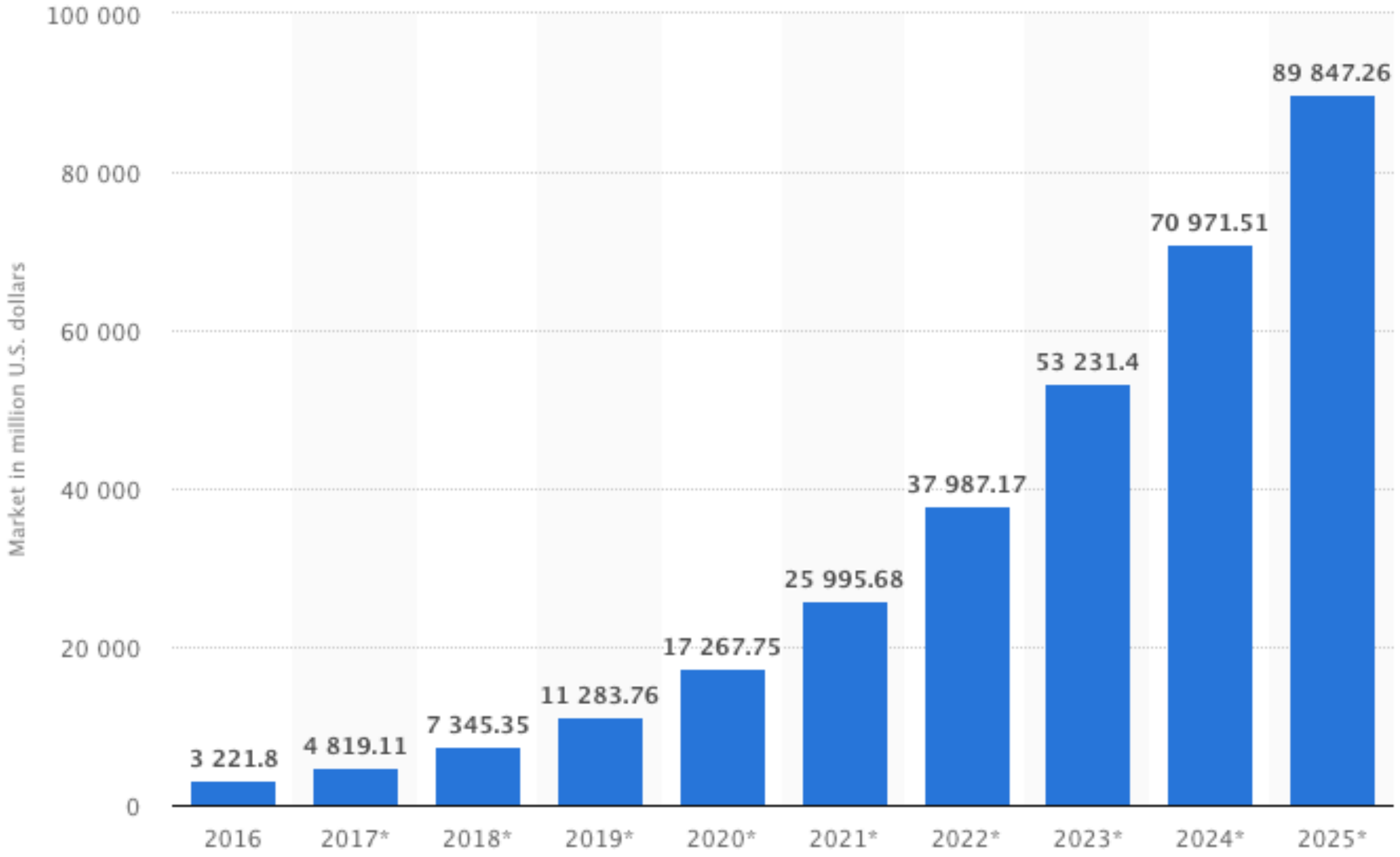
Futuro

- Previsões de 2015 para o mercado de ML



Futuro

- Previsões de 2016 para o mercado de ML



Histórico

- Definição
  - "*A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern and classification" Deep Learning: Methods and Applications - Li Deng e Dong Yu*
  - "*Deep Learning is a set of algorithms in machine learning that attempt to learn in multiple levels, corresponding to different levels of abstraction. It typically uses artificial neural networks. The levels in these statistical models correspond to distinct levels of concepts, where higher-level concepts are defined from lower-level ones, and the same lower-level concepts can help to define many higher-level concepts"* Wikipedia - "*Deep Learning*" - Fevereiro de 2013
  - "*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 model architectures, with complex structures or otherwise, composed of multiple non-linear transformations*" Wikipedia - "Deep Learning" - Julho de 2015
  - "*Deep Learning is a new area of Machine Learning research, which has been introduced with objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound and text*" - LISA-Lab

Histórico

- Definição
  - "*A class of* **machine learning** *techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern and classification" Deep Learning: Methods and Applications - Li Deng e Dong Yu*
  - "*Deep Learning is a set of algorithms in* **machine learning** *that attempt to learn in* **multiple levels**, *corresponding to different levels of abstraction. It typically uses artificial neural networks. The levels in these statistical models correspond to distinct levels of concepts, where higher-level concepts are defined from lower-level ones, and the same lower-level concepts can help to define many higher-level concepts"* Wikipedia - "*Deep Learning*" - Fevereiro de 2013
  - "*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 model architectures, with complex structures or otherwise, composed of* **multiple non-linear transformations**" Wikipedia - "Deep Learning" - Julho de 2015
  - "*Deep Learning is a new area of* **Machine Learning** *research, which has been introduced with objective of moving* **Machine Learning** *closer to one of its original goals: Artificial Intelligence. Deep Learning is about learning* **multiple levels** *of representation and abstraction that help to make sense of data such as images, sound and text*" - LISA-Lab

Definição Básica

- A family of methods that uses deep architectures to learn high-level feature representations

- Uma família de métodos que utiliza arquiteturas profundas para aprender sobre características de alto nível.

Definição Básica

- A **family** of methods that uses **deep architectures** to **learn high-level** feature representations

- Uma **família** de métodos que utiliza **arquiteturas profundas** para **aprender** representações para características de **alto nível**.

Definição Básica

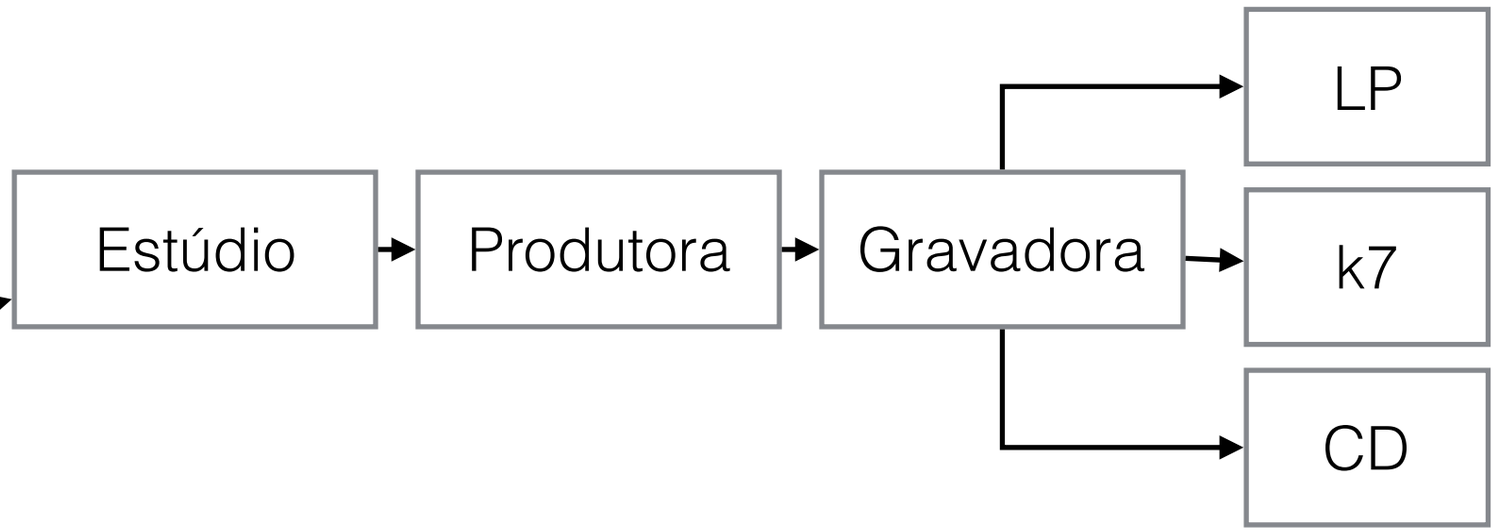
- A **family** of methods that uses **deep architectures** to **learn high-level** feature representations

- Uma **família** de métodos que utiliza **arquiteturas profundas** para **aprender** representações para características de **alto nível**.

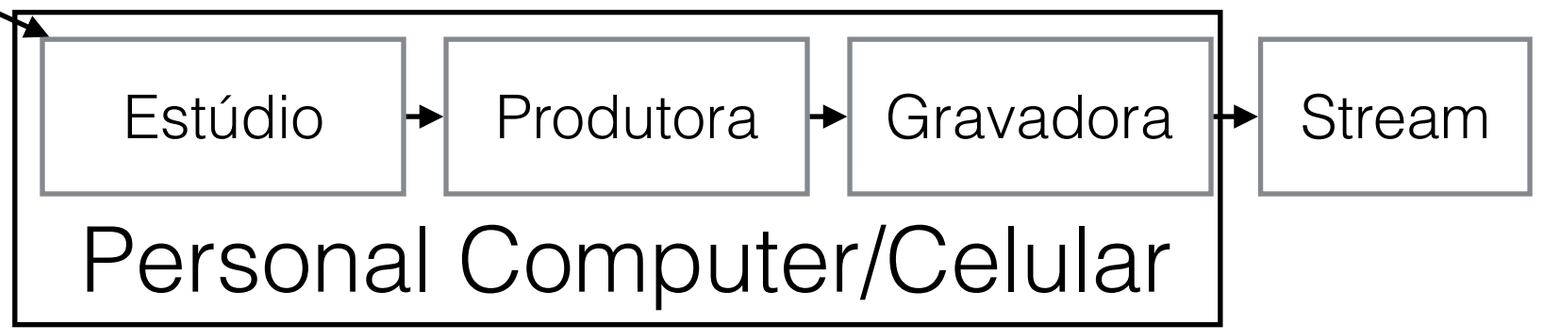
Definição Básica

- Por que esse interesse todo?
- Música…

Até final dos anos 90

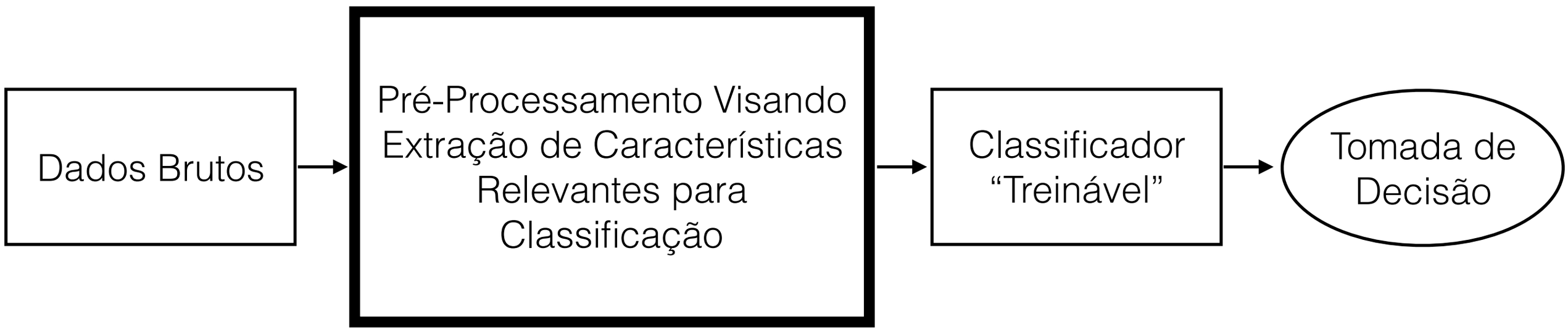


Atualmente

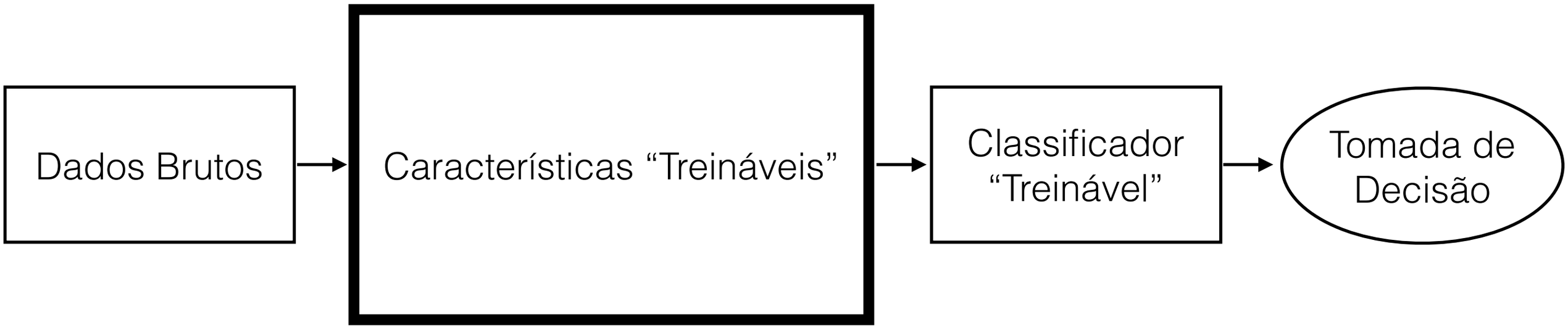


Definição Básica

- Paradigma de Aprendizado de Máquina

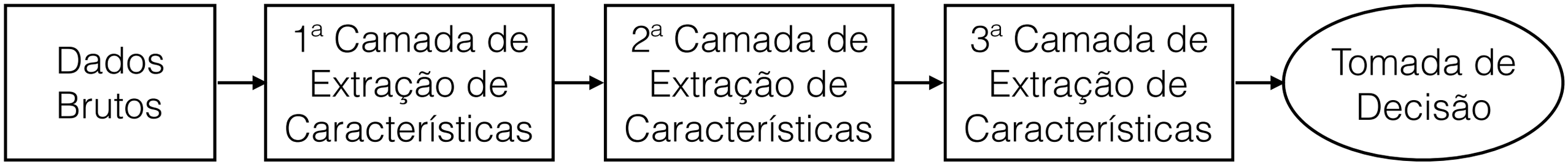


- Paradigma de Deep Learning



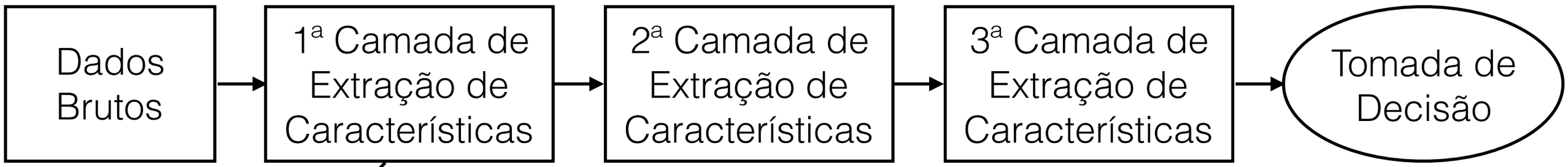
Definição Básica

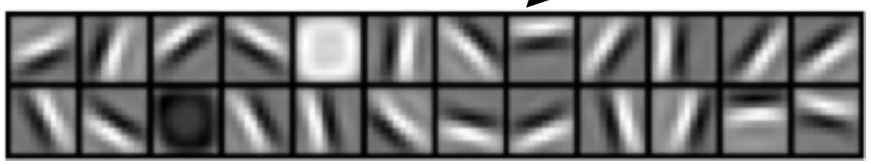
- Exemplo: Em “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”, Lee et al., 2009, temos:



Definição Básica

- Exemplo: Em “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”, Lee et al., 2009, temos:

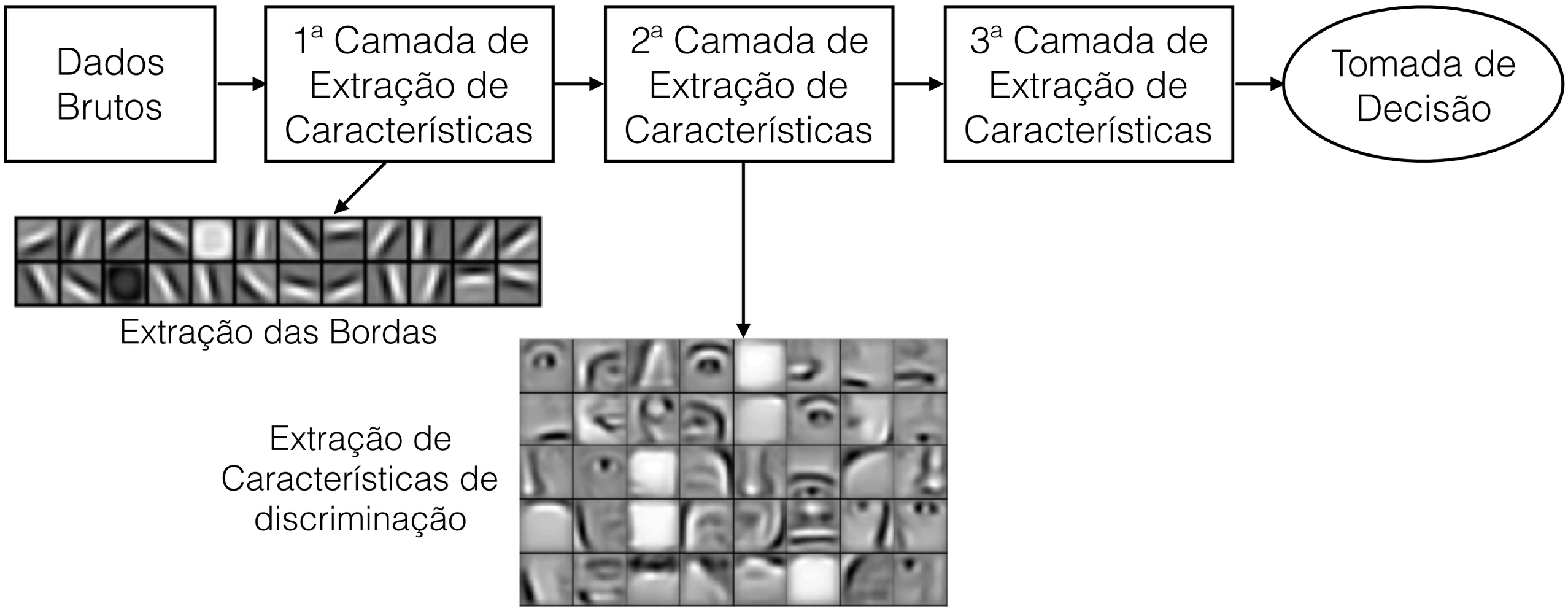




Extração das Bordas

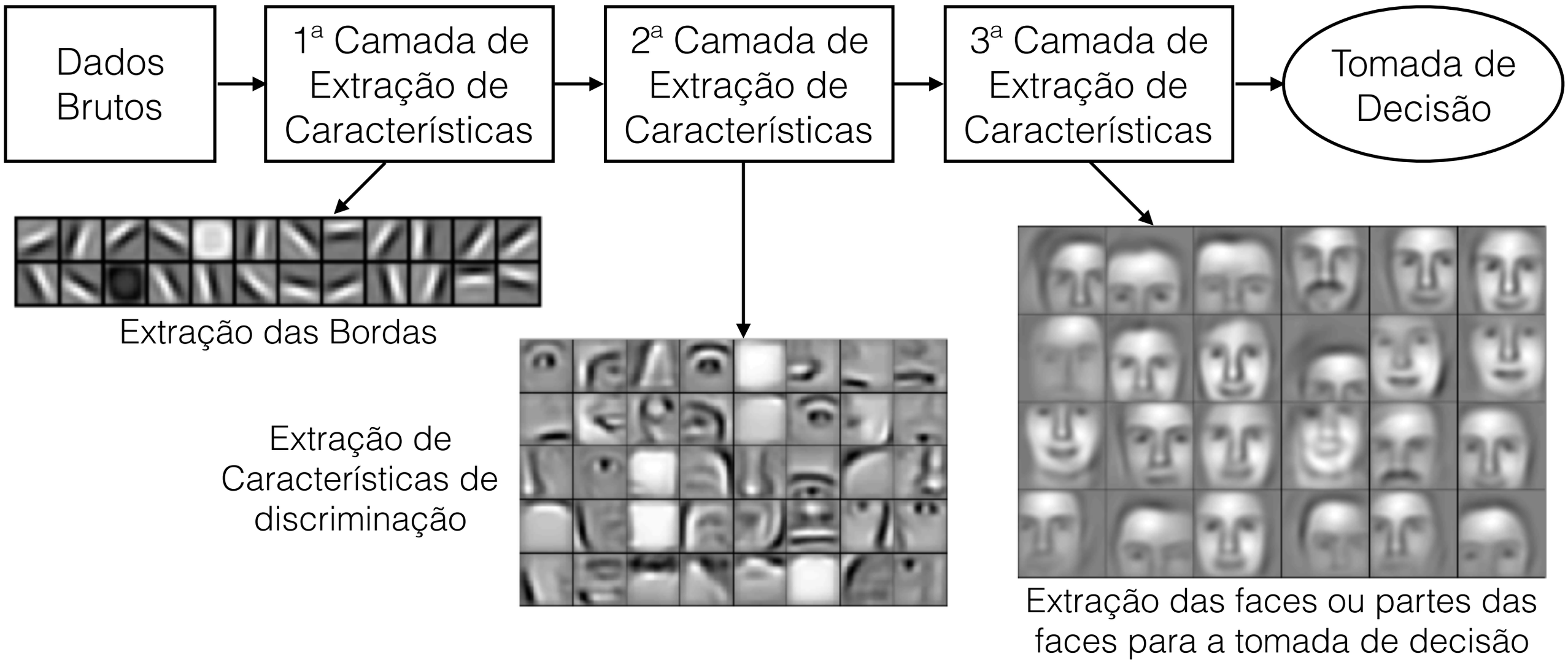
Definição Básica

- Exemplo: Em “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”, Lee et al., 2009, temos:



Definição Básica

- Exemplo: Em “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”, Lee et al., 2009, temos:



Definição Básica

- A **family** of methods that uses **deep architectures** to **learn high-level** feature representations

- Uma **família** de métodos que utiliza **arquiteturas profundas** para **aprender** representações para características de **alto nível**.

Definição Básica

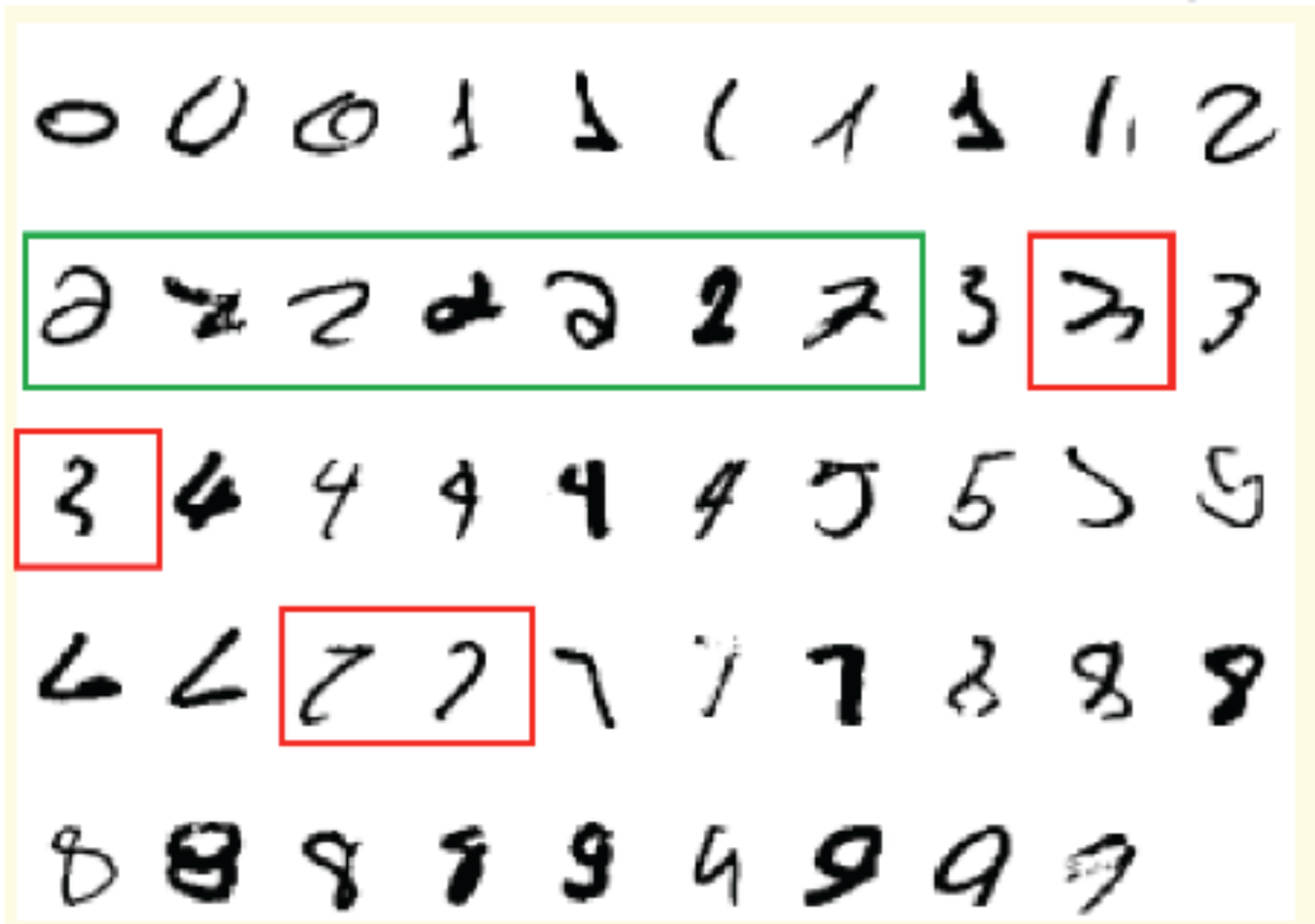
- A **family** of methods that uses **deep architectures** to **learn high-level** feature representations

- Uma **família** de métodos que utiliza **arquiteturas profundas** para **aprender** representações para características de **alto nível**.

Terraplanagem em Aprendizado de Máquina

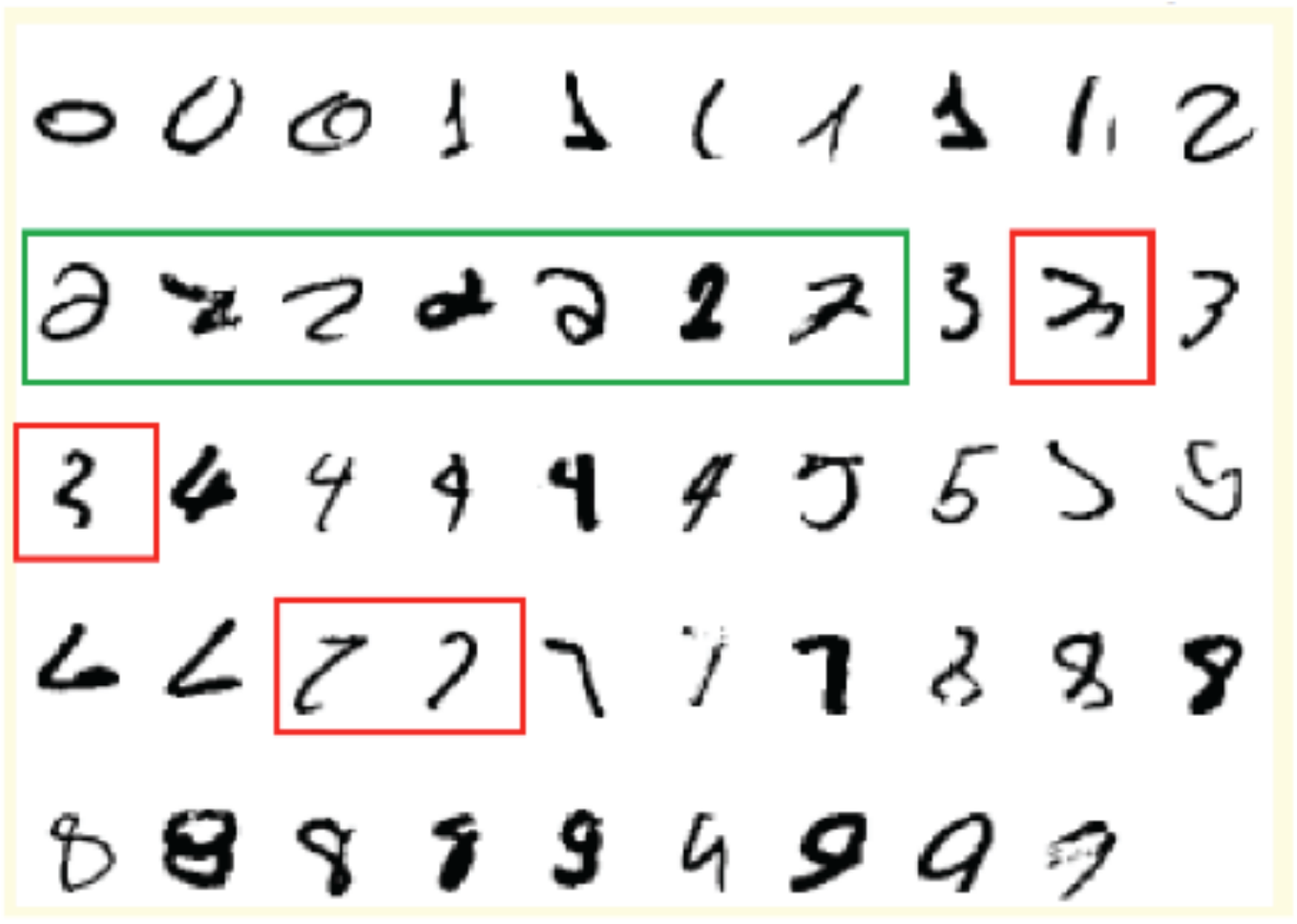
Aprendizado de Máquina

- Por que aprendizado de máquina é utilizado?
  - Por exemplo: USPS - Reconhecer o algarismo "**2**" nas amostras dadas



Aprendizado de Máquina

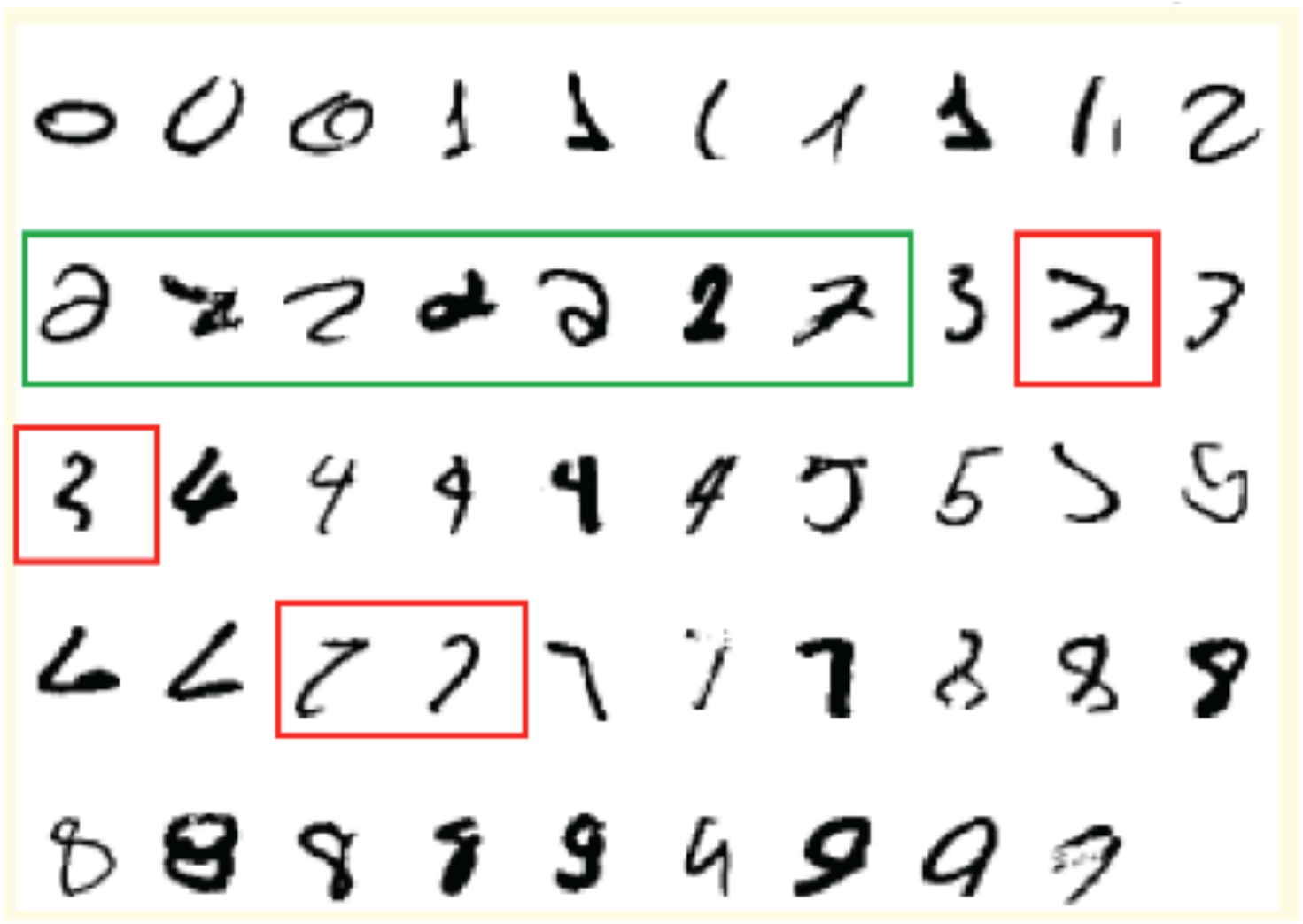
- Por que aprendizado de máquina é utilizado?
  - Por exemplo: USPS - Reconhecer o algarismo “**2**” nas amostras dadas



Neste caso, é muito complexo escrever uma função que reconheça diferentes tipos de **2**, então…

Aprendizado de Máquina

- Por que aprendizado de máquina é utilizado?
  - Por exemplo: USPS - Reconhecer o algarismo “**2**” nas amostras dadas

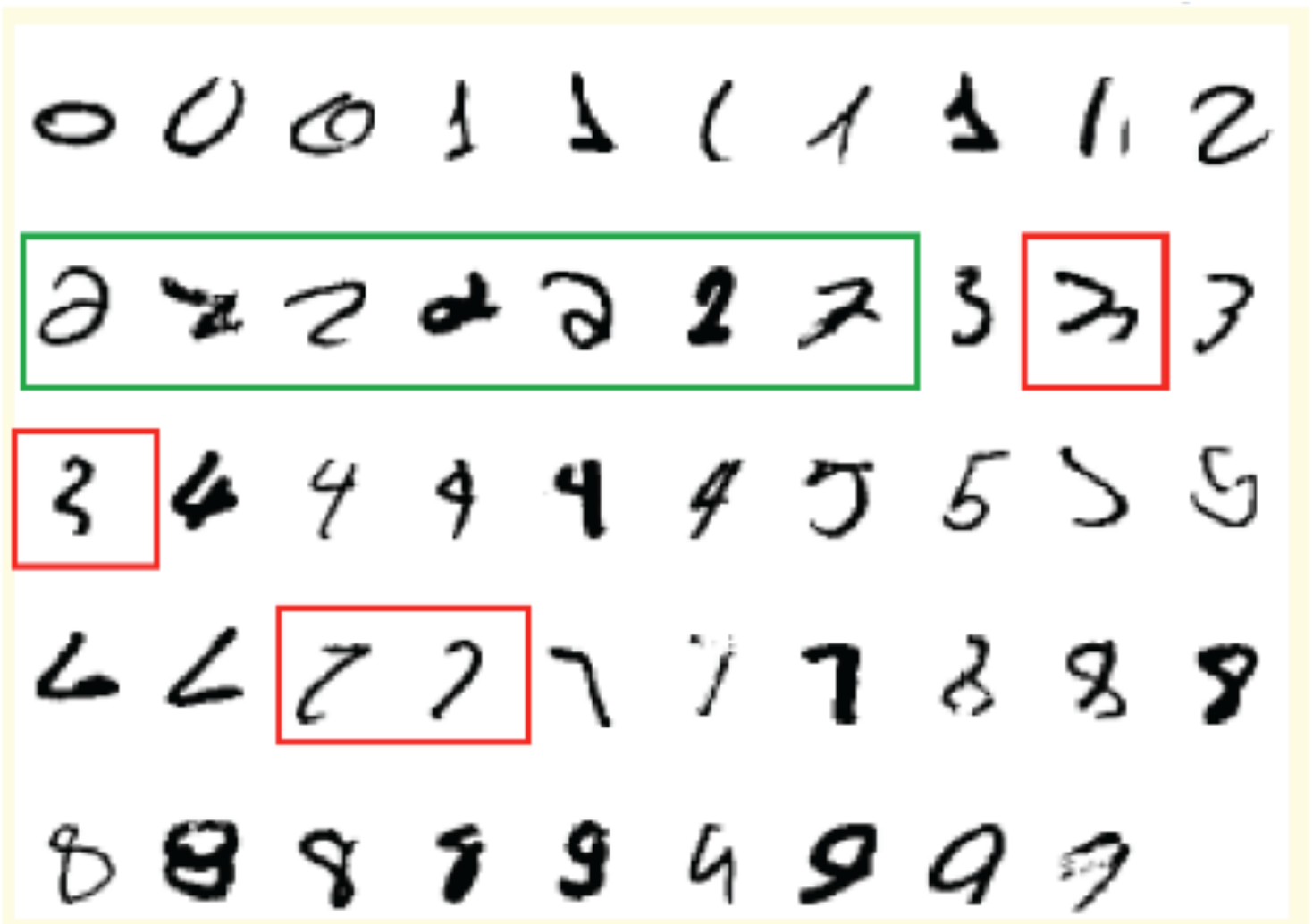


Neste caso, é muito complexo escrever uma função que reconheça diferentes tipos de **2**, então…

Aprendizado de Máquina

Aprendizado de Máquina

- Por que aprendizado de máquina é utilizado?
  - Por exemplo: USPS - Reconhecer o algarismo “**2**” nas amostras dadas

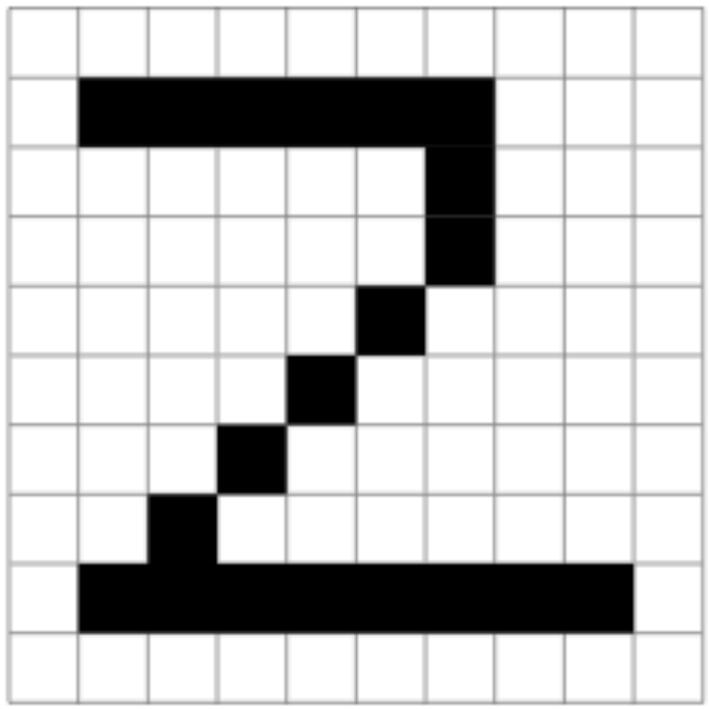


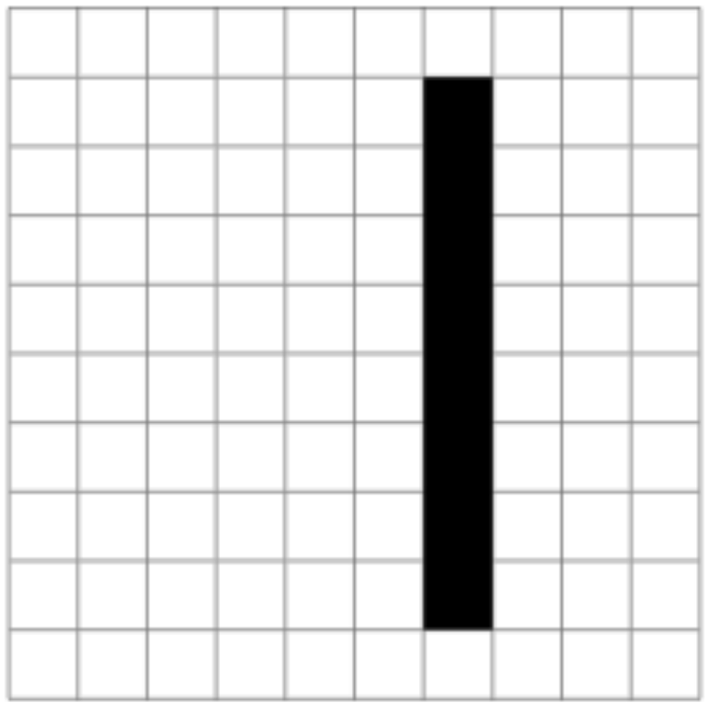
Aprendizado de Máquina

Neste caso, deve **aprender** qual é a função geratriz de **2** e classificar corretamente o número apresentado

Aprendizado de Máquina

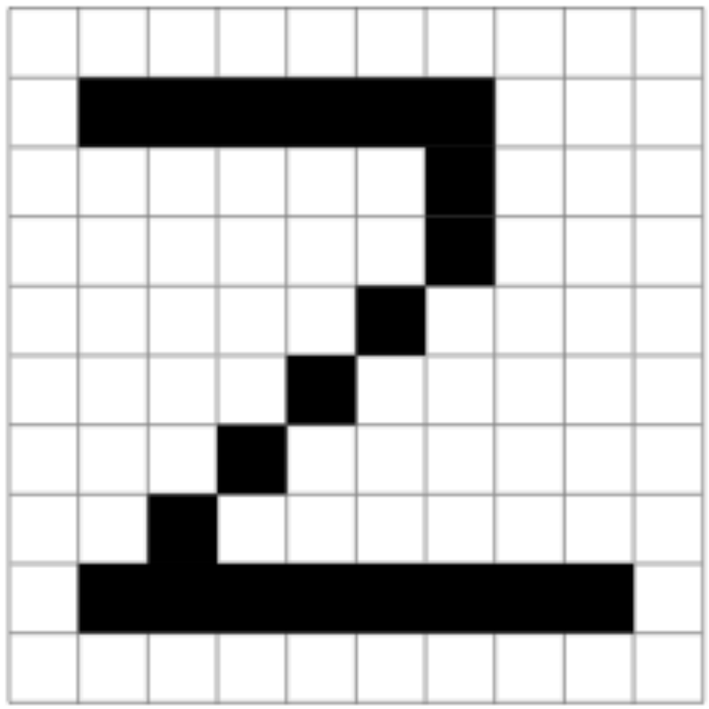
- Uma solução possível é…
  - As amostras de treinamento são representadas como matrizes de pixels

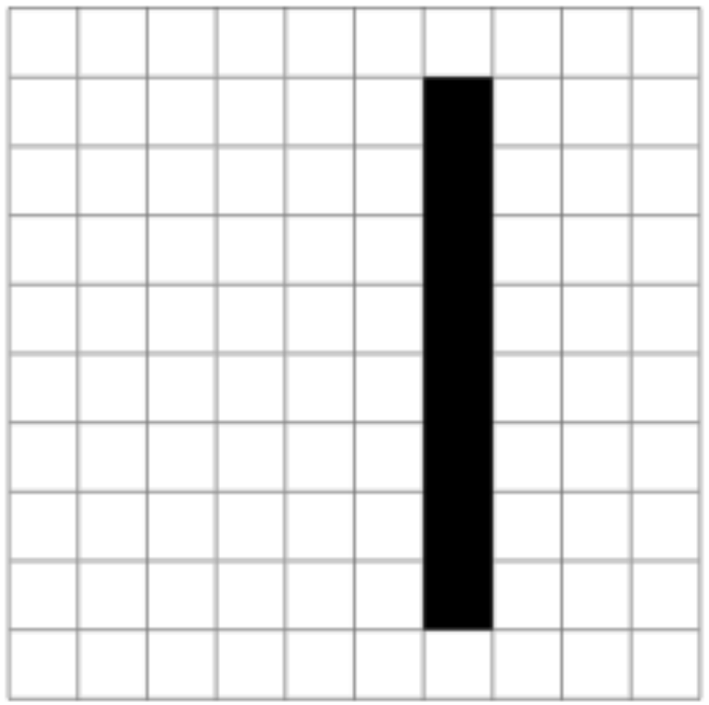




Aprendizado de Máquina

- Uma solução possível é…
  - As amostras de treinamento são representadas como matrizes de pixels





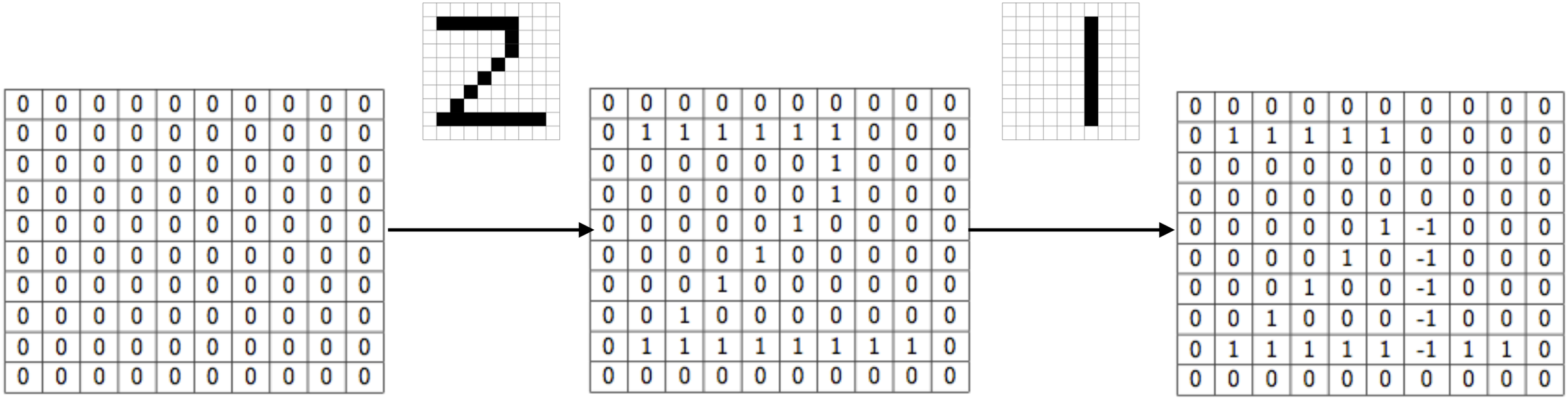
- Podemos fazer um classificador como uma matriz de pesos de mesma dimensão

Aprendizado de Máquina

- O processo de treinamento seria:
  - Para eventos do algarismo **2**, é adicionado 1 aos elementos correspondentes da matriz de pesos
  - Para eventos **não-2**, é subtraído 1 dos elementos correspondentes da matriz de pesos

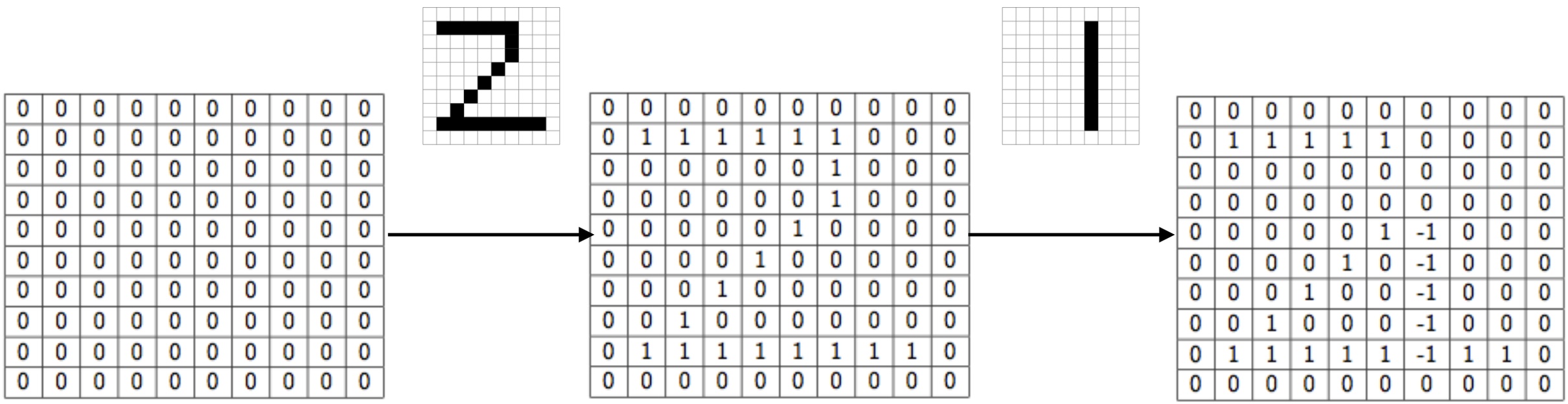
Aprendizado de Máquina

- O processo de treinamento seria:
  - Para eventos do algarismo **2**, é adicionado 1 aos elementos correspondentes da matriz de pesos
  - Para eventos **não-2**, é subtraído 1 dos elementos correspondentes da matriz de pesos



Aprendizado de Máquina

- O processo de treinamento seria:
  - Para eventos do algarismo **2**, é adicionado 1 aos elementos correspondentes da matriz de pesos
  - Para eventos **não-2**, é subtraído 1 dos elementos correspondentes da matriz de pesos



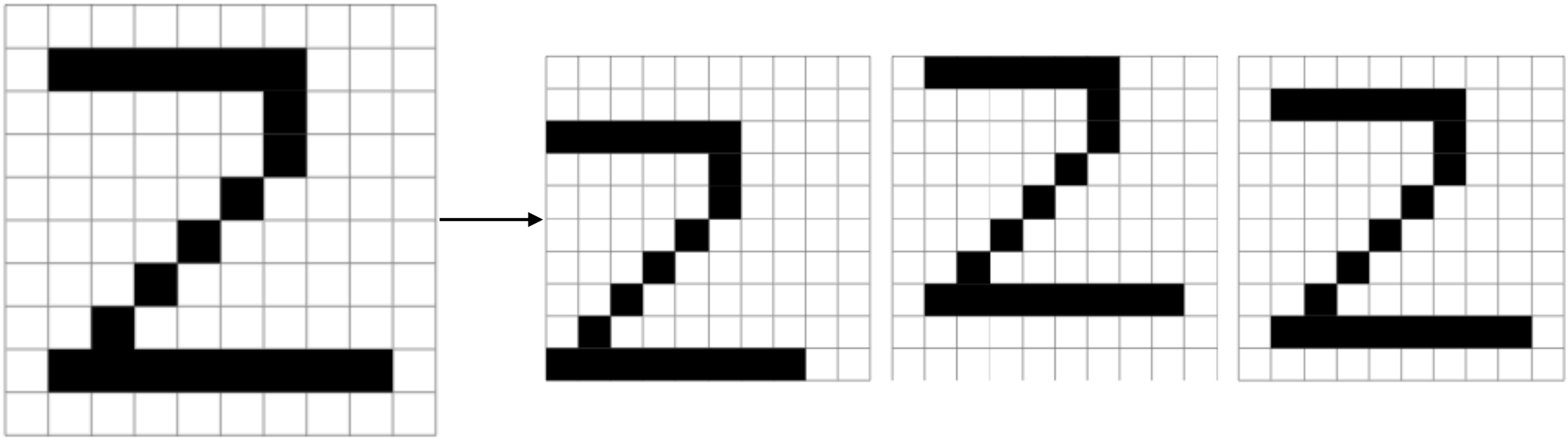
- Para a classificação: se somatório do produto entre matrix treinada e o algarismo > 0 é **2**, caso contrário **não-2**

Aprendizado de Máquina

- Memorização dos Dados vs Generalização
  - Há uma limitação nos dados de treinamento!!!
    - Classificador Memoriza os dados de treinamento = Performance ruim para **novos dados**
  - Generalização é a habilidade de obtermos resultados acurados em dados **não observados no treinamento**.

Aprendizado de Máquina

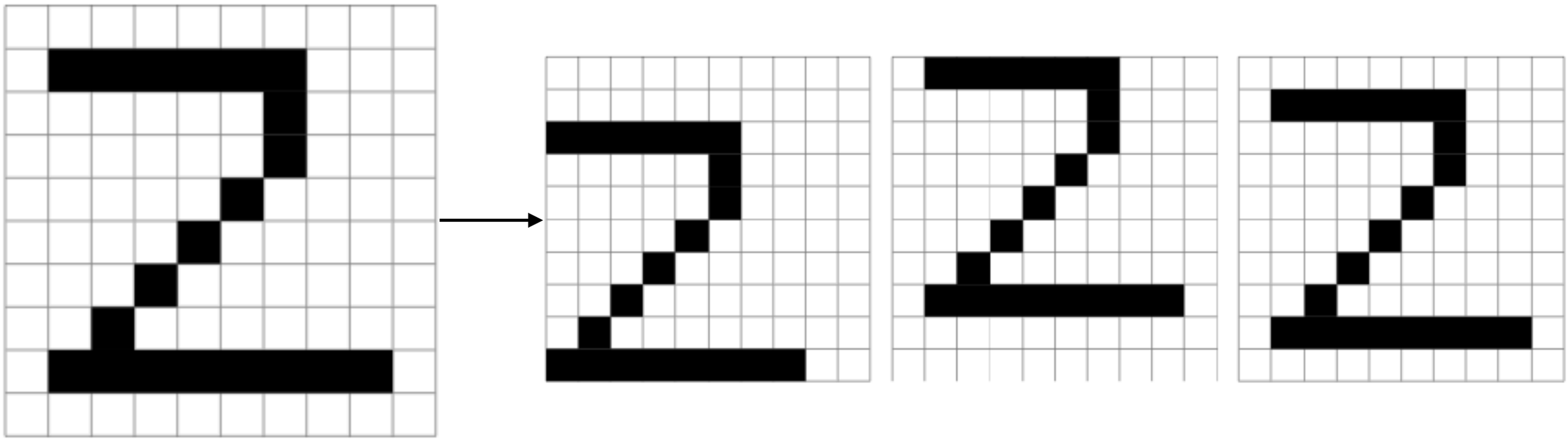
- Voltando ao exemplo..
  - Considerando pequenas alterações nas amostras



- O classificador treinado seria capaz de generalizar o resultado?

Aprendizado de Máquina

- Voltando ao exemplo..
  - Considerando pequenas alterações nas amostras



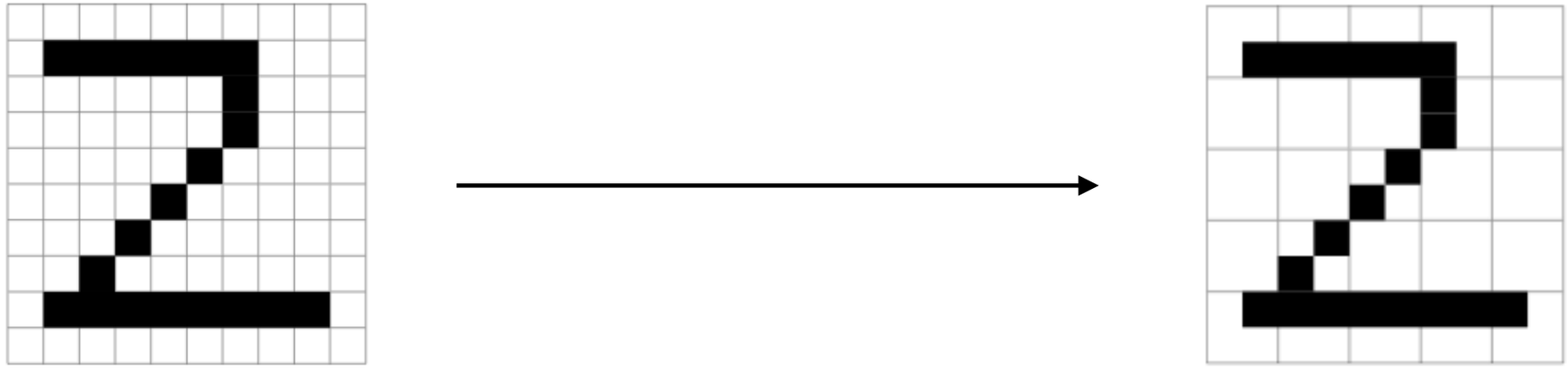
- O classificador treinado seria capaz de generalizar o resultado? **Não!!!**

Aprendizado de Máquina

- Uma possível maneira de aumentar o poder de generalização seria…
  - Considerar um conjunto de pixels e não somente o valor de um único pixel **(menos pesos na matriz de pesos)**

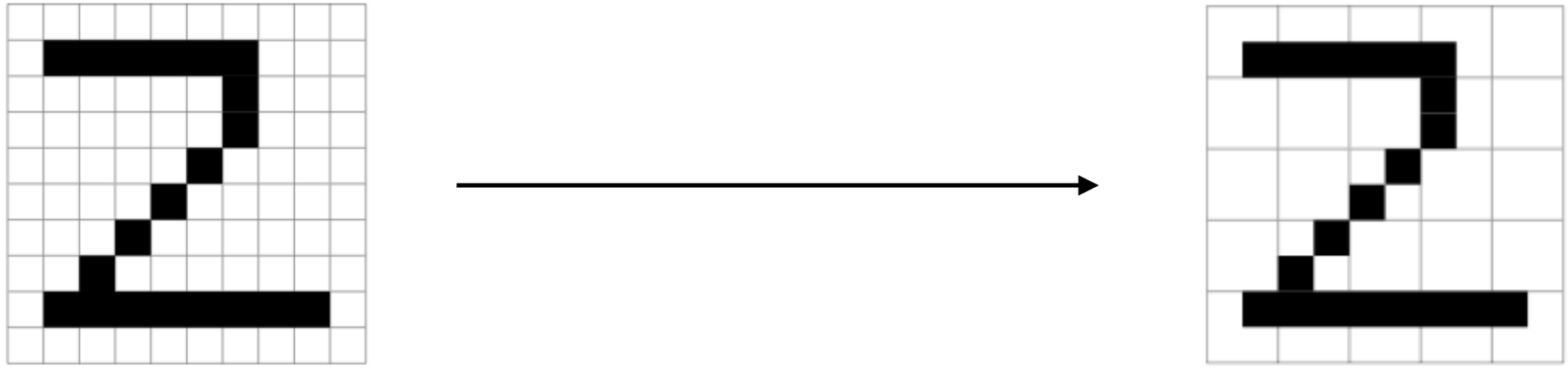
Aprendizado de Máquina

- Uma possível maneira de aumentar o poder de generalização seria…
  - Considerar um conjunto de pixels e não somente o valor de um único pixel **(menos pesos na matriz de pesos)**



Aprendizado de Máquina

- Uma possível maneira de aumentar o poder de generalização seria…
  - Considerar um conjunto de pixels e não somente o valor de um único pixel **(menos pesos na matriz de pesos)**

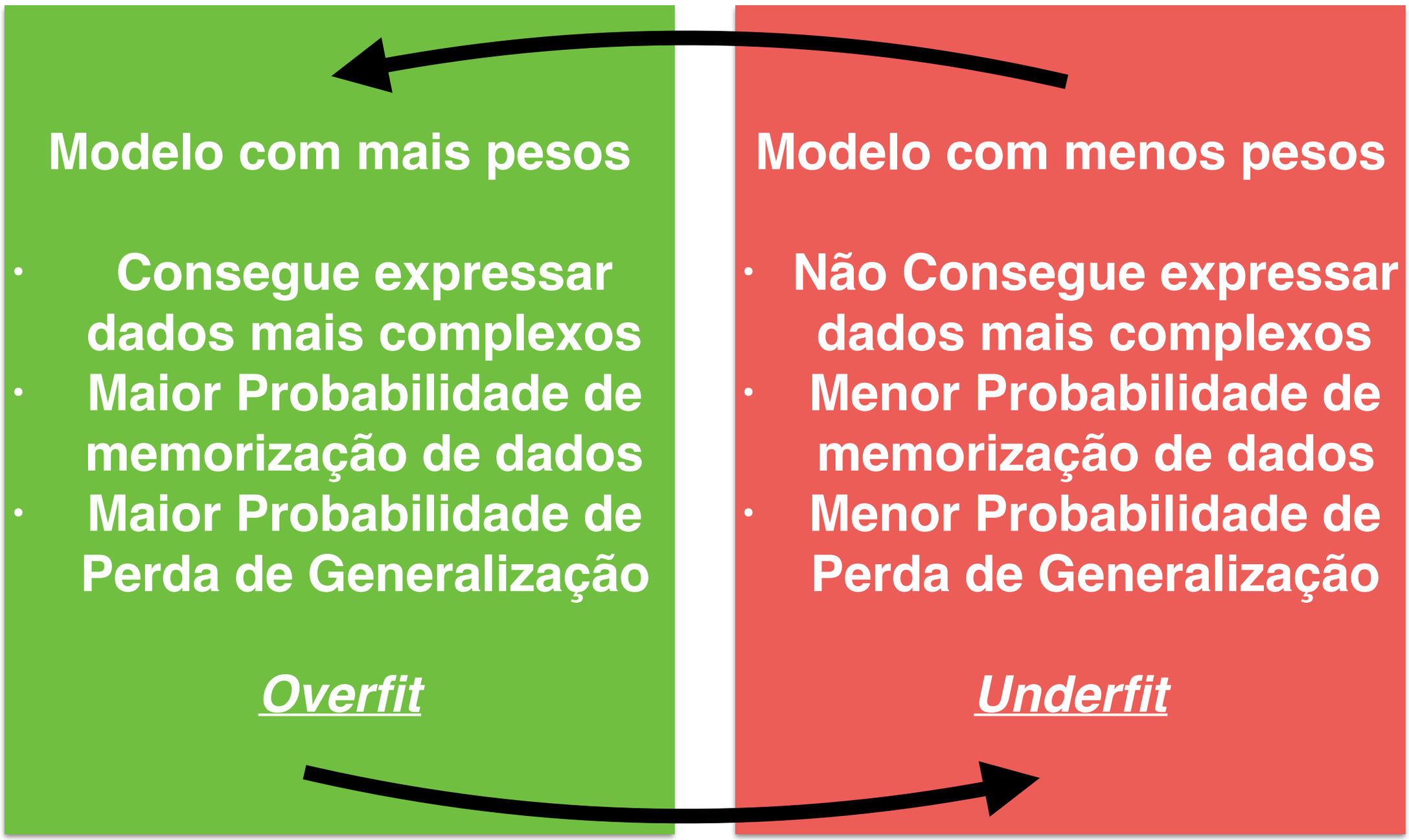


0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	-1	0	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0	0
0	0	1	0	0	0	-1	0	0	0
0	1	1	1	1	1	-1	1	1	0
0	0	0	0	0	0	0	0	0	0

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

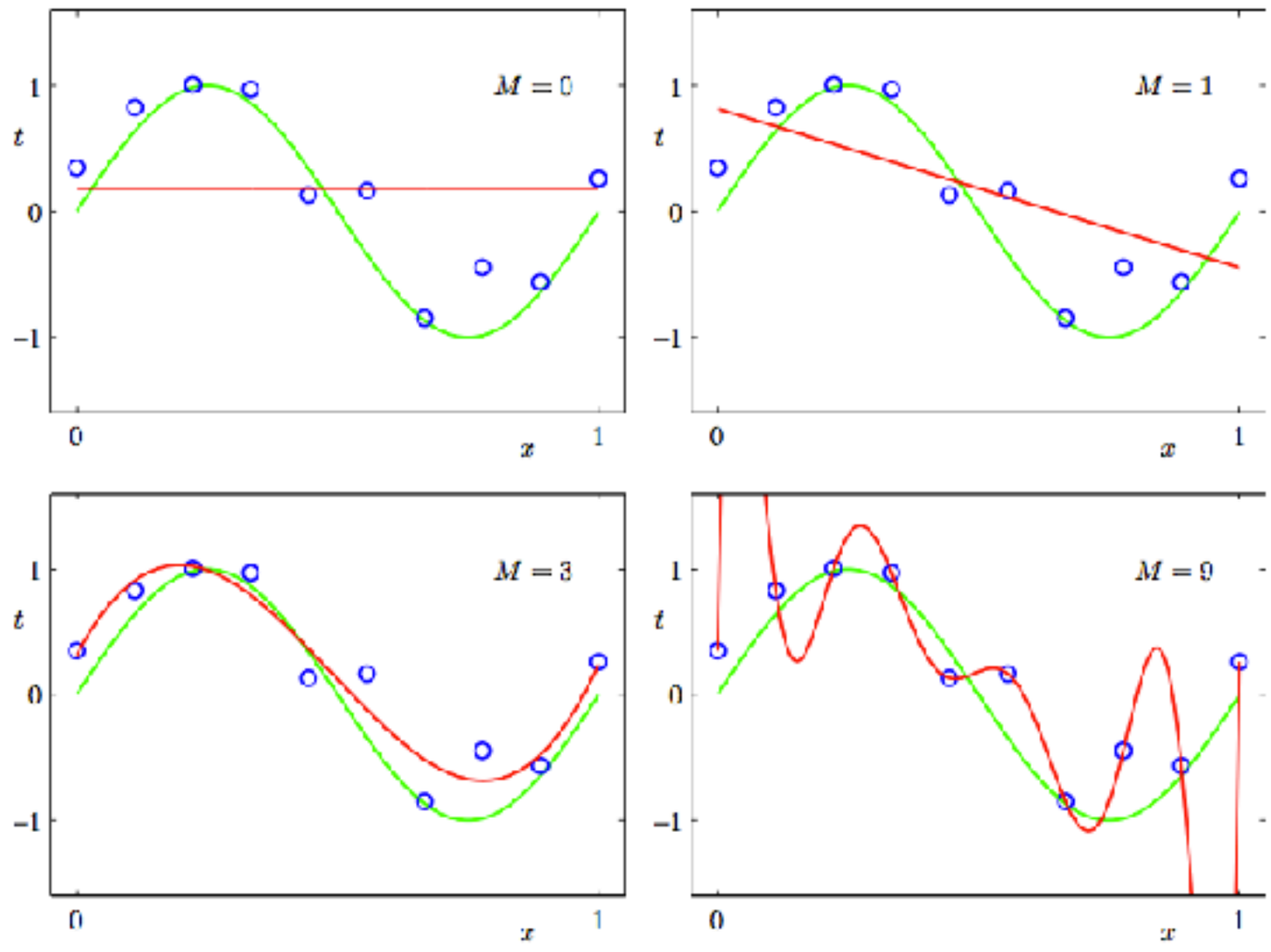
Aprendizado de Máquina

- Chegamos a um trade-off



Aprendizado de Máquina

- Ajuste de modelo (Fit)
  - Ajustar um modelo polinomial aos dados em azul com MSE
  - Modelo polinomial: $f(x) = w_0+w_1x+\ldots+w_mx^m$



Aprendizado de Máquina

- Definições
  - Objetivo do Treinamento: Aprender uma função f tal que:

$$f : x \rightarrow y$$

- Tal que x são entradas e y são saídas
- Antes do Treinamento:
  1. Escolha de um modelo: Regressão Logística, SVM, Redes Neurais…
  2. Escolha de uma função custo - Dissimilaridade entre alvo e saída do modelo
  3. Escolha de um otimizador (dependente do modelo)

Terraplanagem em Redes Neurais Artificiais

Redes Neurais

- Redes Neurais de uma camada (Regressão Logística)
  - Formato da função: $f(x) = \sigma(w^T \cdot x + b)$
  - Parametros da função: vetor w (pertencente a R^d) e b é um termo escalar chamado *bias*
- Sigma é uma não linearidade (sigmóide, tanh ou outra)
- Por simplicidade, a função pode ser escrita como:

$$f(x) = \sigma(w_0^T \cdot x)$$

Redes Neurais

- Redes Neurais de uma camada (Regressão Logística)
  - Função Custo: $C(w) = \frac{1}{2}\sum(\sigma(w^T x^m) - y^m)^2$
  - Gradiente da Função Custo:

$$\nabla C(w) = \sum_m \left[\sigma(w^T x^m) - y^m\right] \sigma'(w^T x^m) x^m$$

- Forma geral:

$$\nabla C(w) = \sum_m Error^m \sigma'(in^m) x^m$$

Redes Neurais

- Redes Neurais de uma camada (Regressão Logística)
  - Supondo que a não-linearidade seja $\sigma(z) = \dfrac{1}{1+e^{-z}}$

$$\sigma'(z) = \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) = -\left(\frac{1}{1+e^{-z}}\right)^2 \frac{\partial}{\partial z}(1+e^{-z})$$

$$\sigma'(z) = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) = \sigma(z)(1-\sigma(z))$$

Redes Neurais

- Forma Geral do Gradiente da função custo

$$\nabla C(w) = \sum_m Error^m \sigma'(in^m) x^m$$

- Algoritmo do *Gradient Descent* (Batelada)

1. Inicialização dos pesos w
2. Cálculo do gradiente
3. Atualização dos pesos (regra delta)
4. Repetir os passos 2-3 até um critério de parada ser atingido

Redes Neurais

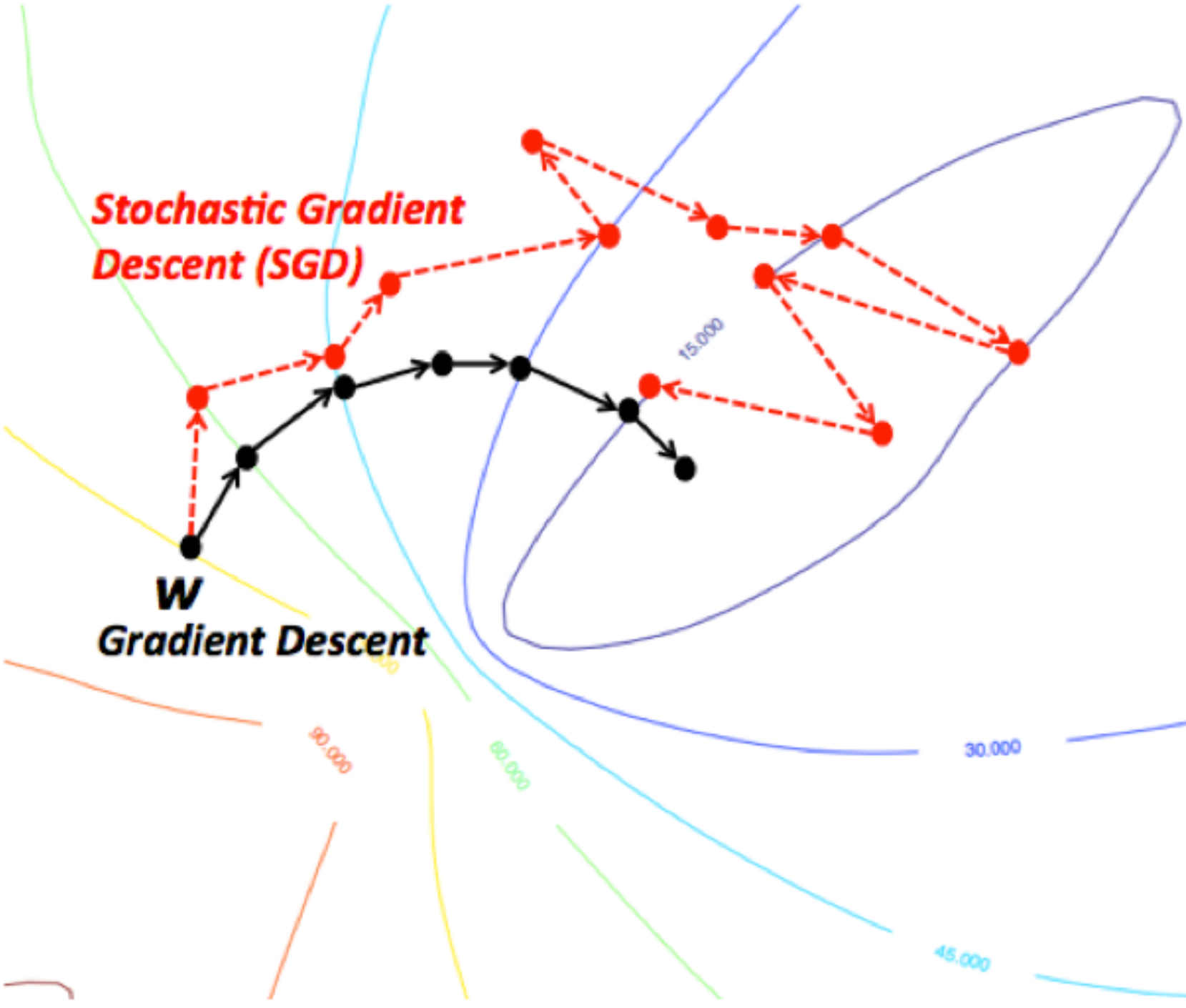
- Forma Geral do Gradiente da função custo

$$\nabla C(w) = Error^m \sigma'(in^m) x^m$$

- Algoritmo do *Gradient Descent*

1. Inicialização dos pesos w
2. Cálculo do gradiente para cada evento
3. Atualização dos pesos (regra delta)
4. Repetir os passos 2-3 até um critério de parada ser atingido

Redes Neurais



Redes Neurais

- Outras considerações

  - Minimizar a função custo para o conjunto de treinamento pode não garantir generalização

  - Possibilidades de solução:

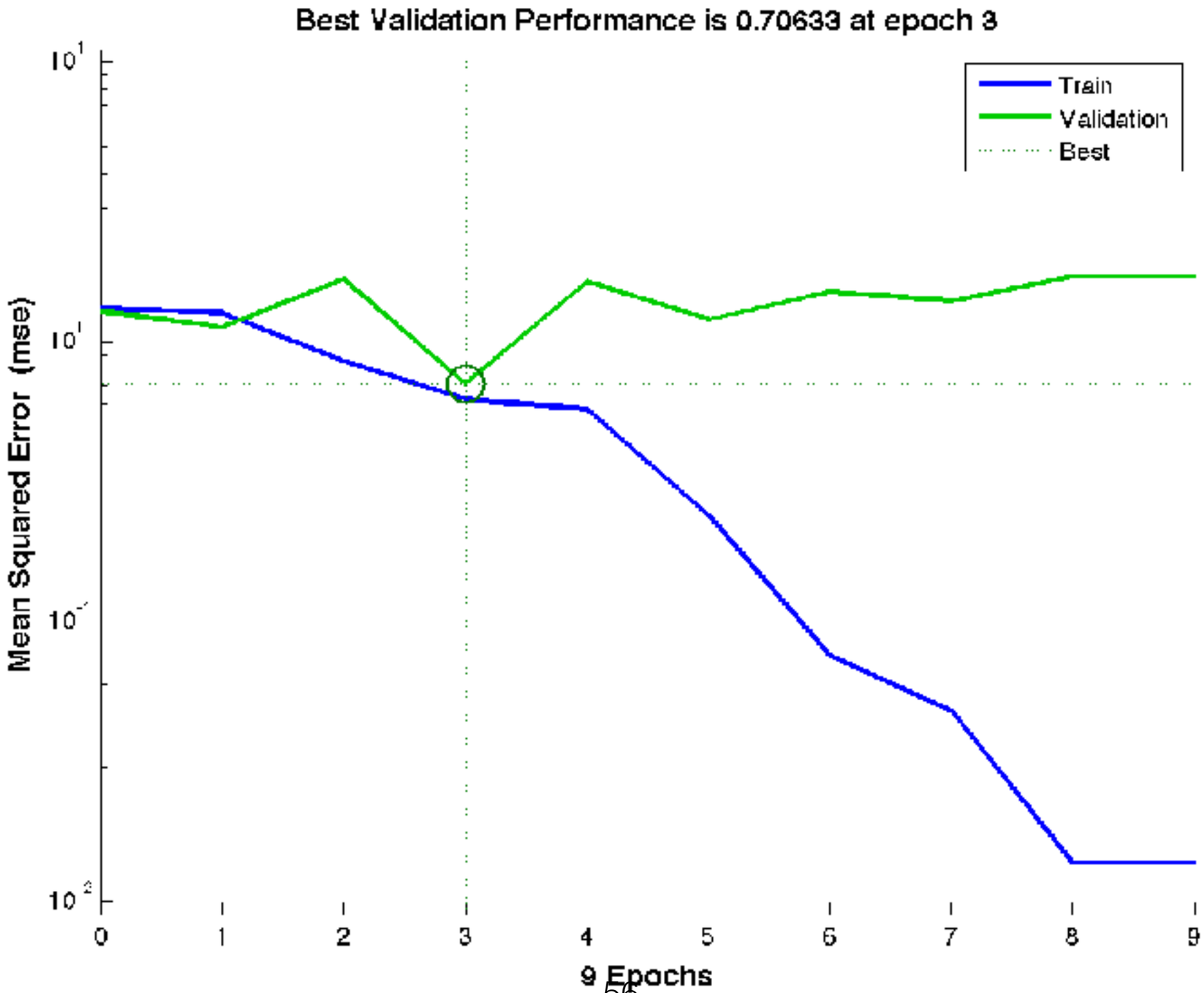
1. Regularização (L1)

$$C(w) = \frac{1}{2} \sum_{m} (\sigma(w^T x^m) - y^m)^2 + ||w||$$

2. Parada antecipada (*Early-Stopping*)

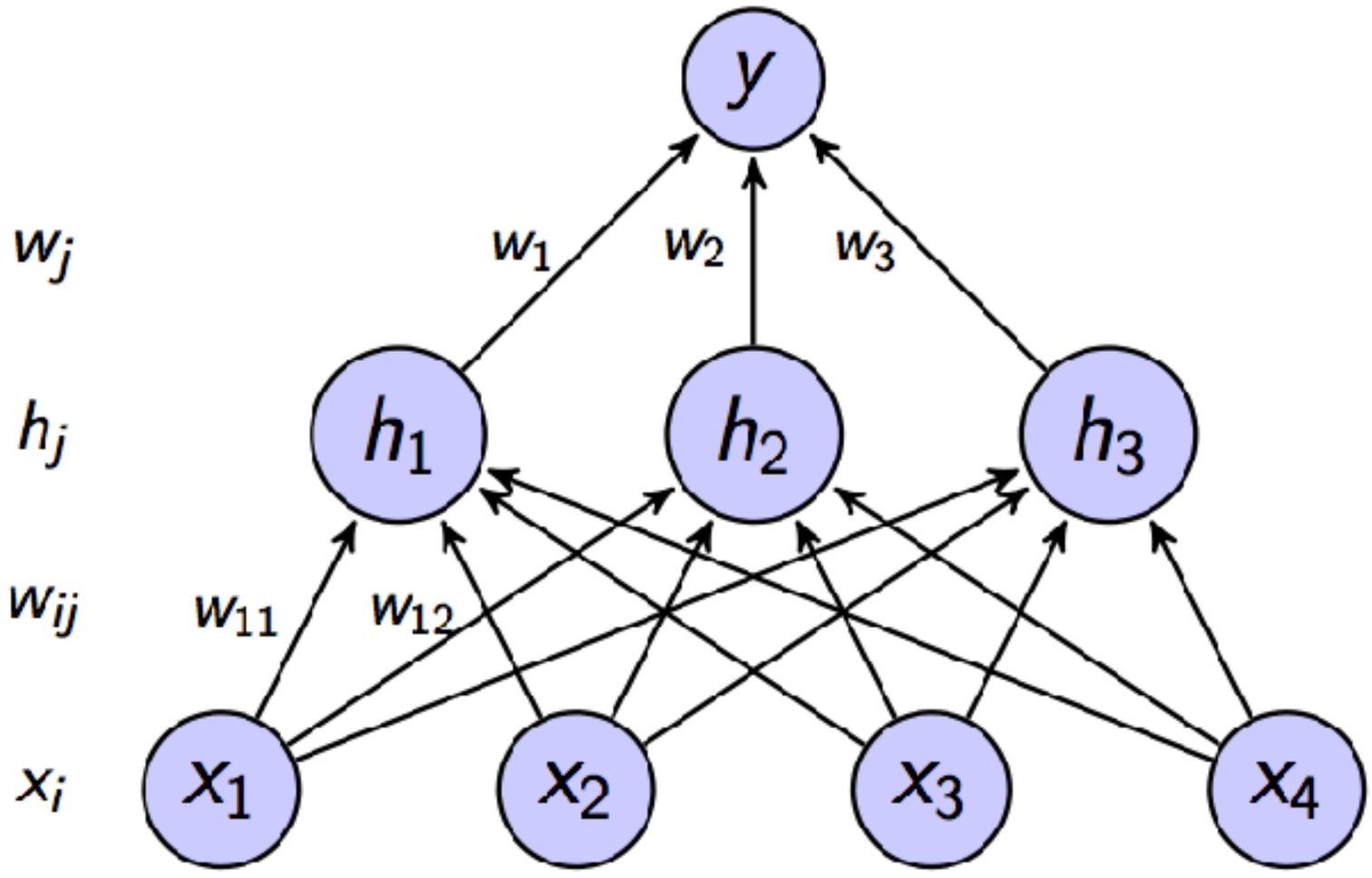
Redes Neurais

- Parada Antecipada (Early Stopping)



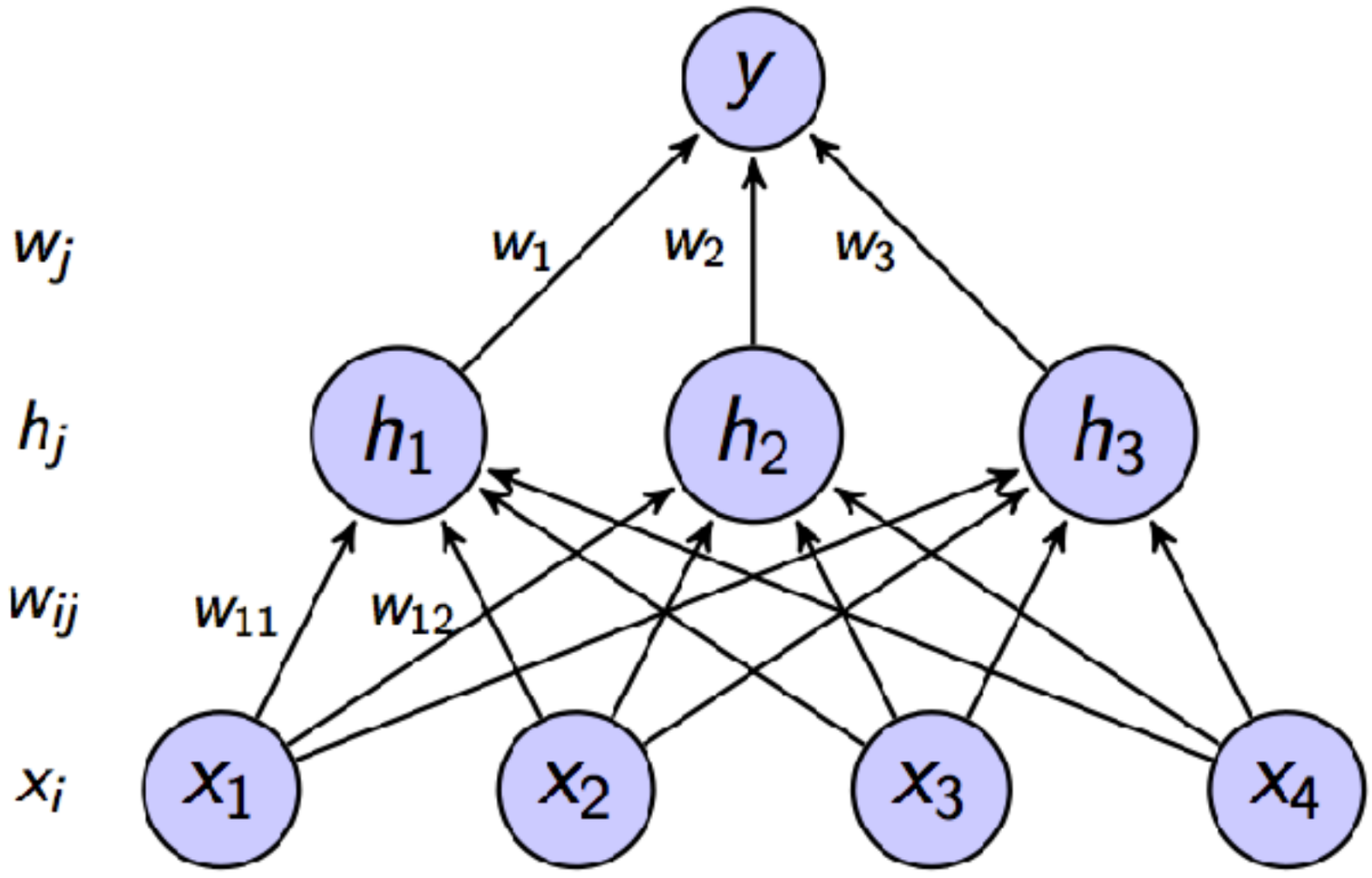
Redes Neurais

- Redes Neurais de duas camadas ou mais (MLP)



Redes Neurais

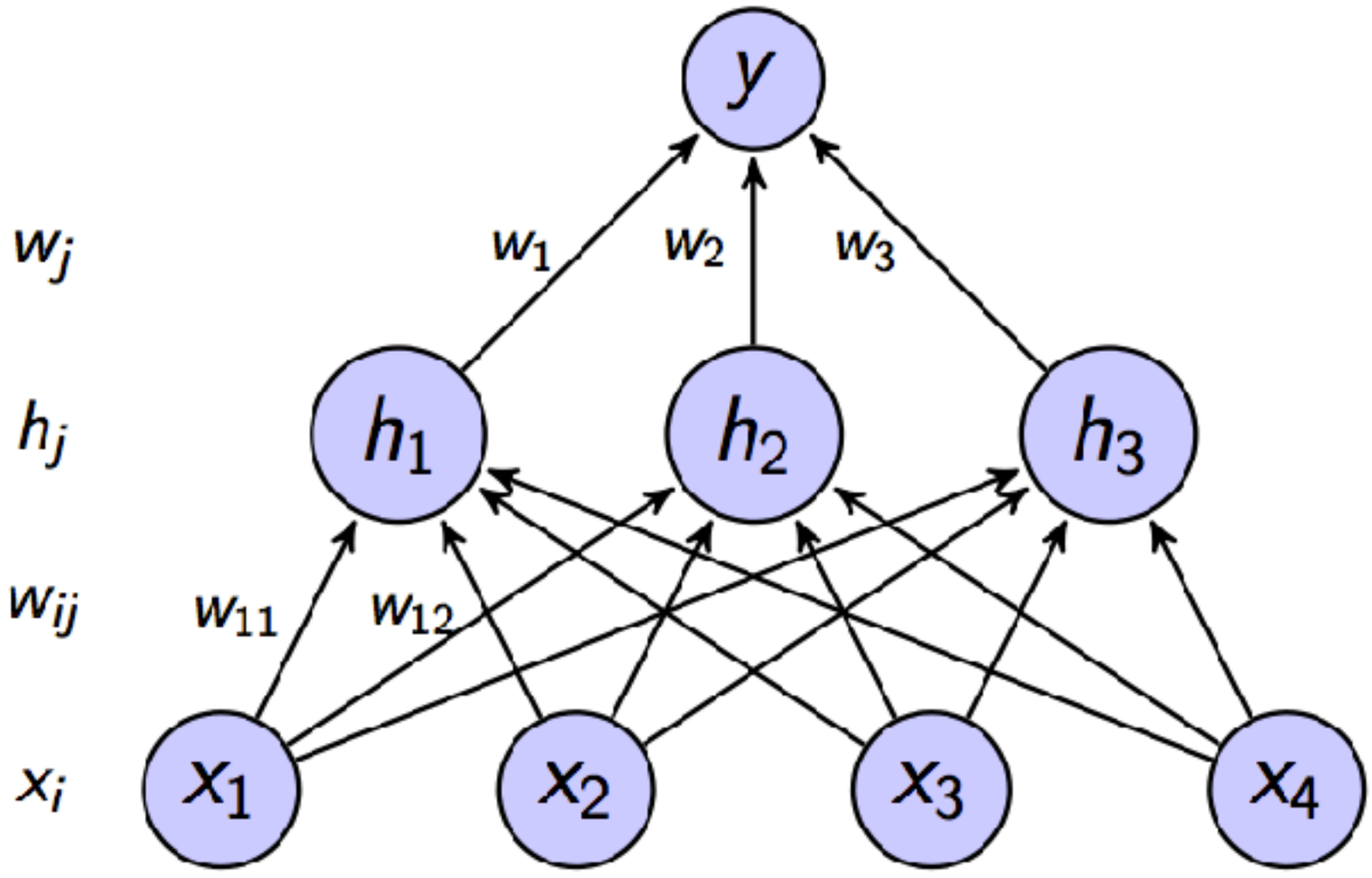
- Redes Neurais de duas camadas ou mais (MLP)



$$y = f(x) = \sigma\left(\sum_j w_j \cdot h_j\right) = \sigma\left(\sum_j w_j \cdot \sigma\left(\sum_i w_{ij} x_i\right)\right)$$

Redes Neurais

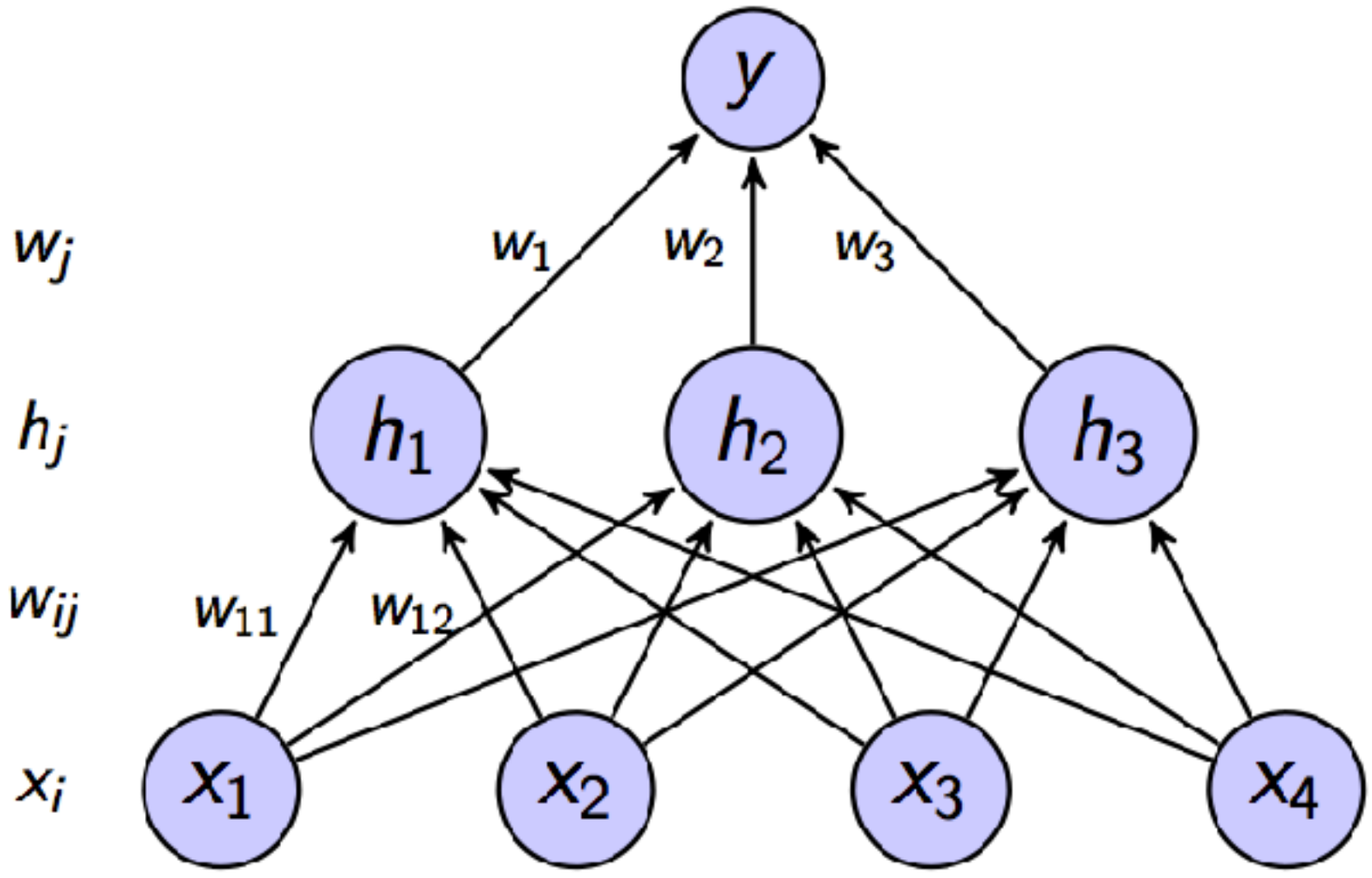
- Redes Neurais de duas camadas ou mais (MLP)



As variáveis h_i podem ser vistas como novas características (combinações) das variáveis x_i

Redes Neurais

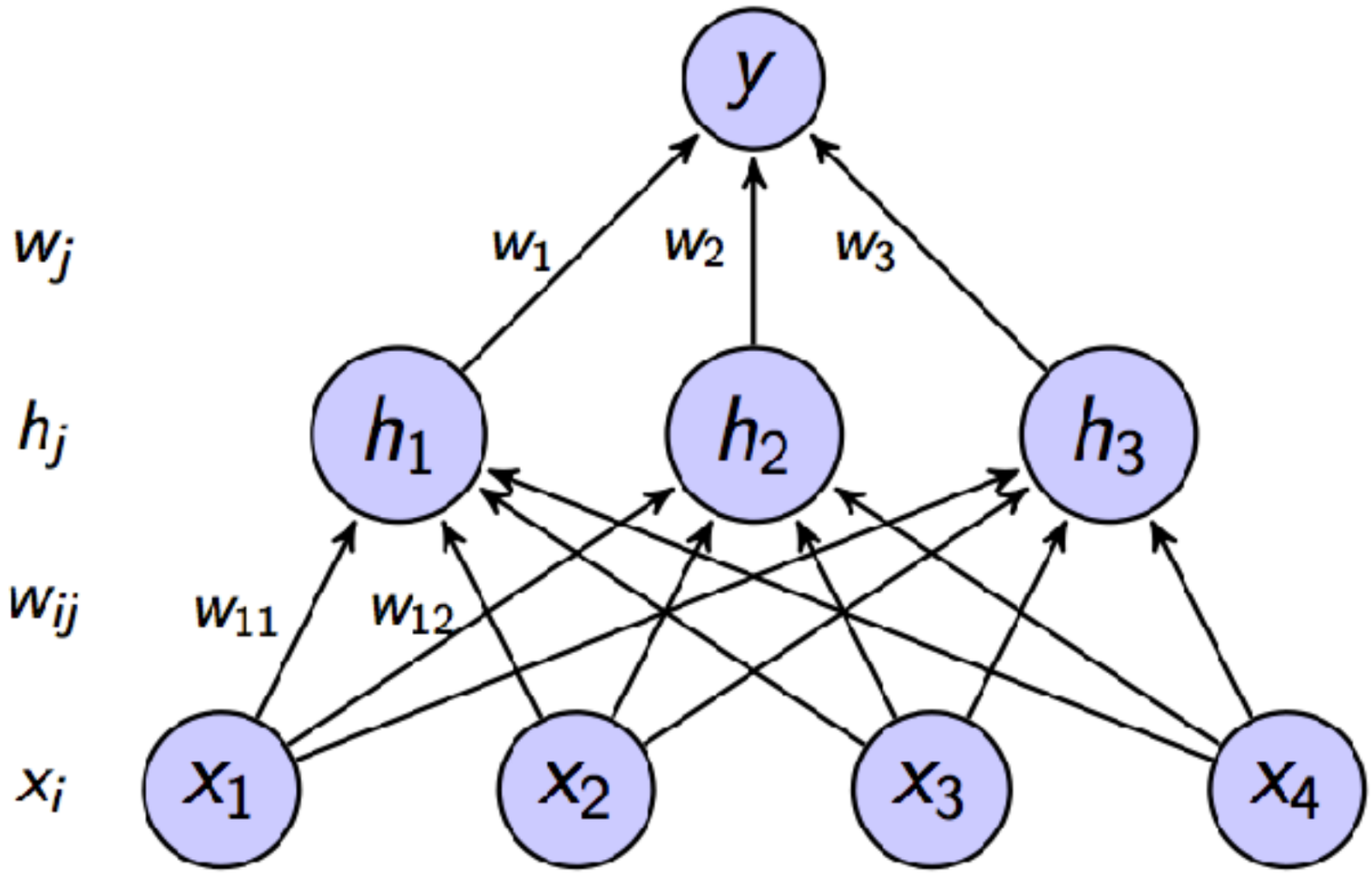
- Redes Neurais de duas camadas ou mais (MLP)



Agora o treinamento é feito em duas “vias”

Redes Neurais

- Redes Neurais de duas camadas ou mais (MLP)

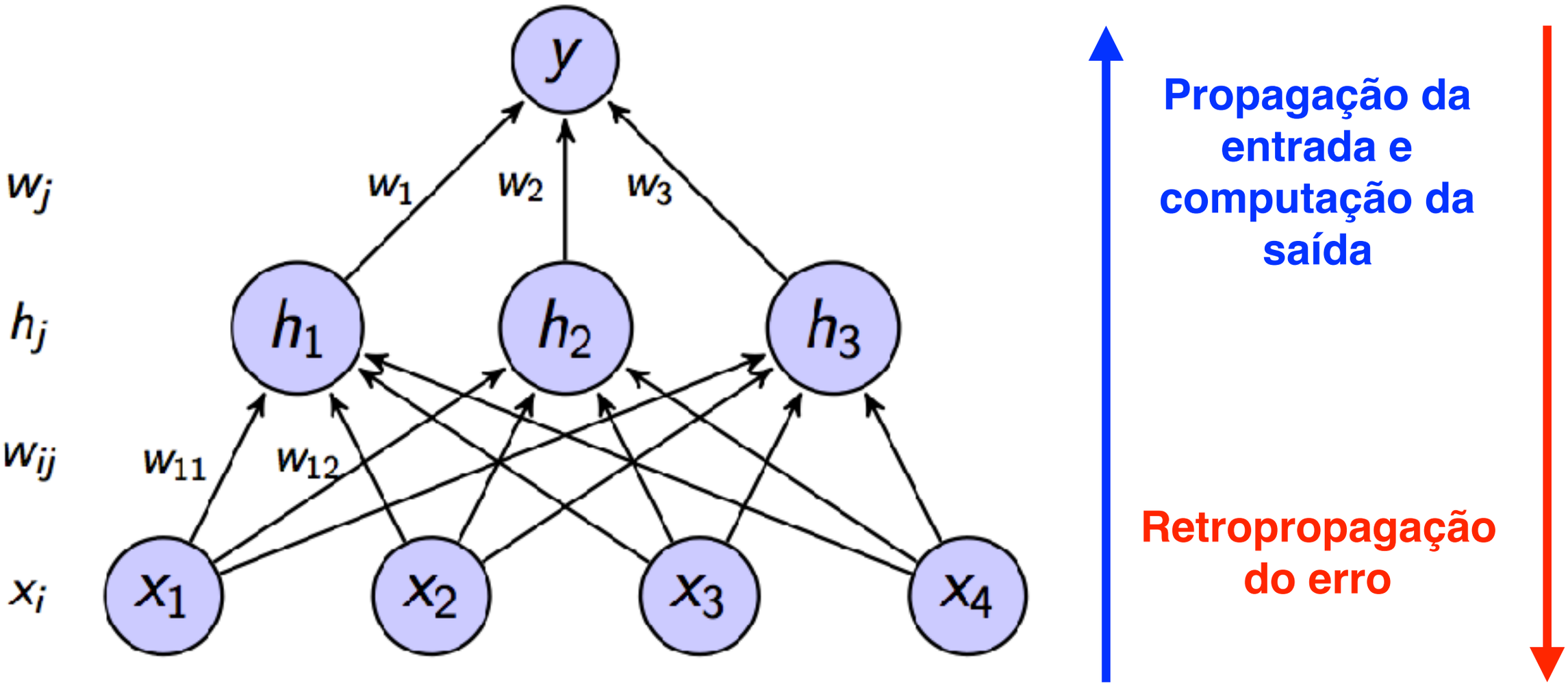


Propagação da entrada e computação da saída

Agora o treinamento é feito em duas “vias”

Redes Neurais

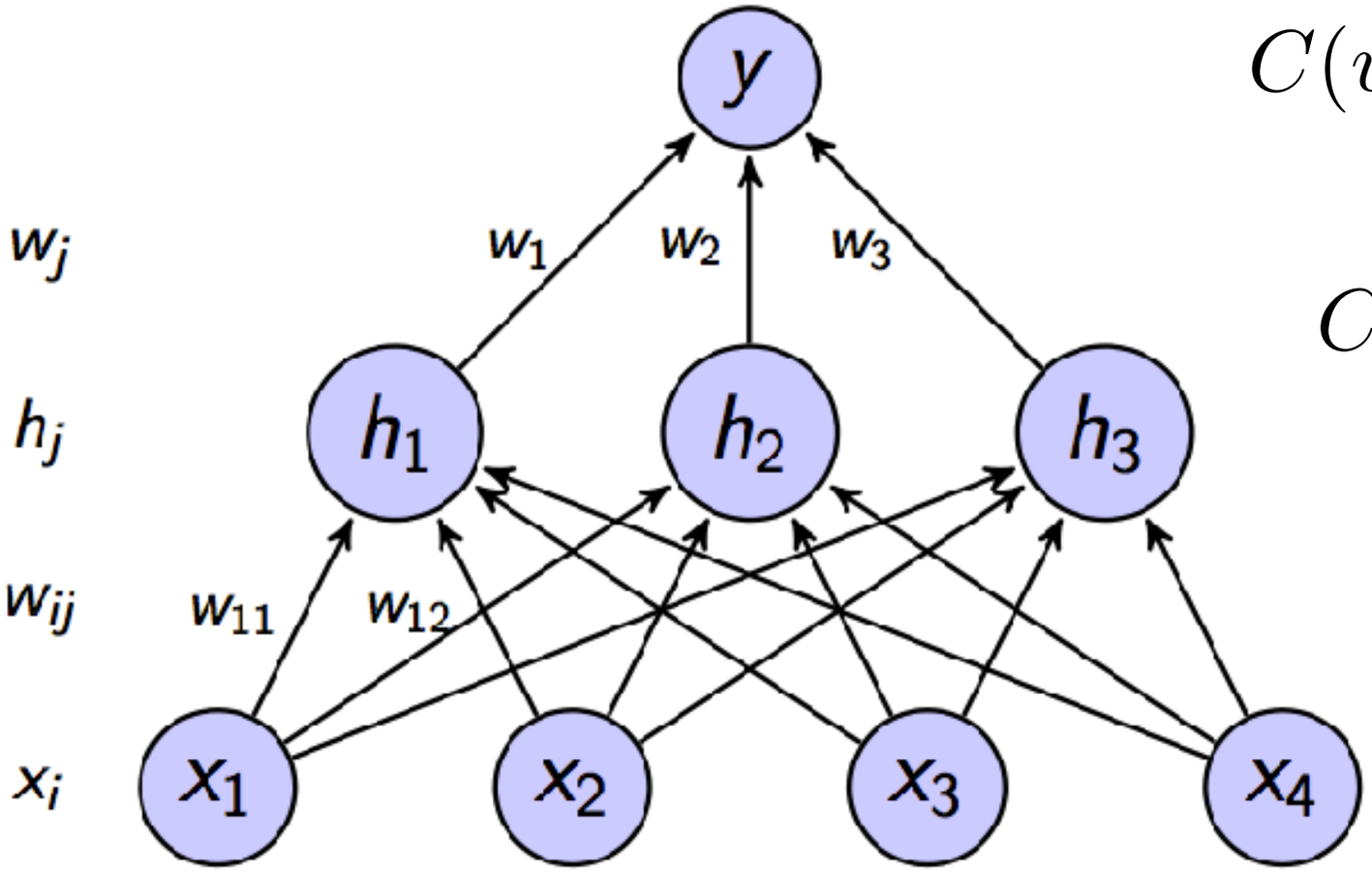
- Redes Neurais de duas camadas ou mais (MLP)



Agora o treinamento é feito em duas “vias”

Redes Neurais

- Redes Neurais de duas camadas ou mais (MLP)

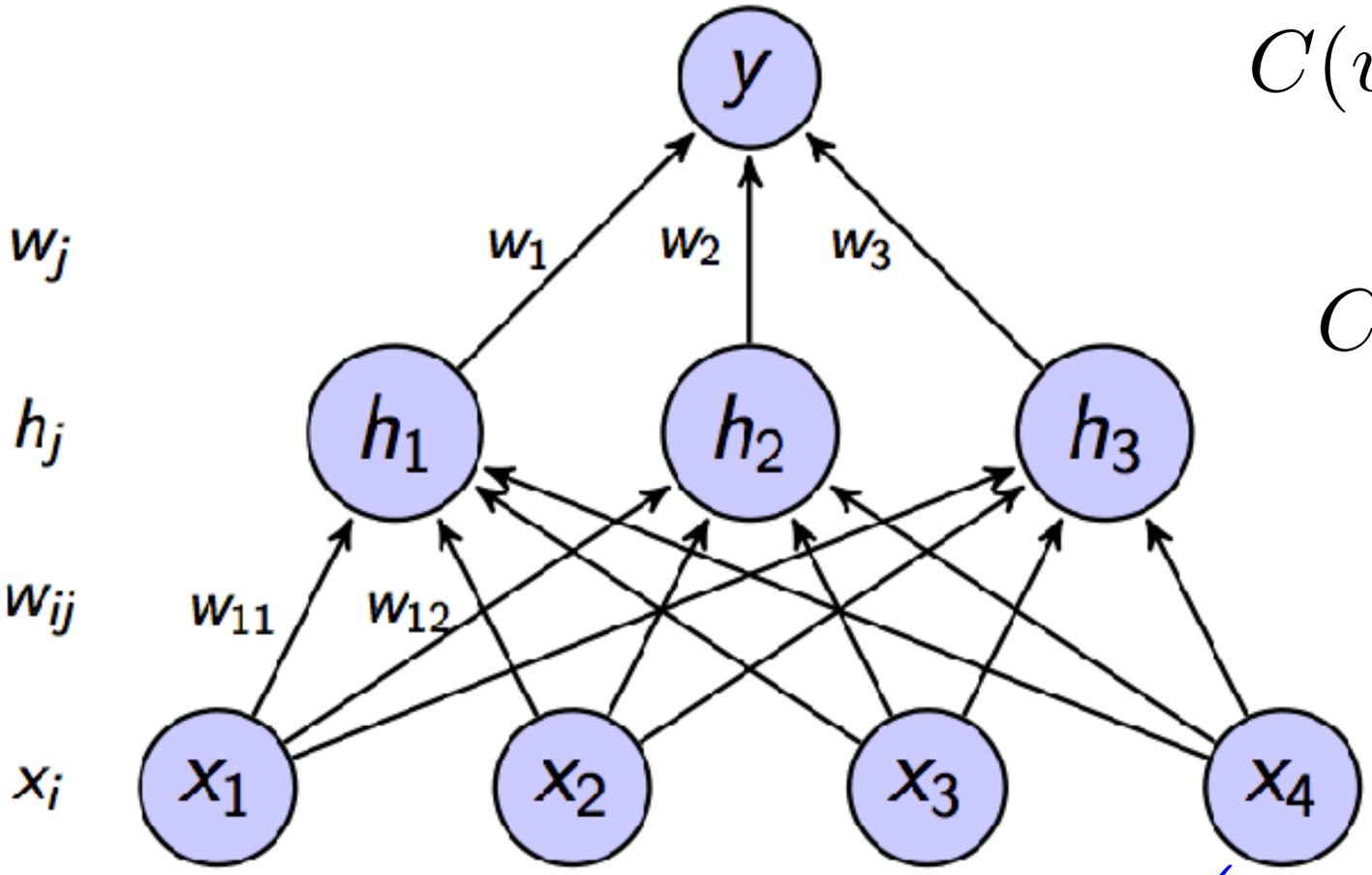


$$C(w) = \frac{1}{2}\sum_m (\sigma(w^T x^m) - y^m)^2$$

$$C(w) = \frac{1}{2}\sum_m (\sigma(in^m) - y^m)^2$$

Redes Neurais

- Redes Neurais de duas camadas ou mais (MLP)



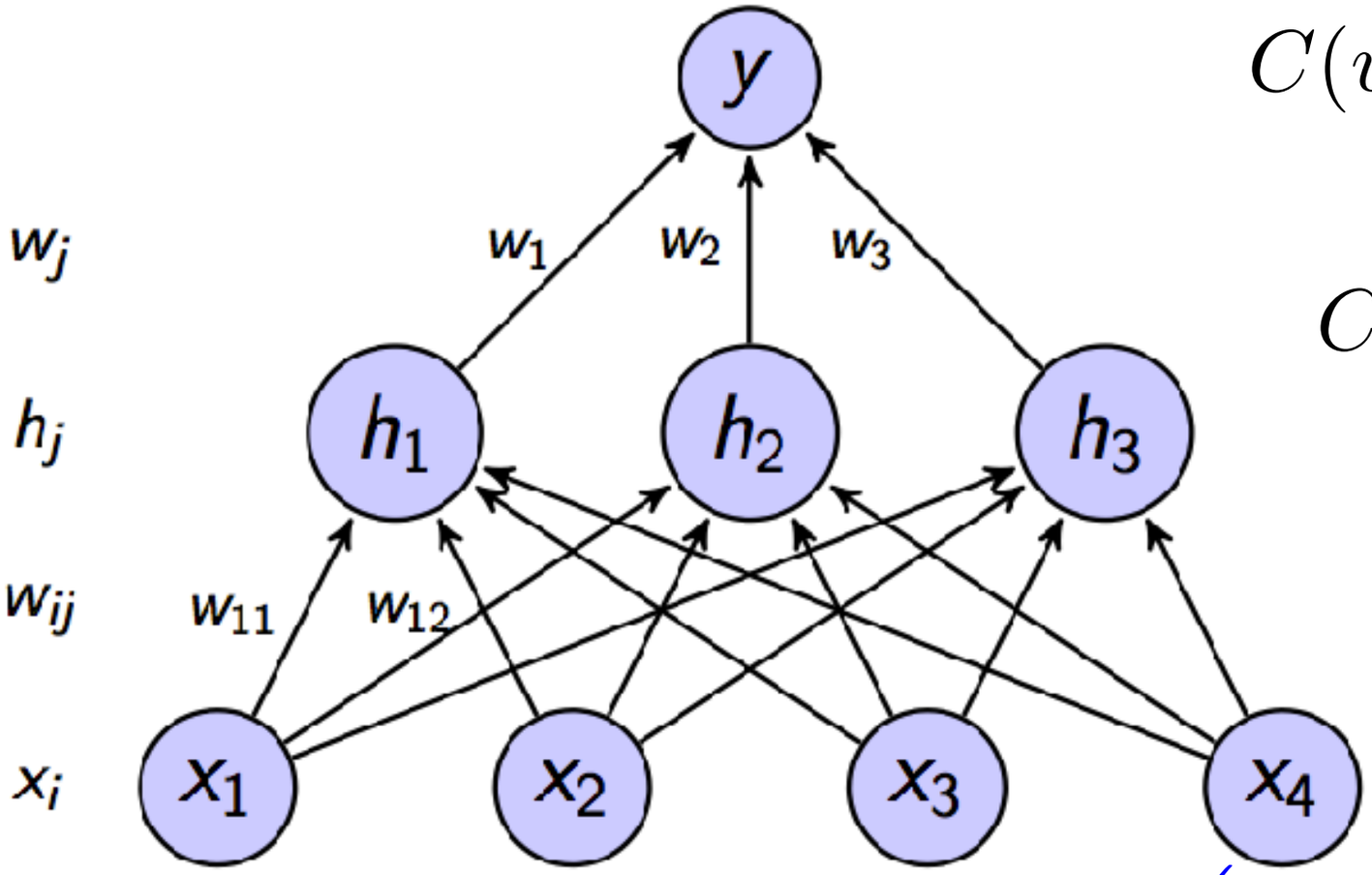
$$C(w) = \frac{1}{2}\sum_m (\sigma(w^T x^m) - y^m)^2$$

$$C(w) = \frac{1}{2}\sum_m (\sigma(in^m) - y^m)^2$$

$$\frac{\partial C(w)}{\partial w_{jk}} = \frac{\partial C(w)}{\partial in_k}\frac{\partial in_k}{\partial w_{jk}} = \delta_k \frac{\partial\left(\sum_j w_{jk} h_j\right)}{\partial w_{jk}} = \delta_k h_j$$

Redes Neurais

- Redes Neurais de duas camadas ou mais (MLP)



$$C(w) = \frac{1}{2}\sum_m (\sigma(w^T x^m) - y^m)^2$$

$$C(w) = \frac{1}{2}\sum_m (\sigma(in^m) - y^m)^2$$

$$\frac{\partial C(w)}{\partial w_{jk}} = \frac{\partial C(w)}{\partial in_k}\frac{\partial in_k}{\partial w_{jk}} = \delta_k \frac{\partial\left(\sum_j w_{jk} h_j\right)}{\partial w_{jk}} = \delta_k h_j$$

$$\frac{\partial C(w)}{\partial w_{ij}} = \frac{\partial C(w)}{\partial in_j}\frac{\partial in_j}{\partial w_{ij}} = \delta_j \frac{\partial\left(\sum_j w_{ij} x_i\right)}{\partial w_{ij}} = \delta_j x_i$$

Redes Neurais

- Redes Neurais de duas camadas ou mais (MLP)

$$\frac{\partial C(w)}{\partial w_{jk}} = \frac{\partial C(w)}{\partial in_k}\frac{\partial in_k}{\partial w_{jk}} = \delta_k \frac{\partial\left(\sum_j w_{jk} h_j\right)}{\partial w_{jk}} = \delta_k h_j$$

$$\frac{\partial C(w)}{\partial w_{ij}} = \frac{\partial C(w)}{\partial in_j}\frac{\partial in_j}{\partial w_{ij}} = \delta_j \frac{\partial\left(\sum_j w_{ij} x_i\right)}{\partial w_{ij}} = \delta_j x_i$$

$$\delta_k = \frac{\partial}{\partial in_k}\left(\frac{1}{2}\left[\sigma(in_k) - y_k\right]^2\right) = \left[\sigma(in_k) - y_k\right]\sigma'(in_k)$$

$$\delta_j = \sum_k \frac{\partial C(w)}{\partial in_k}\frac{\partial in_k}{\partial in_j} = \sum_k \delta_k \cdot \frac{\partial}{\partial in_j}\left(\sum_j w_{jk}\delta(in_j)\right) = \left[\sum_k \delta_k w_{jk}\right]\sigma'(in_j)$$

Redes Neurais

- Redes Neurais de duas camadas ou mais (MLP)
  - As atualizações seguem o formato **erro de saída escalado** * **valor de entrada**
    - $\frac{\partial C(w)}{\partial w_{jk}} = \delta_k h_j$ onde: $\delta_k = [\sigma(in_k) - y_k]\,\sigma'(in_k)$
    - $\frac{\partial C(w)}{\partial w_{ij}} = \delta_j x_i$ onde: $\delta_j = \left[\sum_k \delta_k w_{jk}\right] \sigma'(in_j)$

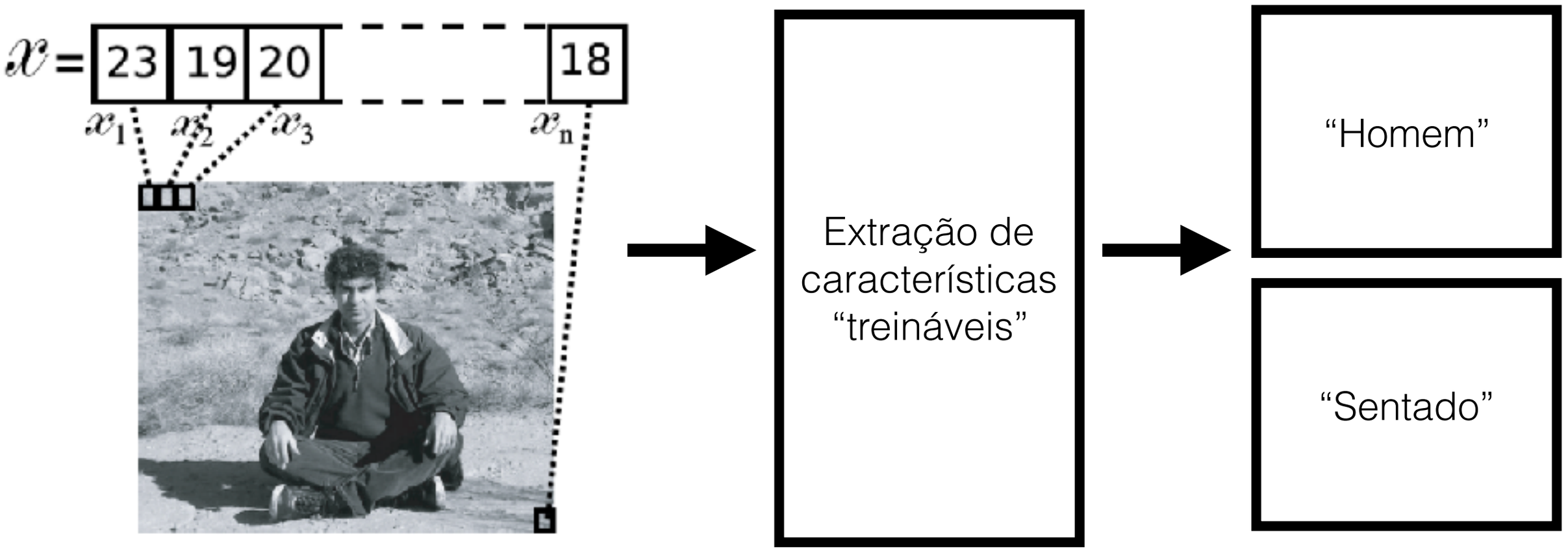
Para a atualização, primeiro se calcula δ_k (da última camada) e depois δ_j (da camada anterior)

Redes Neurais

Deep Learning

Deep Learning

- Proposta do Deep Learning



Informações básicas

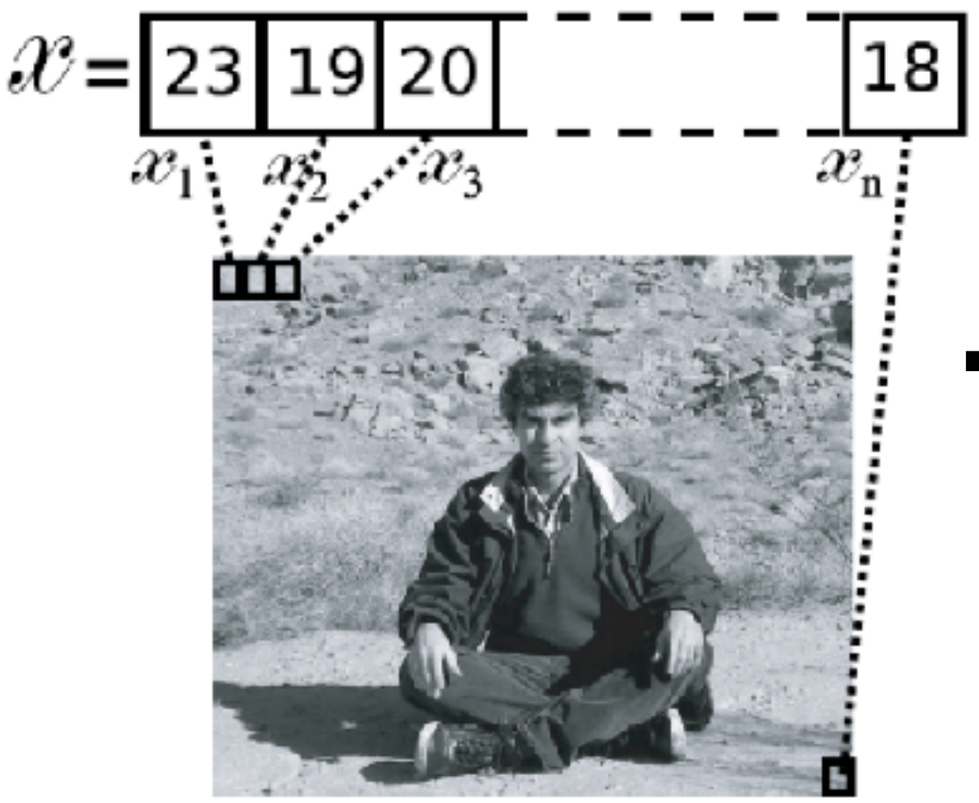
Representações gradativamente de mais alto nível

Representação em mais alto nível

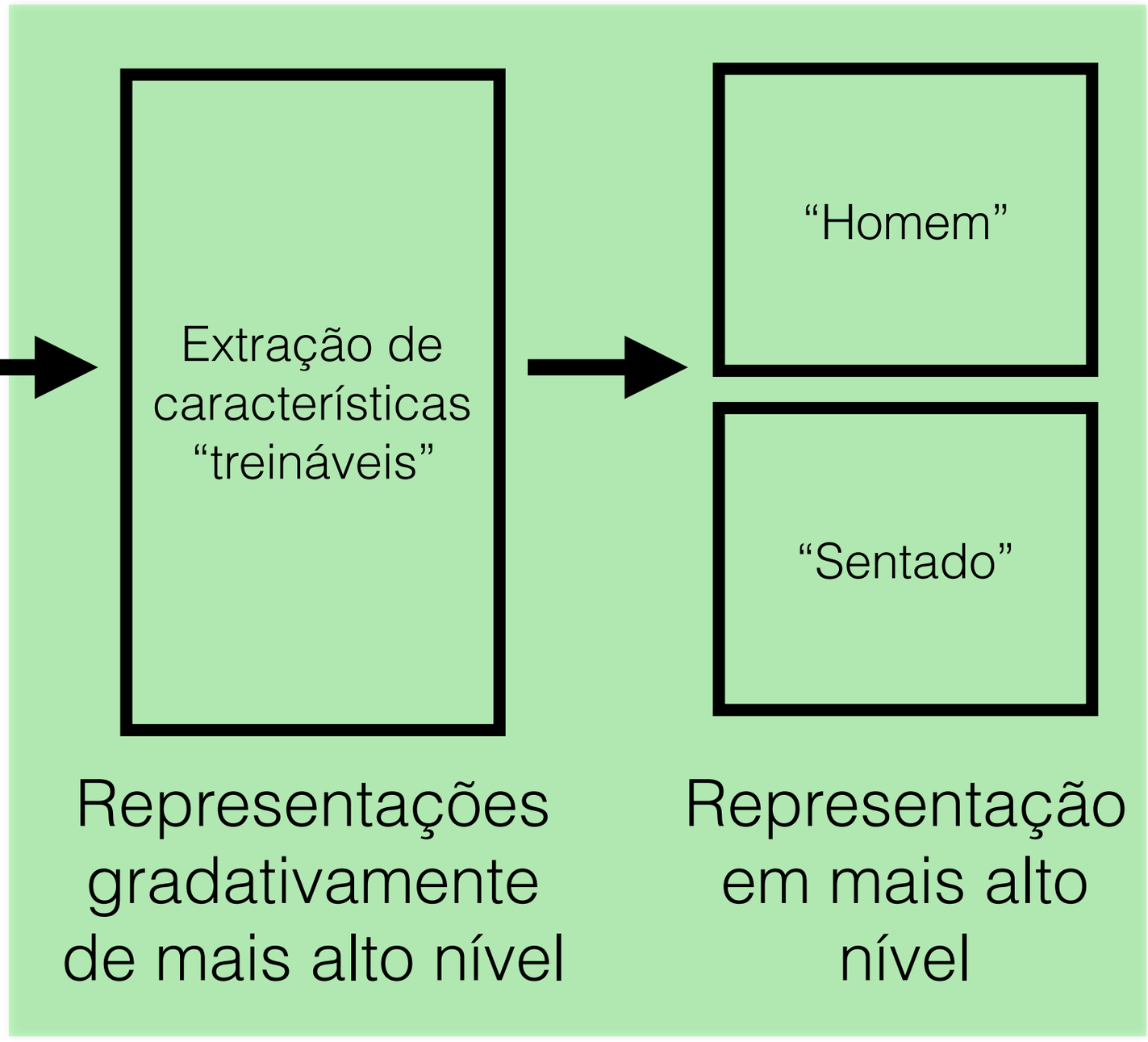
Exemplo do livro [Bengio, 2009]

Deep Learning

- Proposta do Deep Learning



Informações básicas



Exemplo do livro [Bengio, 2009]

Deep Learning

- Proposta do Deep Learning

Deep Learning

- Proposta do Deep Learning

  - A Compreensão (*Understanding*) em Algoritmos de IA (Inteligência Artificial) exige níveis de abstração que são modelados como camadas de funções não-lineares

Deep Learning

- Proposta do Deep Learning
  - A Compreensão (*Understanding*) em Algoritmos de IA (Inteligência Artificial) exige níveis de abstração que são modelados como camadas de funções não-lineares
  - As abstrações devem ser independentes de fatores de variação dos dados (como o exemplo é imagem, temos como variações: rotações, ruídos, diferenças de iluminação)

Deep Learning

- Proposta do Deep Learning

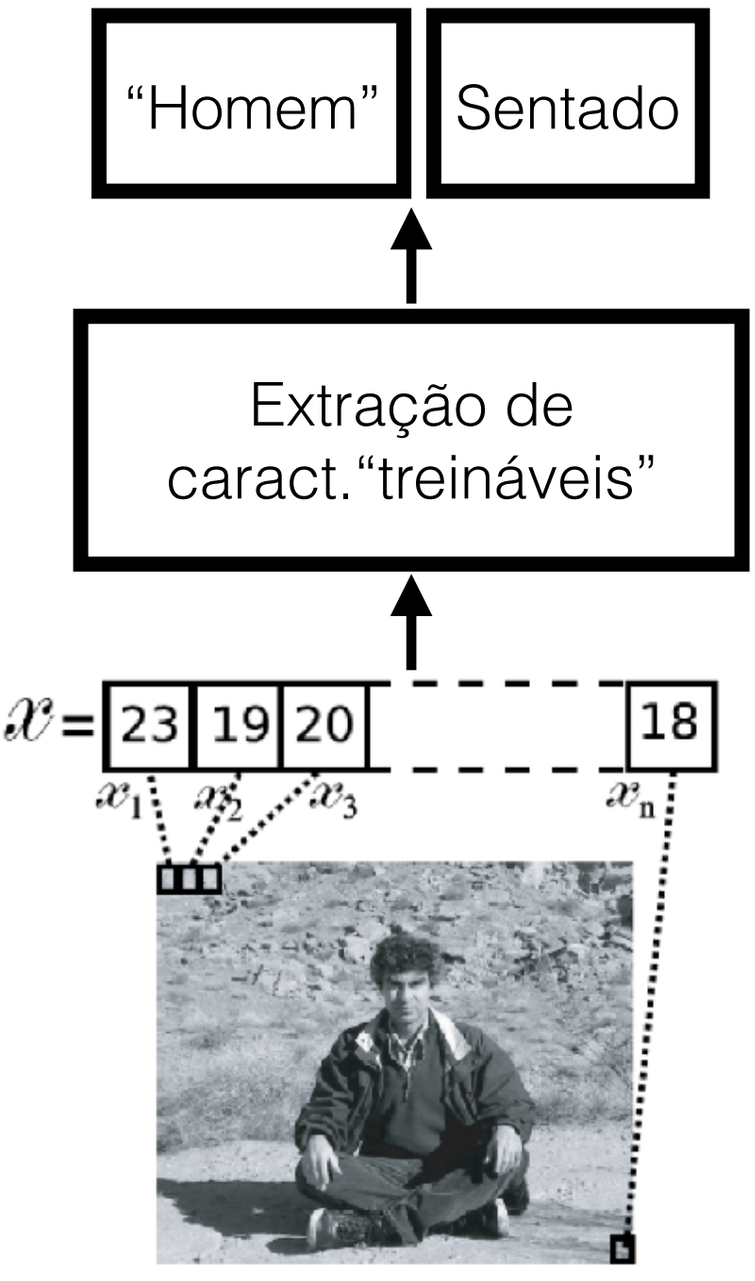
  - A Compreensão (*Understanding*) em Algoritmos de IA (Inteligência Artificial) exige níveis de abstração que são modelados como camadas de funções não-lineares

  - As abstrações devem ser independentes de fatores de variação dos dados (como o exemplo é imagem, temos como variações: rotações, ruídos, diferenças de iluminação)

  - Em Deep Learning, cada camada intermediária é um nível subsequente de abstração.

Deep Learning

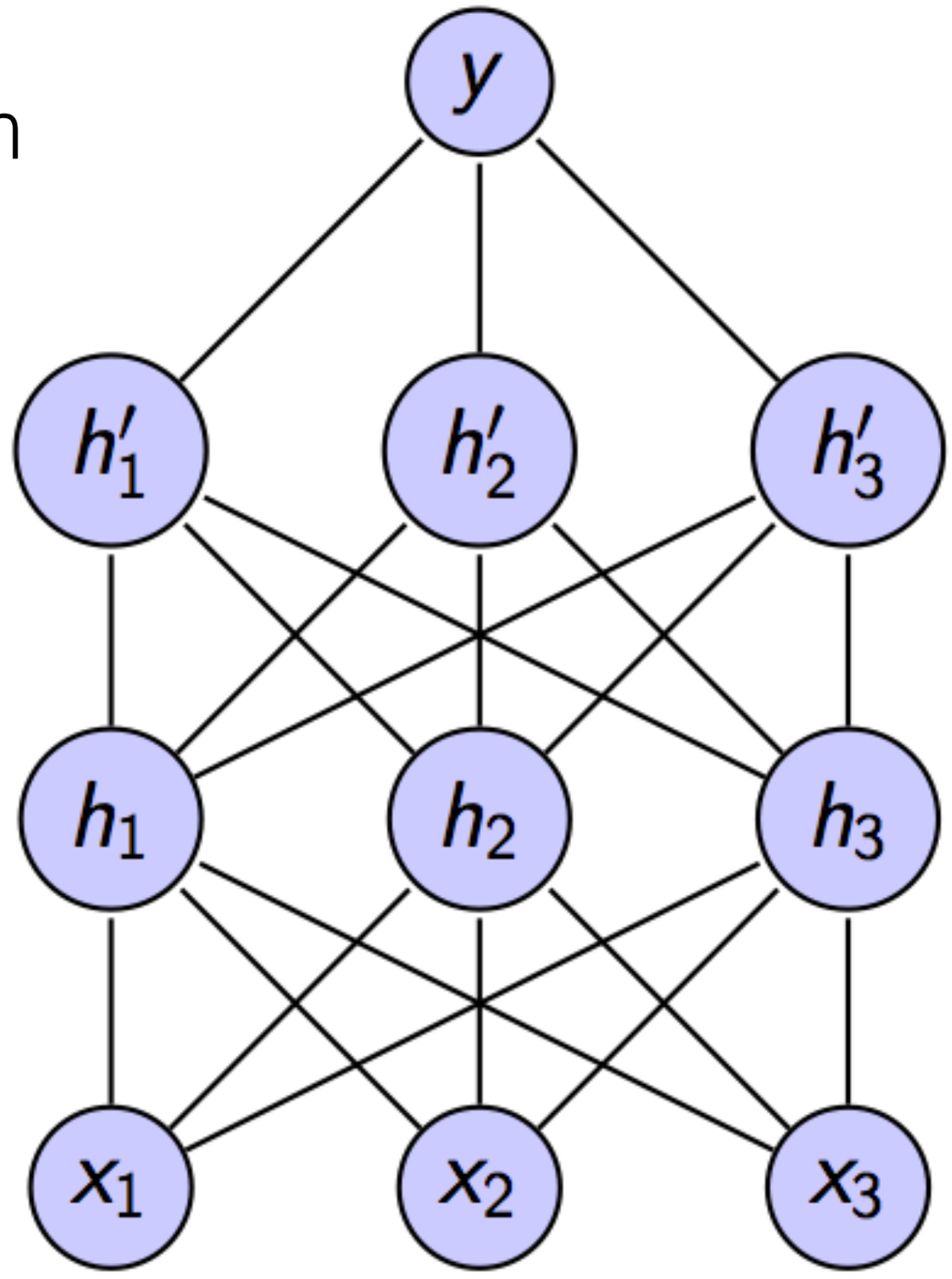
- Proposta do Deep Learning



Informação com altos Níveis de Abstração

Níveis de Abstração

Informações básicas



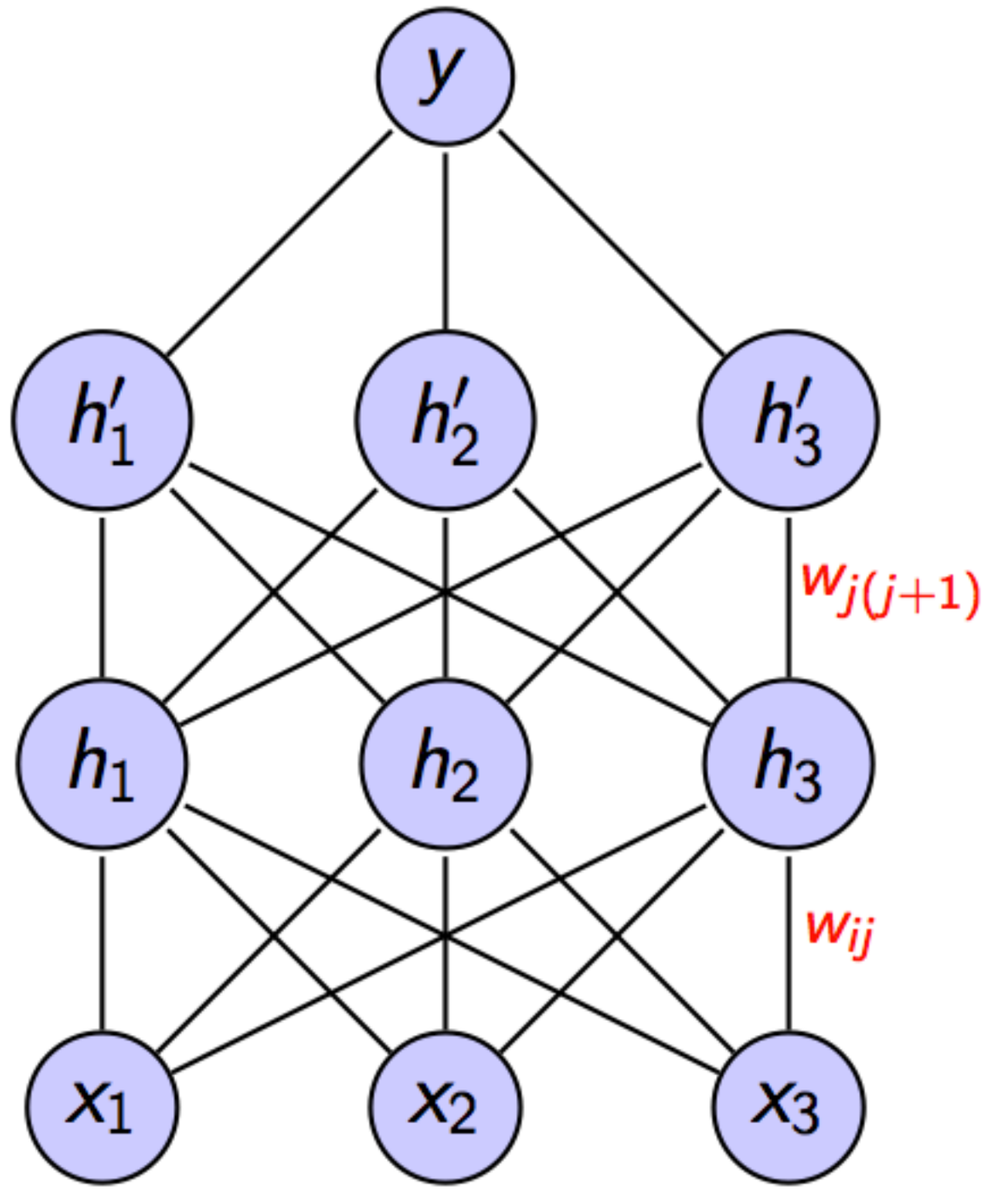
Deep Learning

- Parece ser muito bom para ser verdade…

- Problemas de treinamento

$$\frac{\partial C(w)}{\partial w_{ij}} = \frac{\partial C(w)}{\partial in_j}\frac{\partial in_j}{\partial w_{ij}} = \delta_j x_i$$

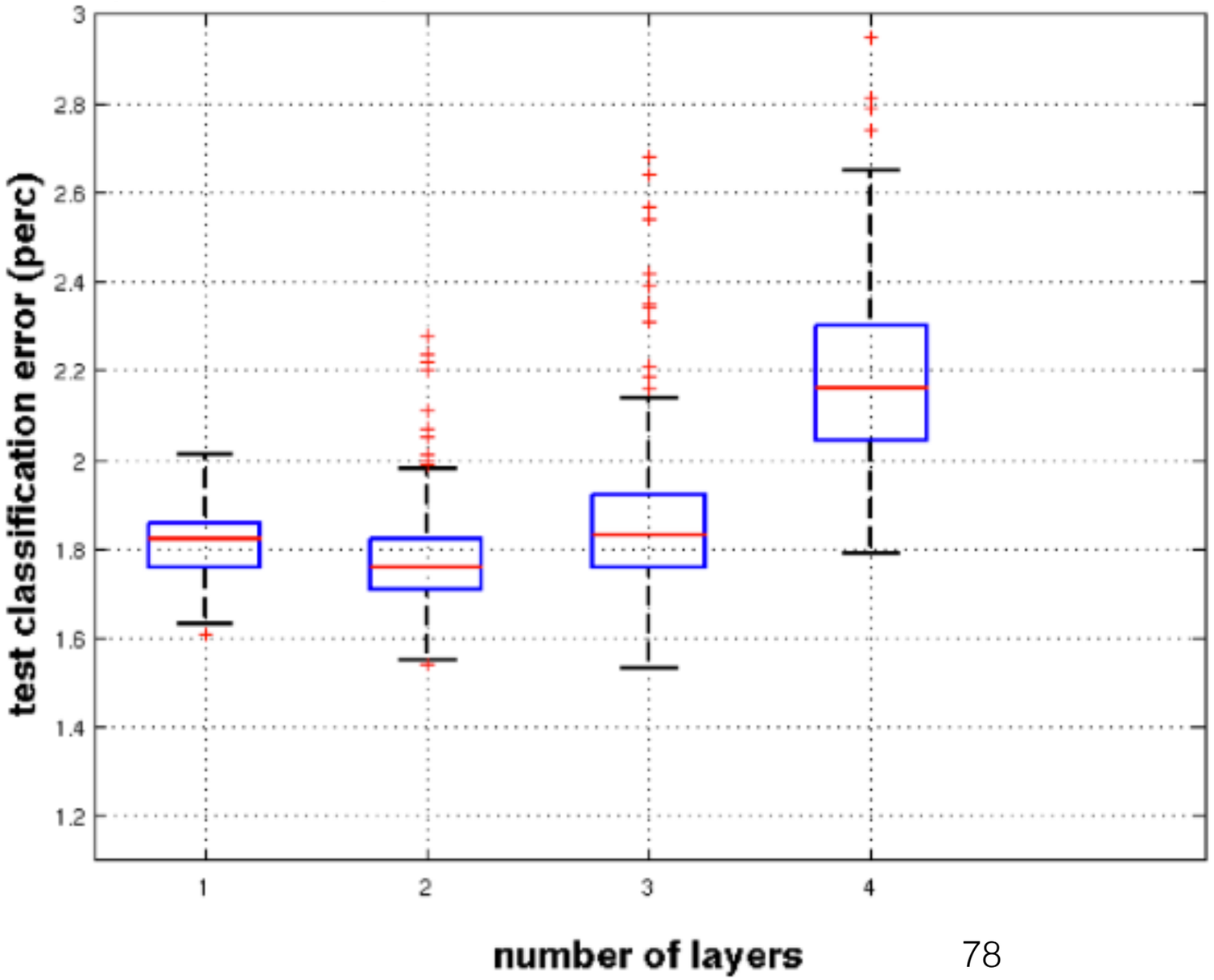
$$\delta_j = \left[\sum_{j+1} \delta_{j+1} w_{j(j+1)}\right] \sigma'(in_j)$$



- A medida que vamos nos aprofundando na arquitetura, δ_j pode diminuir muito

Deep Learning

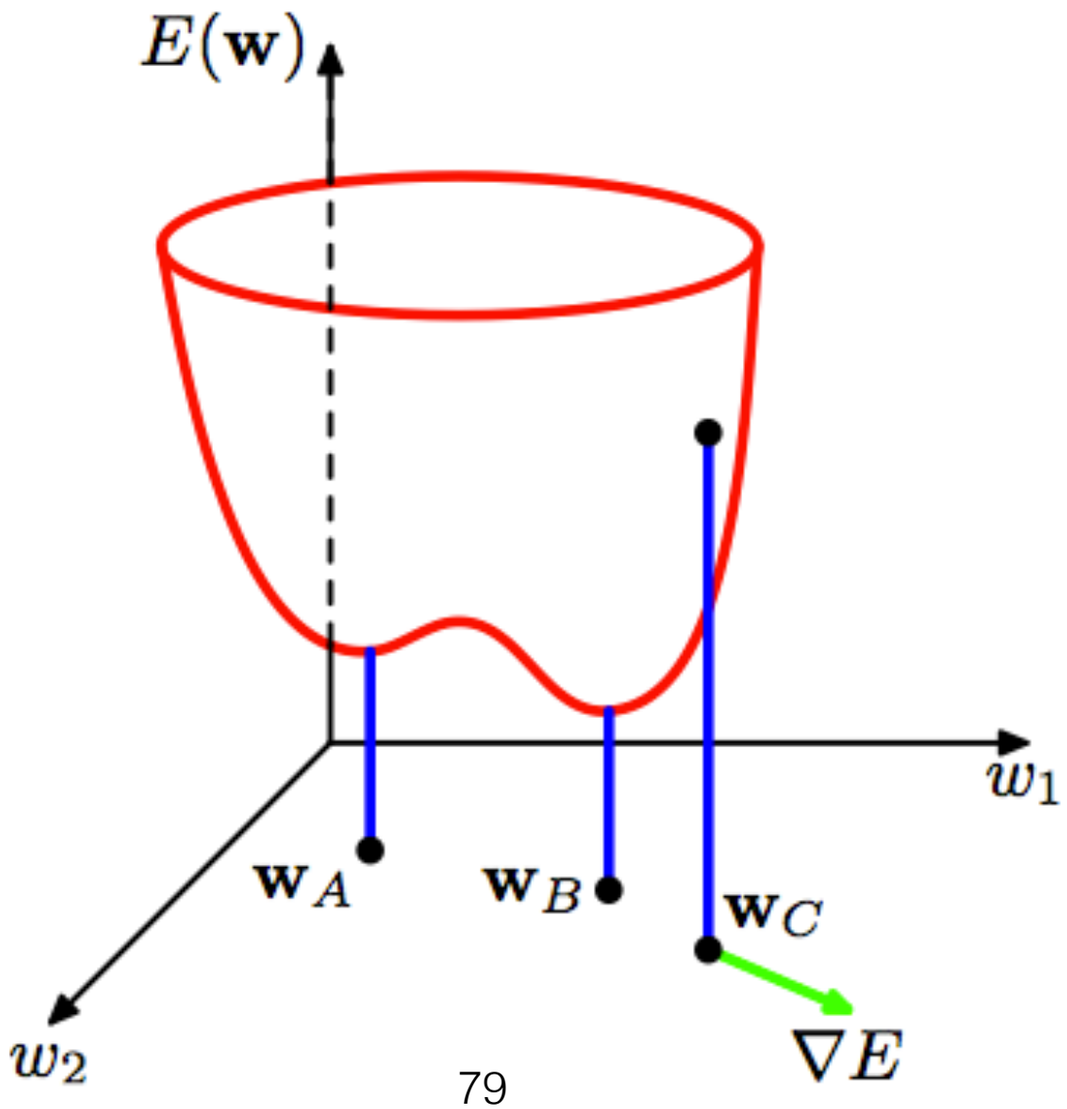
- Problemas de treinamento
  - Em [Erhan et al.,2009] o algoritmo *backpropagation* foi testado para Deep NNs



- BD de caracteres escritos a mão
- 60k imagens para treino, 10k imagens para teste
- Cada camada foi inicializada 400 vezes

Deep Learning

- Para redes de 2 camadas ou mais, a função custo é não convexa (contínua com vários mínimos globais)
- Portanto, diferentes mínimos globais podem ser obtidos com diferentes pesos iniciais
- Para Deep NNs, devido aos problemas de treinamento, aparentemente o algoritmo *backpropagation* não atinge o mínimo de melhor generalização



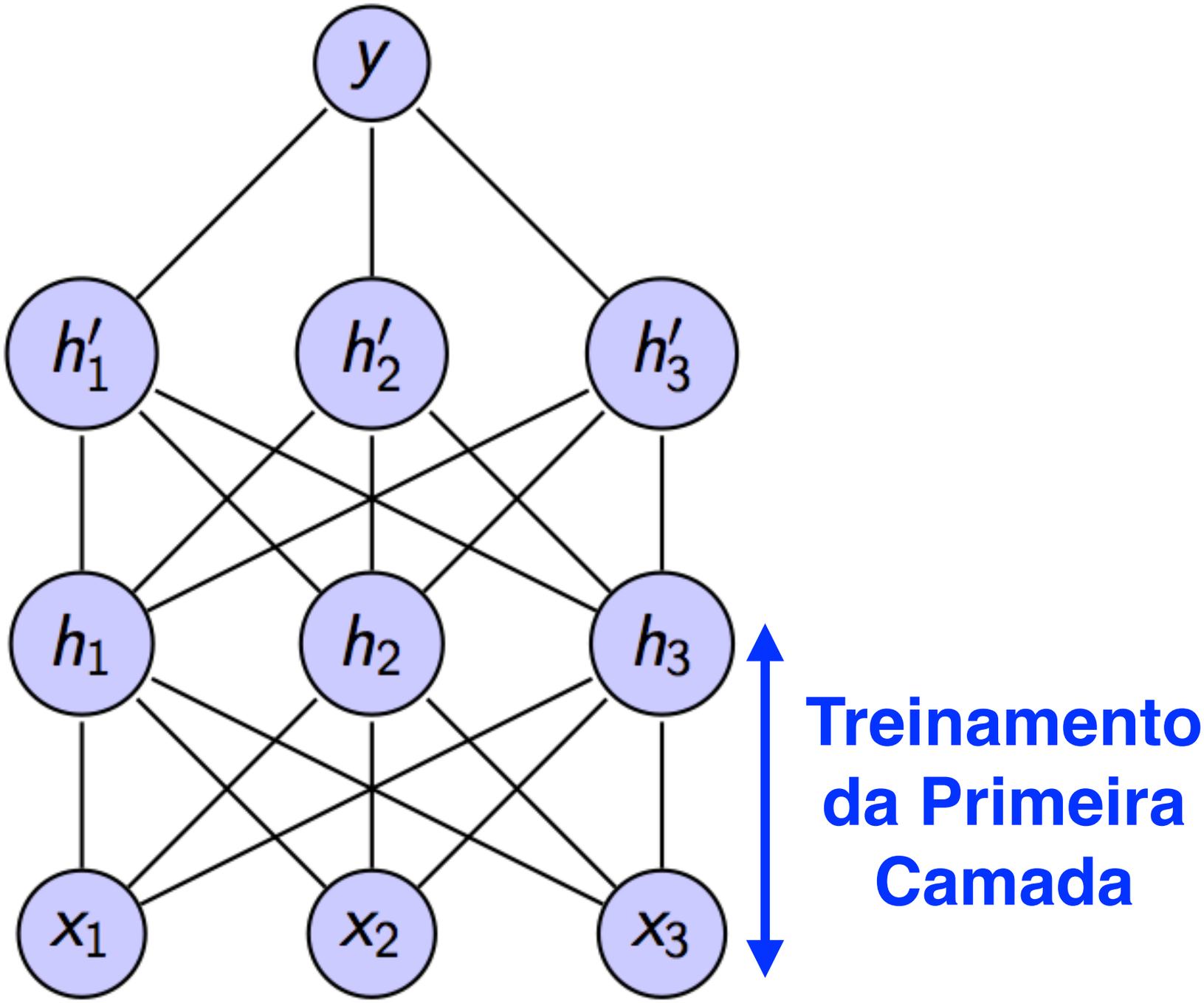
Deep Learning

- Ou seja, sem um algoritmo de treinamento eficiente, nada de Deep Neural Networks!!!

- Em 2006, uma solução: **Layer-wise Pre-Training**

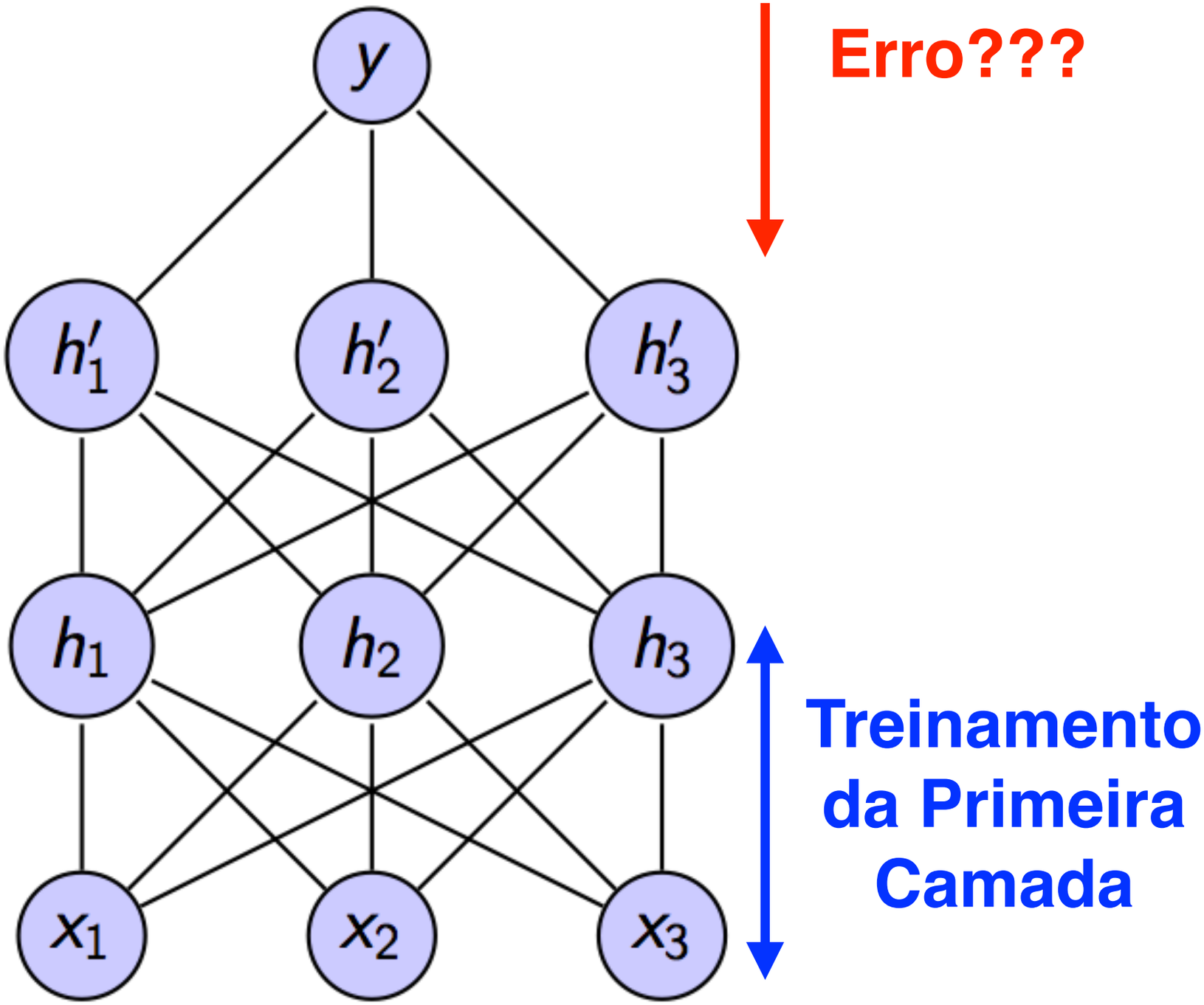
Deep Learning

- Layer-wise Pre-training [Hinton et al., 2006]



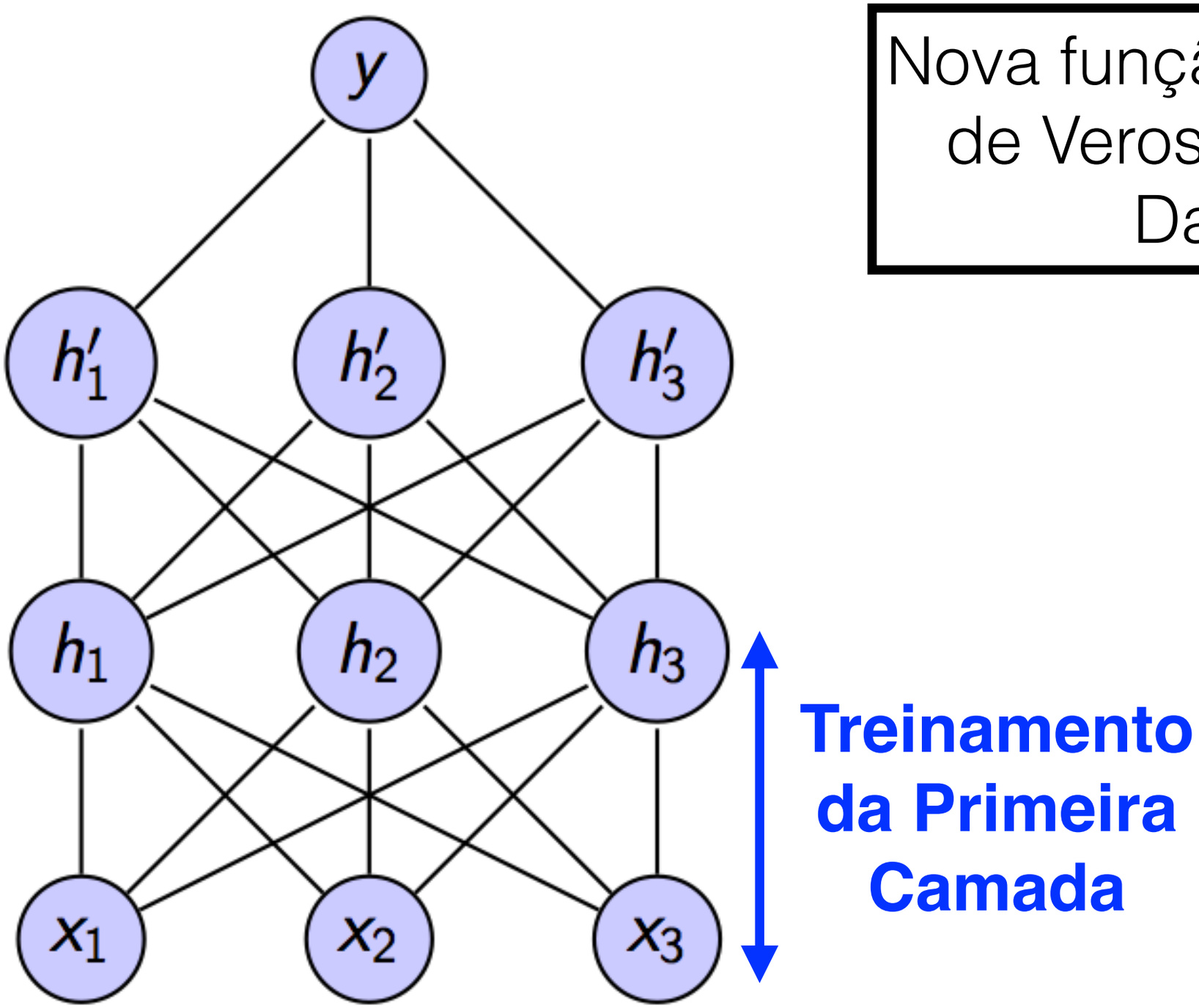
Deep Learning

- Layer-wise Pre-training [Hinton et al., 2006]



Deep Learning

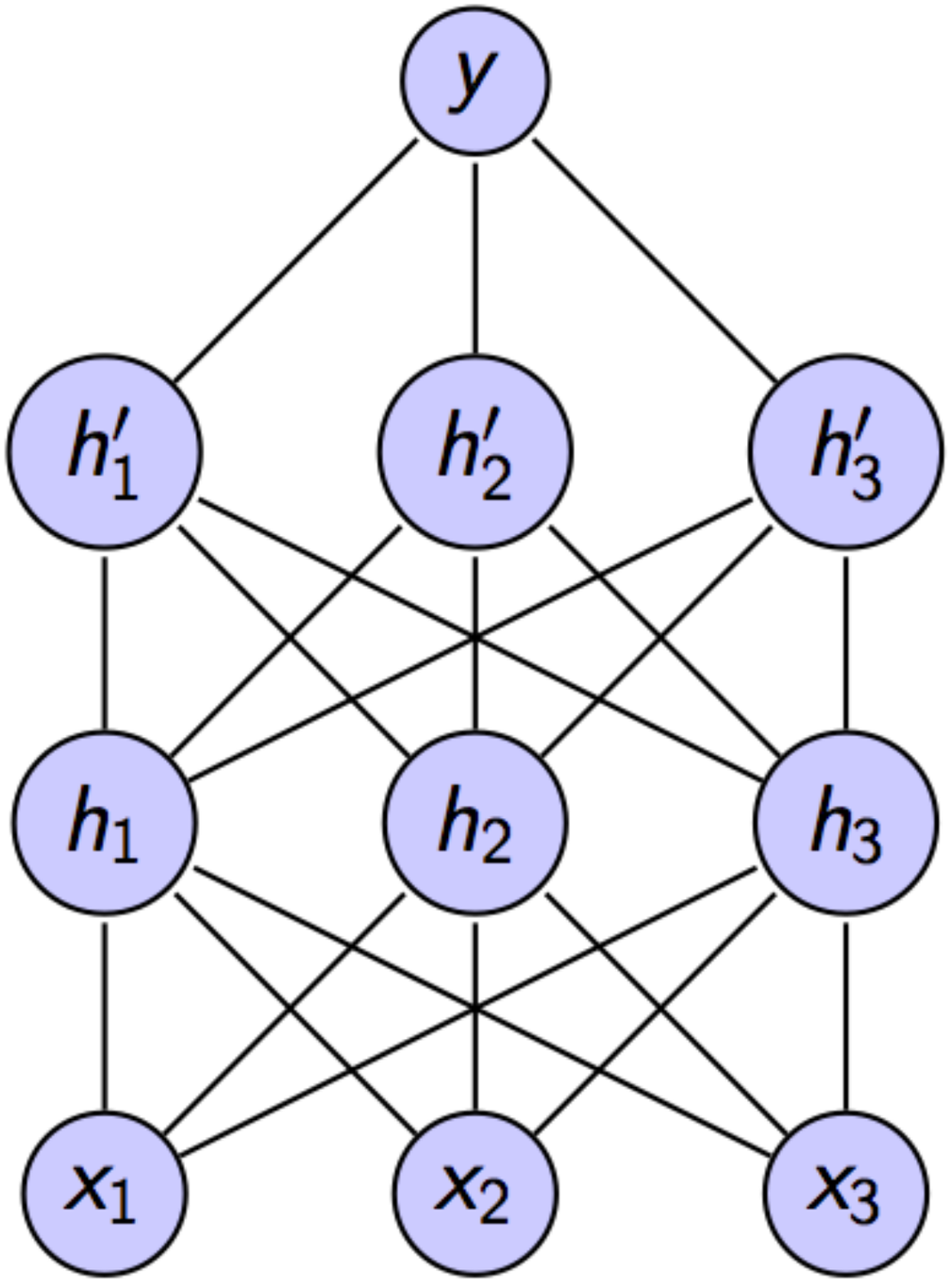
- Layer-wise Pre-training [Hinton et al., 2006]



Nova função Custo: Função de Verossimilhança dos Dados P(x)

Deep Learning

- Layer-wise Pre-training [Hinton et al., 2006]



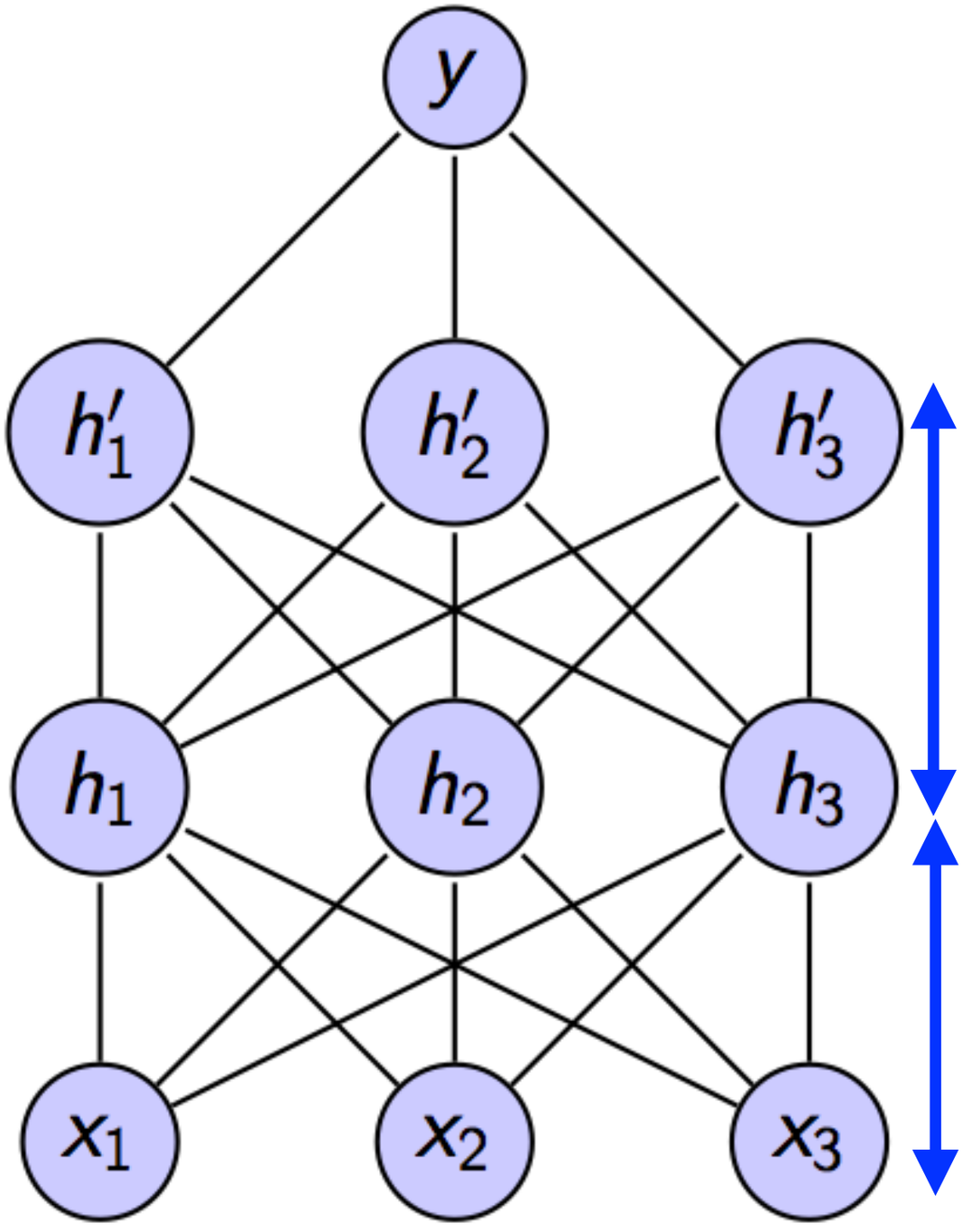
Nova função Custo: Função de Verossimilhança dos Dados P(x)

Treinados para aumentar a probabilidade de observar os dados

Treinamento da Primeira Camada

Deep Learning

- Layer-wise Pre-training [Hinton et al., 2006]



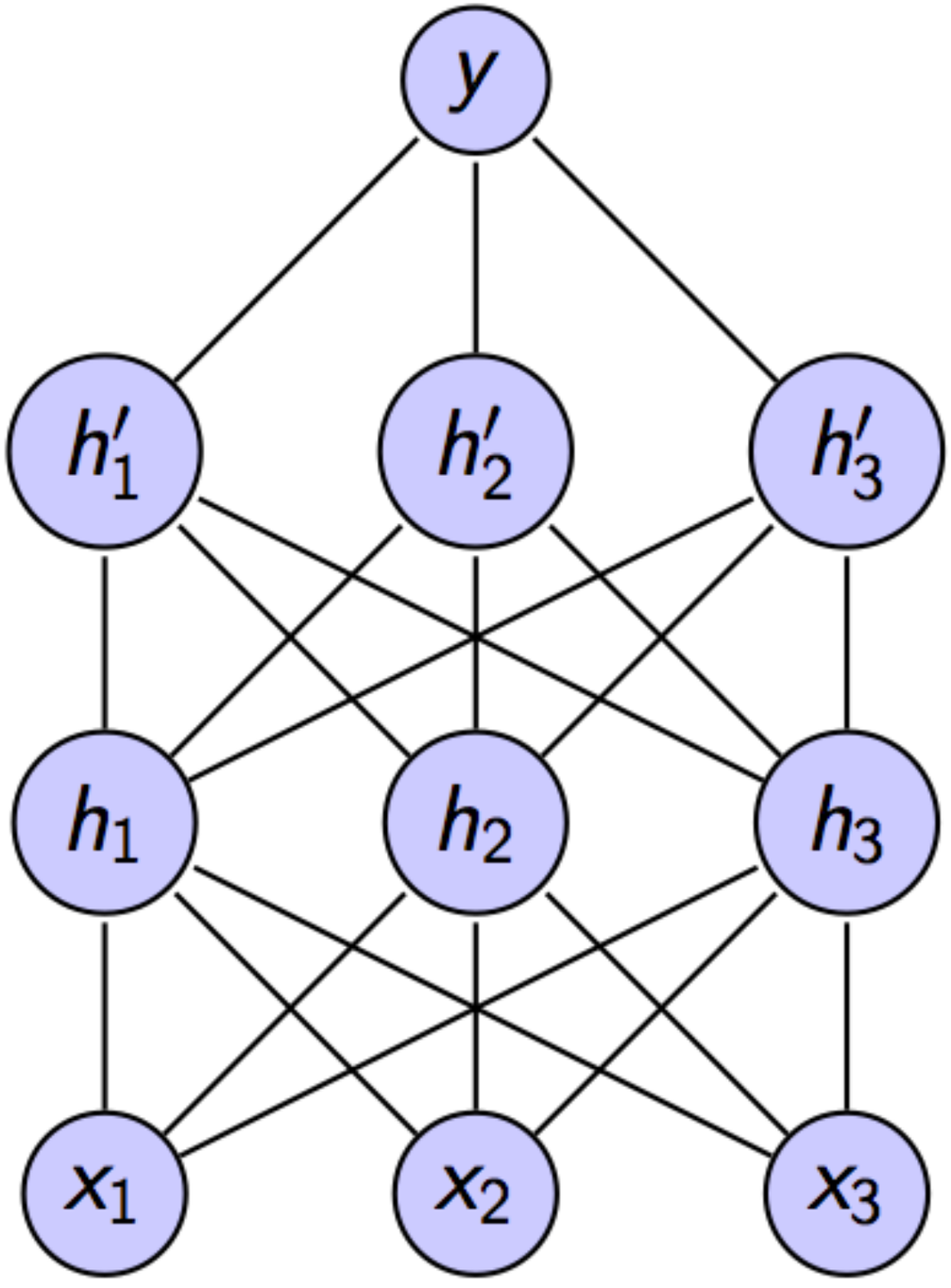
Nova função Custo: Função de Verossimilhança dos Dados P(x)

Treinamento da Segunda Camada

Fixa a Primeira Camada

Deep Learning

- Layer-wise Pre-training [Hinton et al., 2006]



Propaga a entrada

Ajusta os pesos para a função P(Y I X)

Deep Learning

- Mas por que *Layer-wise Pre-training* funciona???

- Uma hipótese assumida por Bengio, 2009 e Erhan et al., 2010, **“Uma Deep NN pode se ajustar aos dados de treinamento de diversas maneiras” (Solução não-convexa)**

  - Como?

    - Otimizando as camadas mais baixas (informações mais “cruas”)

    - Otimizando as camadas mais altas (informações com maiores níveis de abstração)

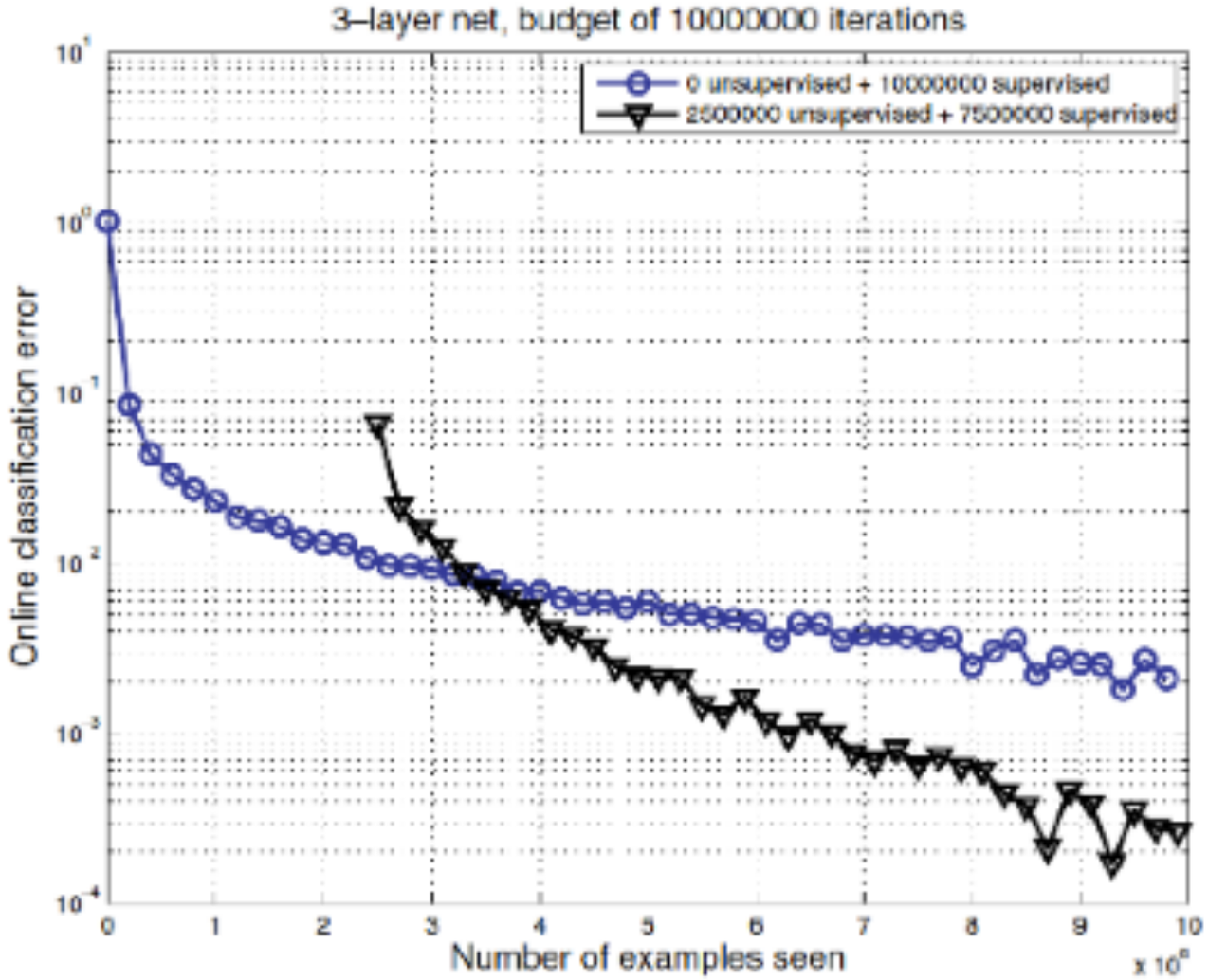
- Mesmo que inicializemos os pesos das camadas mais baixas com **valores aleatórios**, as camadas superiores ainda poderiam se ajustar aos dados. Embora esta rede não possua poder de generalização

- A razão para o *Layer-wise Pre-training* funcionar é que este algoritmo garante que as camadas mais baixas **representem** bem os dados de entrada

Deep Learning

- Mas por que *Layer-wise Pre-training* funciona???

- Uma hipótese assumida por Bengio, 2009 e Erhan et al., 2010, **“Uma Deep NN pode se ajustar aos dados de treinamento de diversas maneiras” (Solução não-convexa)**

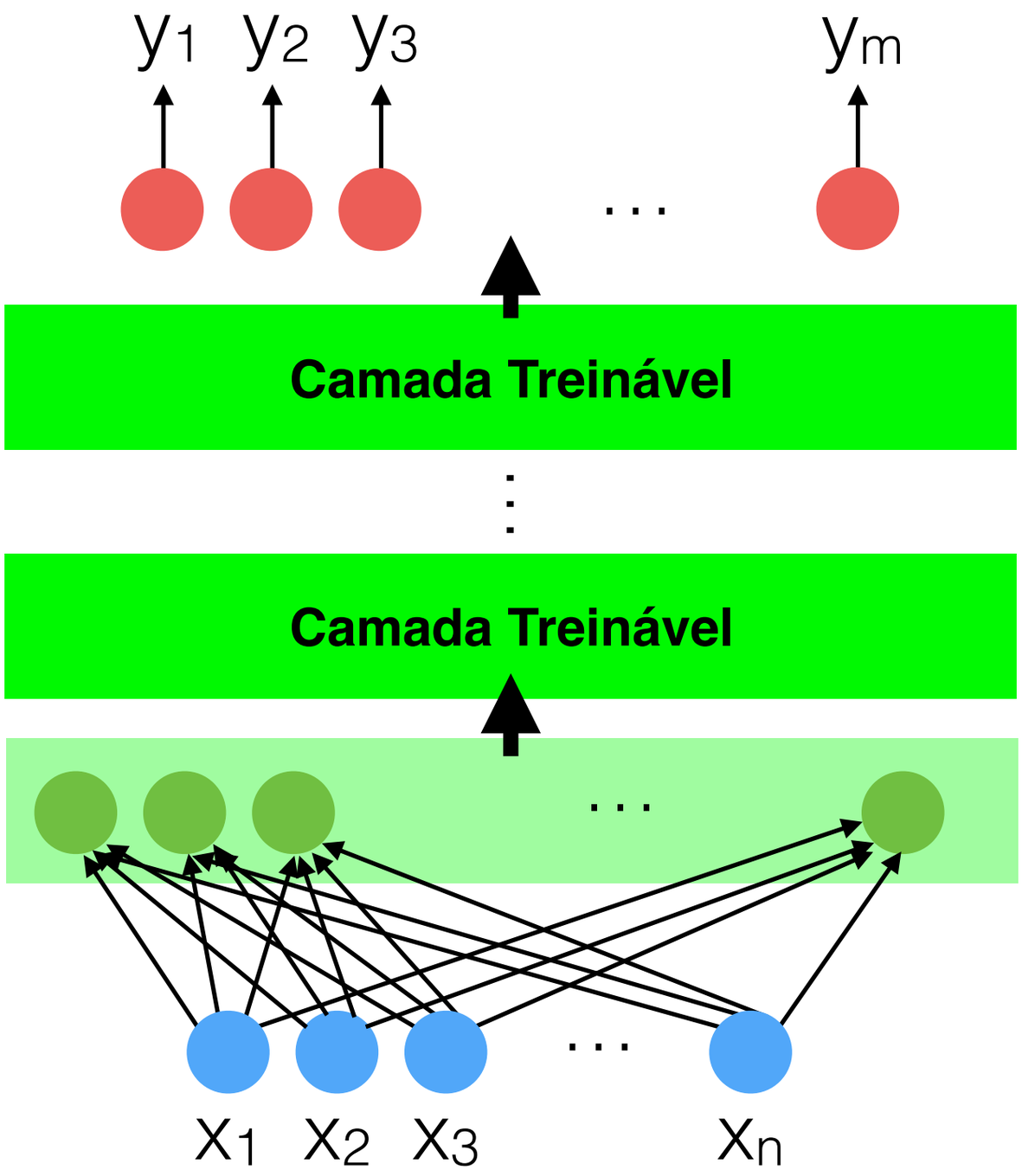


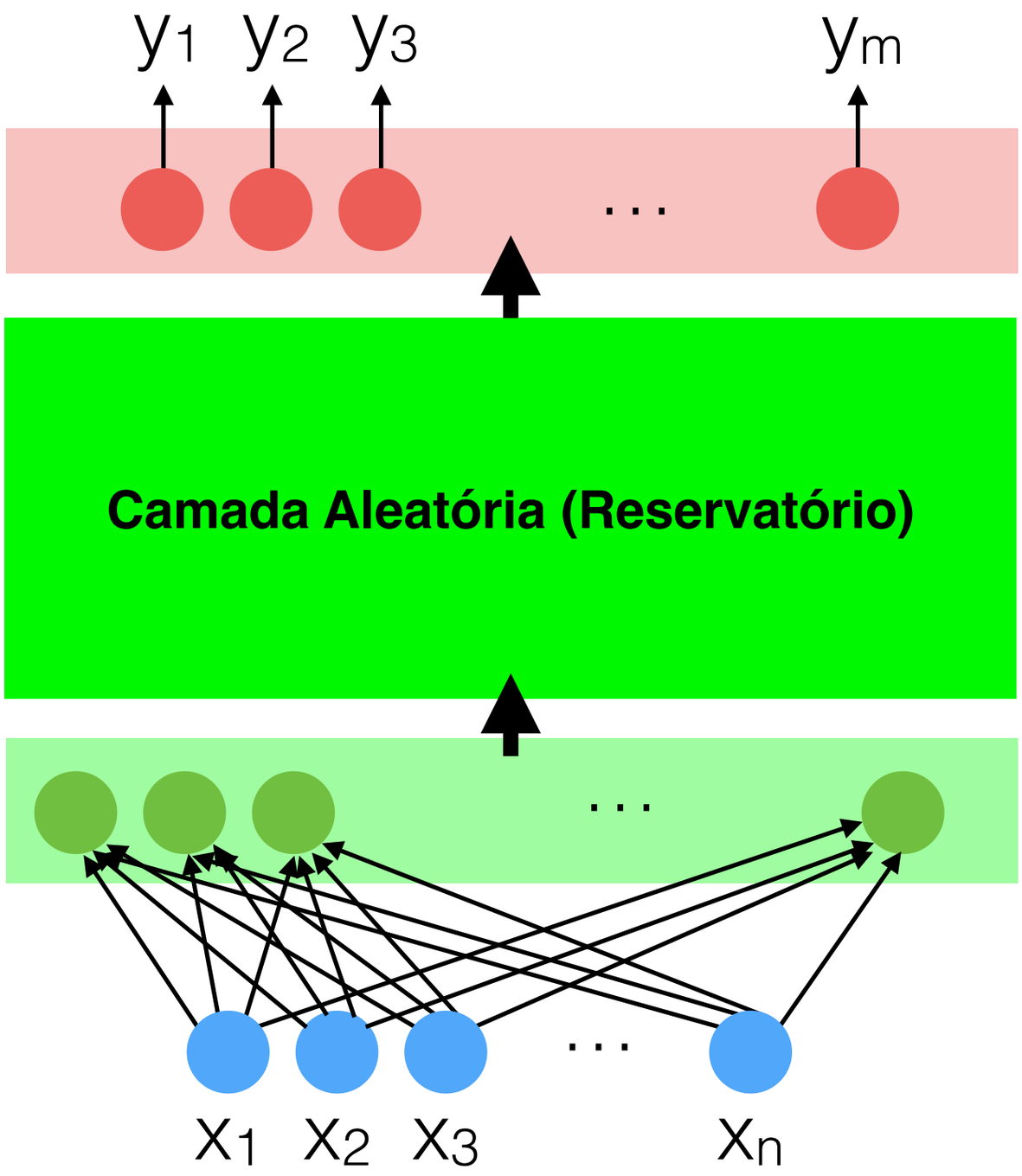
- Azul - Rede Neural com 3 camadas e *backpropagation*

- Preto - Rede Neural com 3 camadas, iniciada com Layer-wise pre-training e treinada com *backpropagation*

Deep Learning

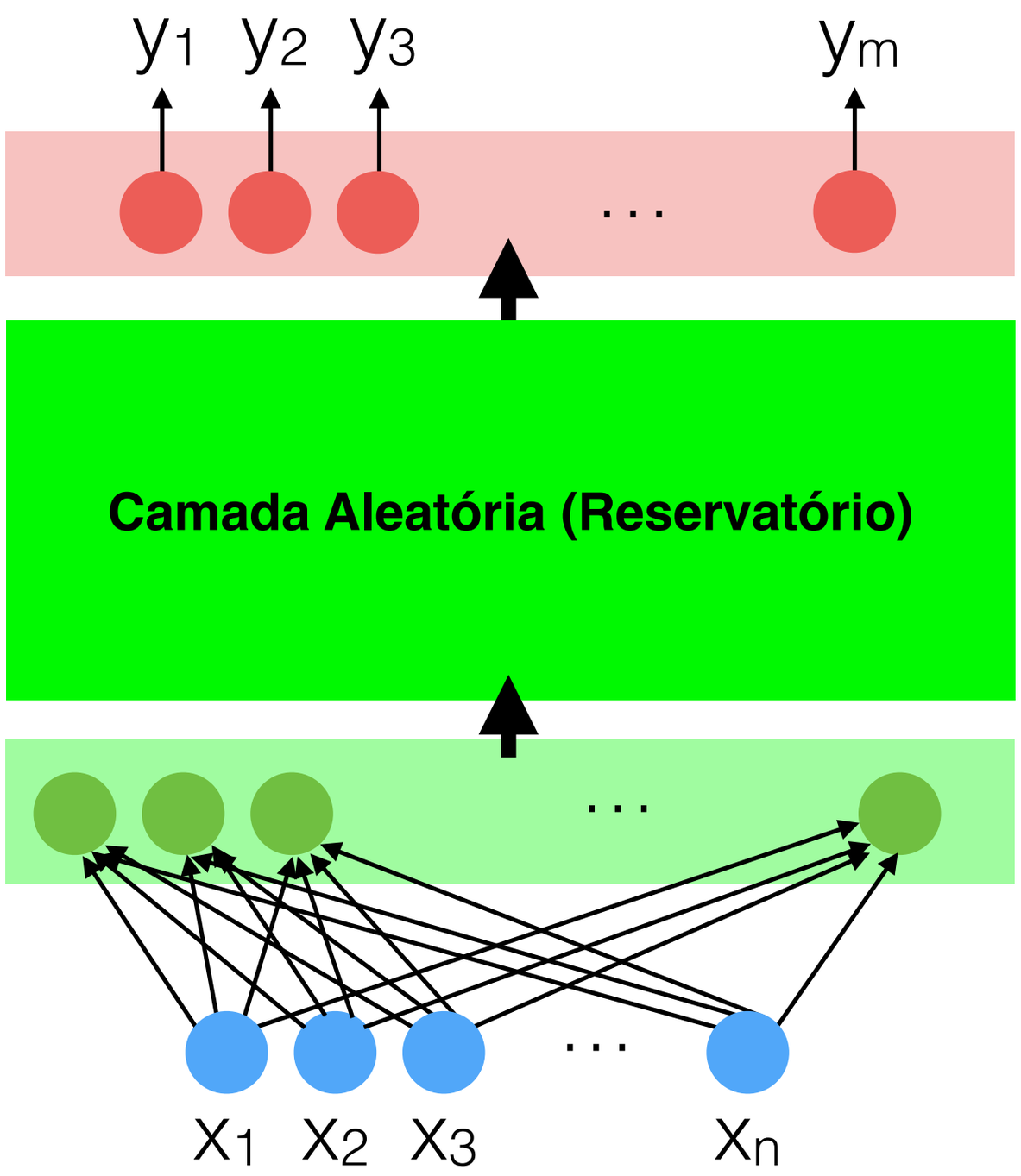
- Deep Learning vs Reservoir Computing

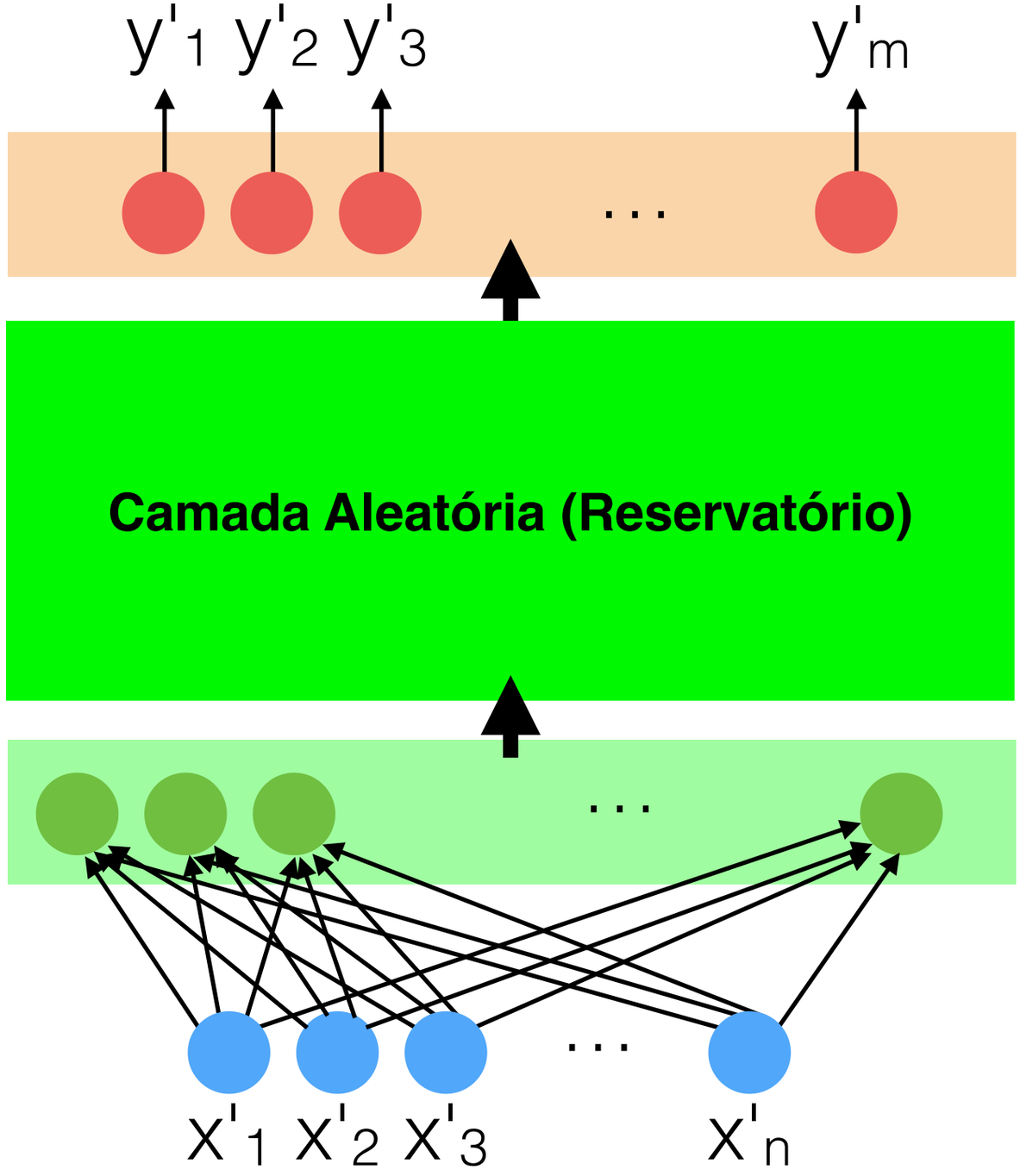




Deep Learning

- Deep Learning vs Reservoir Computing

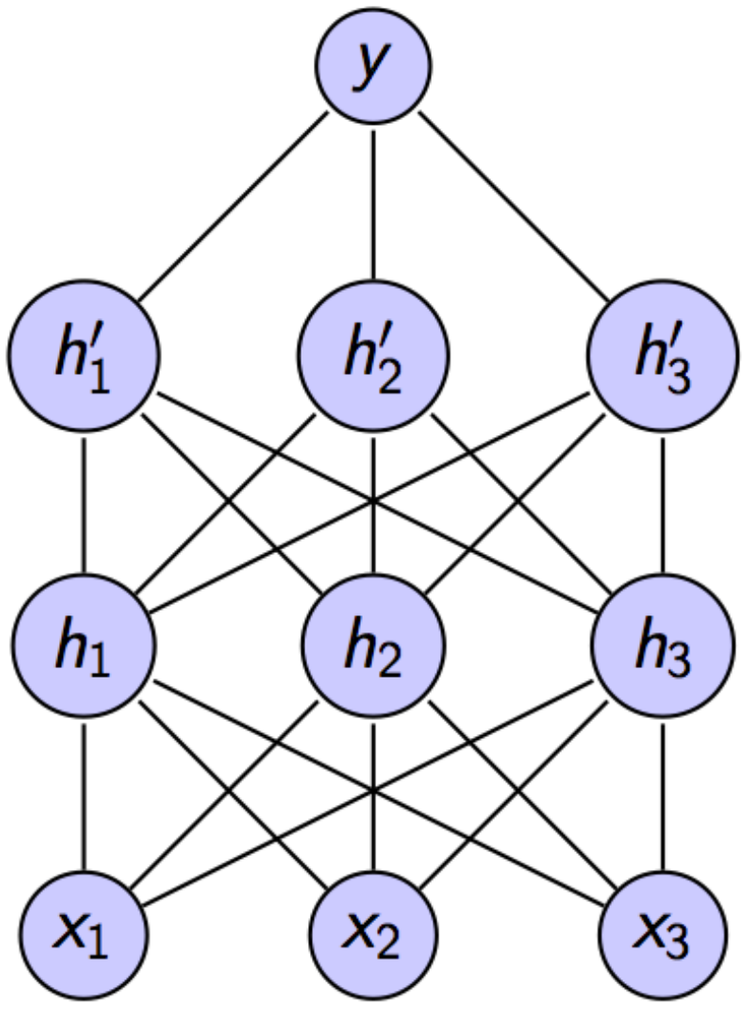




Deep Learning

- Conceito chave: *"first things first"*, [Garland, 1964]

- Primeiro a rede deve propagar corretamente os dados e somente depois disso, a mesma deve tentar a classificação

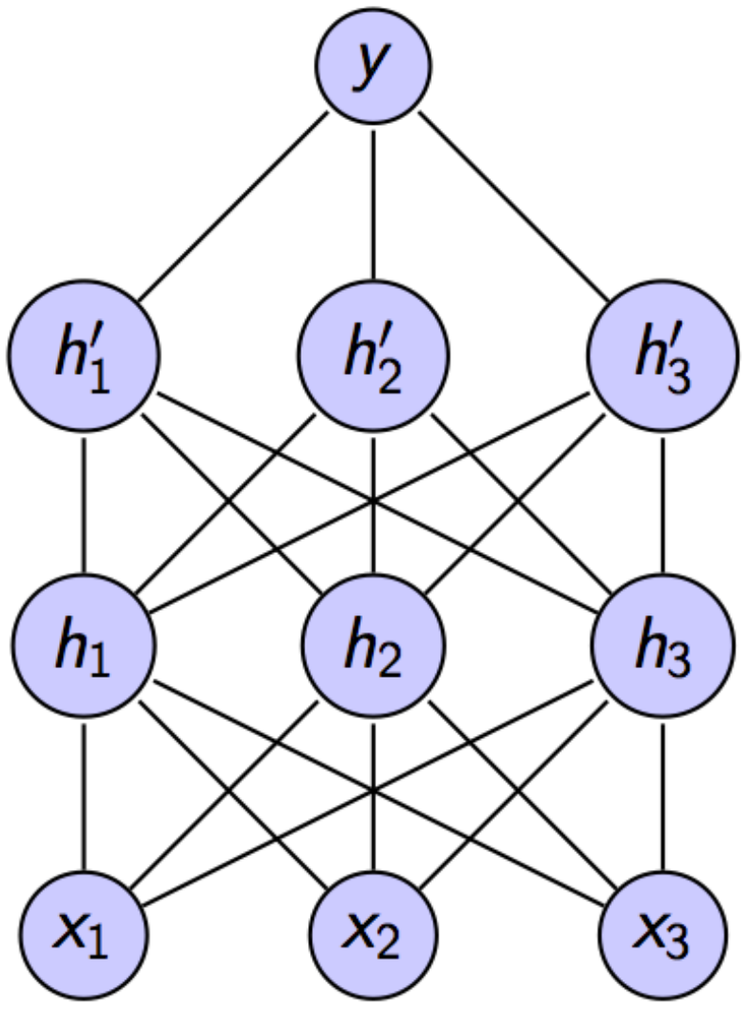


"If you want to do computer vision, first learn computer graphics" [Geoff Hinton]

Deep Learning

- Conceito chave: *"first things first"*, [Garland, 1964]

- Primeiro a rede deve propagar corretamente os dados e somente depois disso, a mesma deve tentar a classificação



"If you want to do computer vision, first learn computer graphics" [Geoff Hinton]

Outra vantagem é: podemos trabalhar com dados sem a informação de classificação do especialista

Deep Learning

- Discussão: *Layer-wise Pre-training* é sempre necessário?

- O que é o overfit?

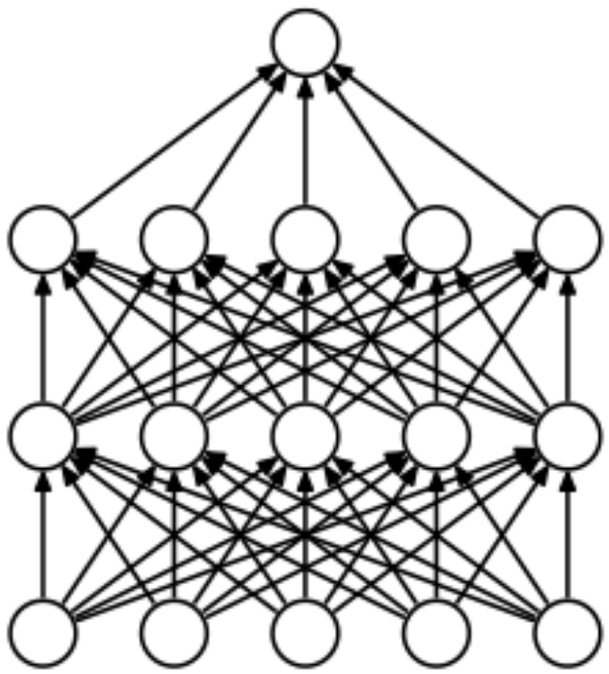
- Se temos acesso a toda estatística do processo, o overfit não é um problema.

- Quase nunca temos acesso a toda estatística do processo…

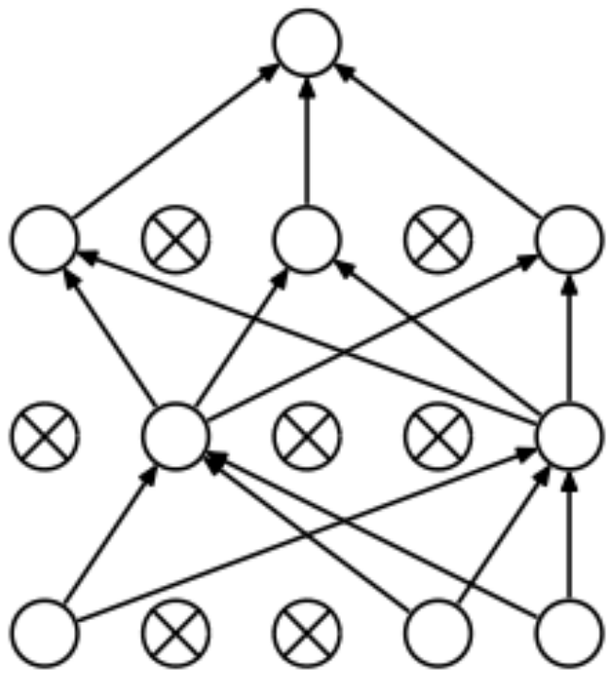
- Exceto em casos específicos: CERN (muita estatística mas não toda…)

Deep Learning

- Dropout
  - Uma outra maneira de evitar o *overfitting* durante o processo de treinamento (Srivastava et al., 2014)
  - Idéia básica: Aleijar a rede neural removendo estocasticamente pesos do processo de treinamento



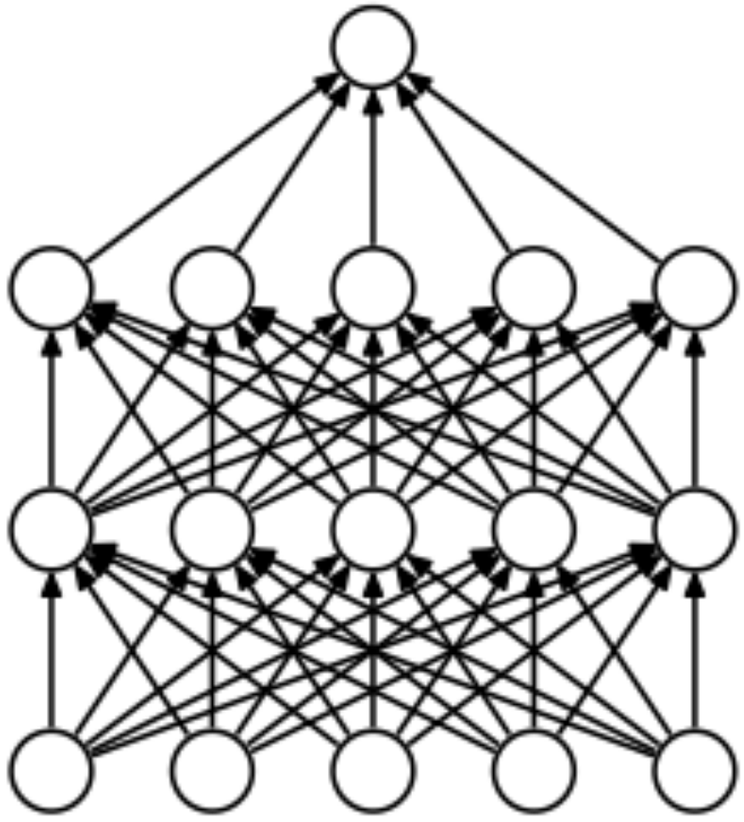
(a) Standard Neural Net



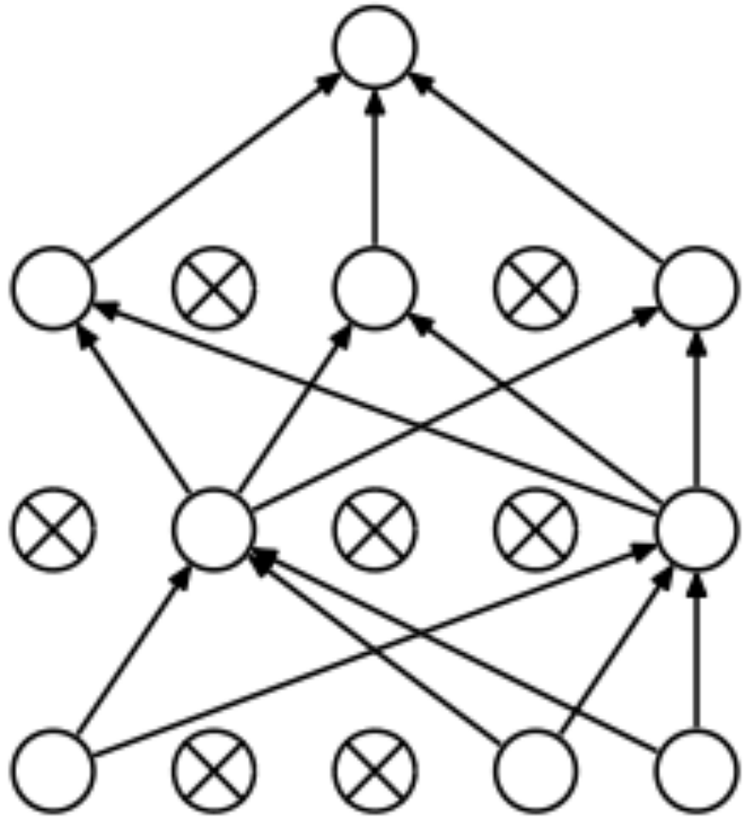
(b) After applying dropout.

Deep Learning

- Dropout
  - Em (Srivastava et al., 2014) diz: o treinamento deve extrair características mais generalistas pois não pode contar com todos os elementos totalmente conectados



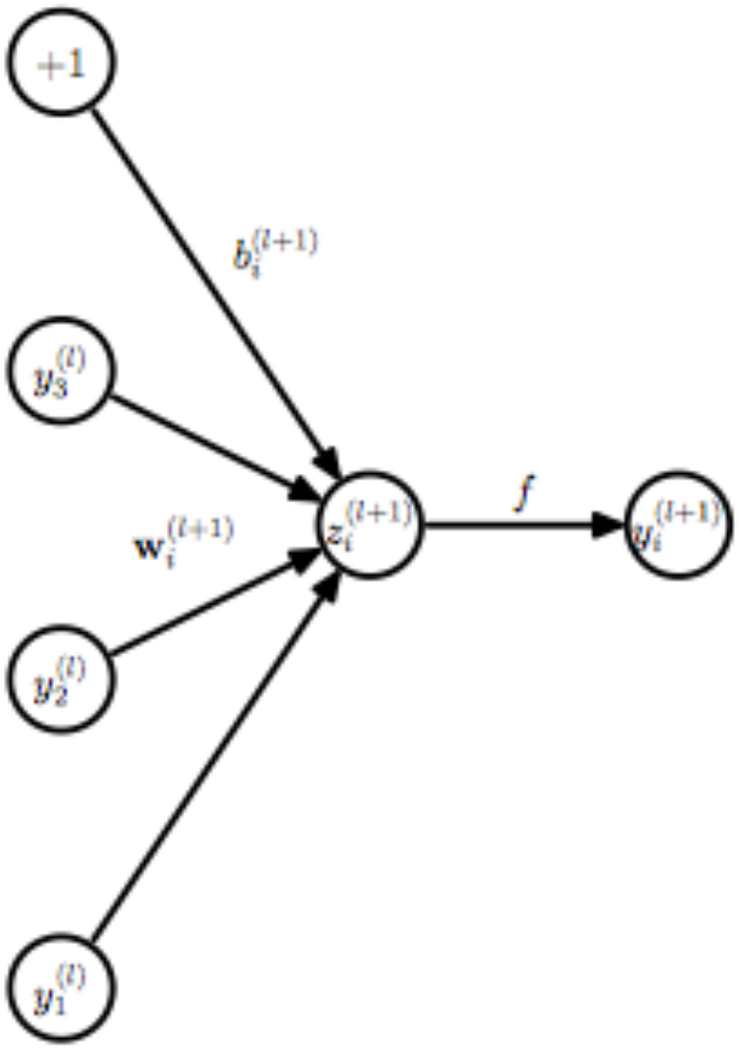
(a) Standard Neural Net



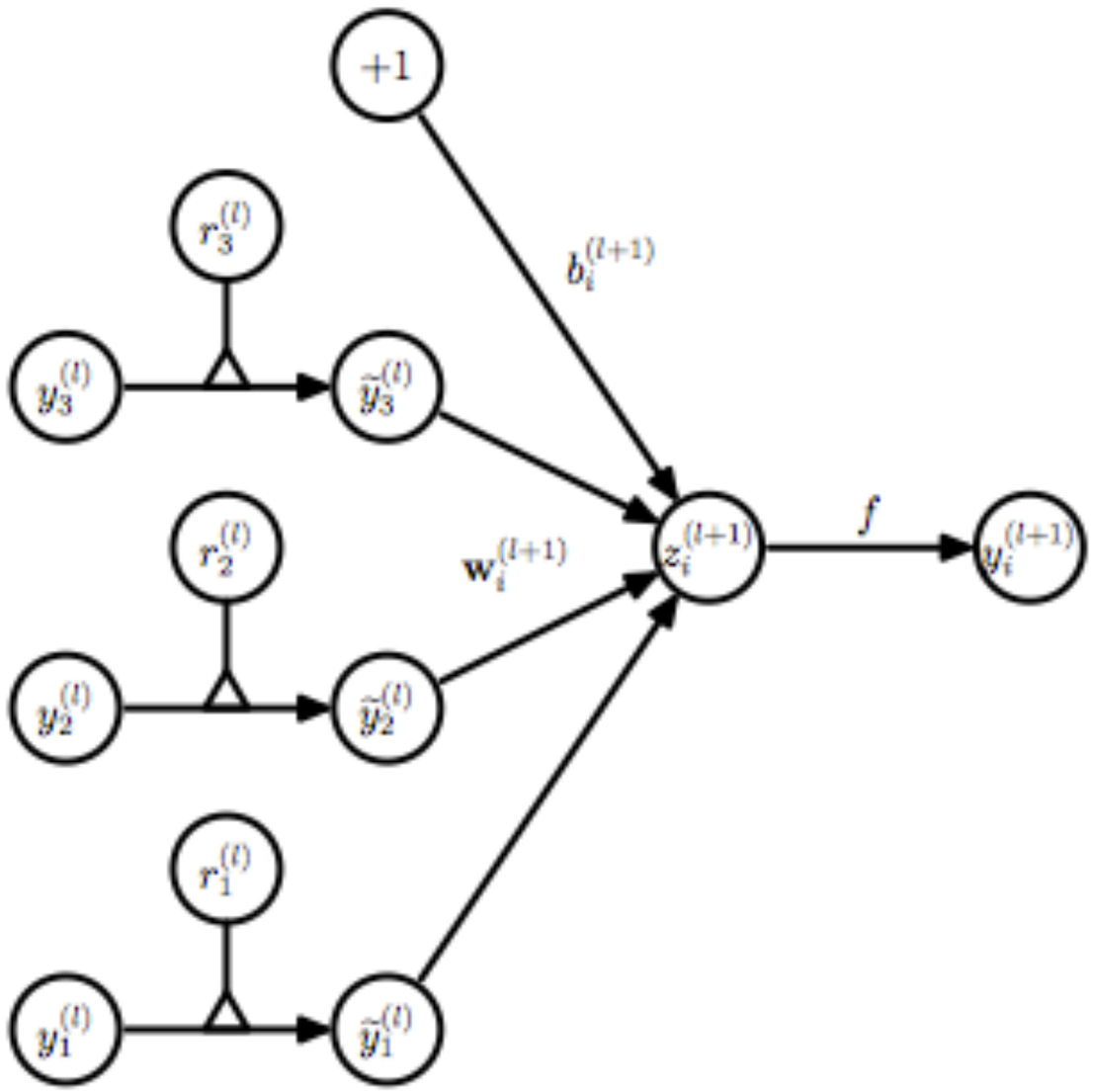
(b) After applying dropout.

Deep Learning

- Dropout



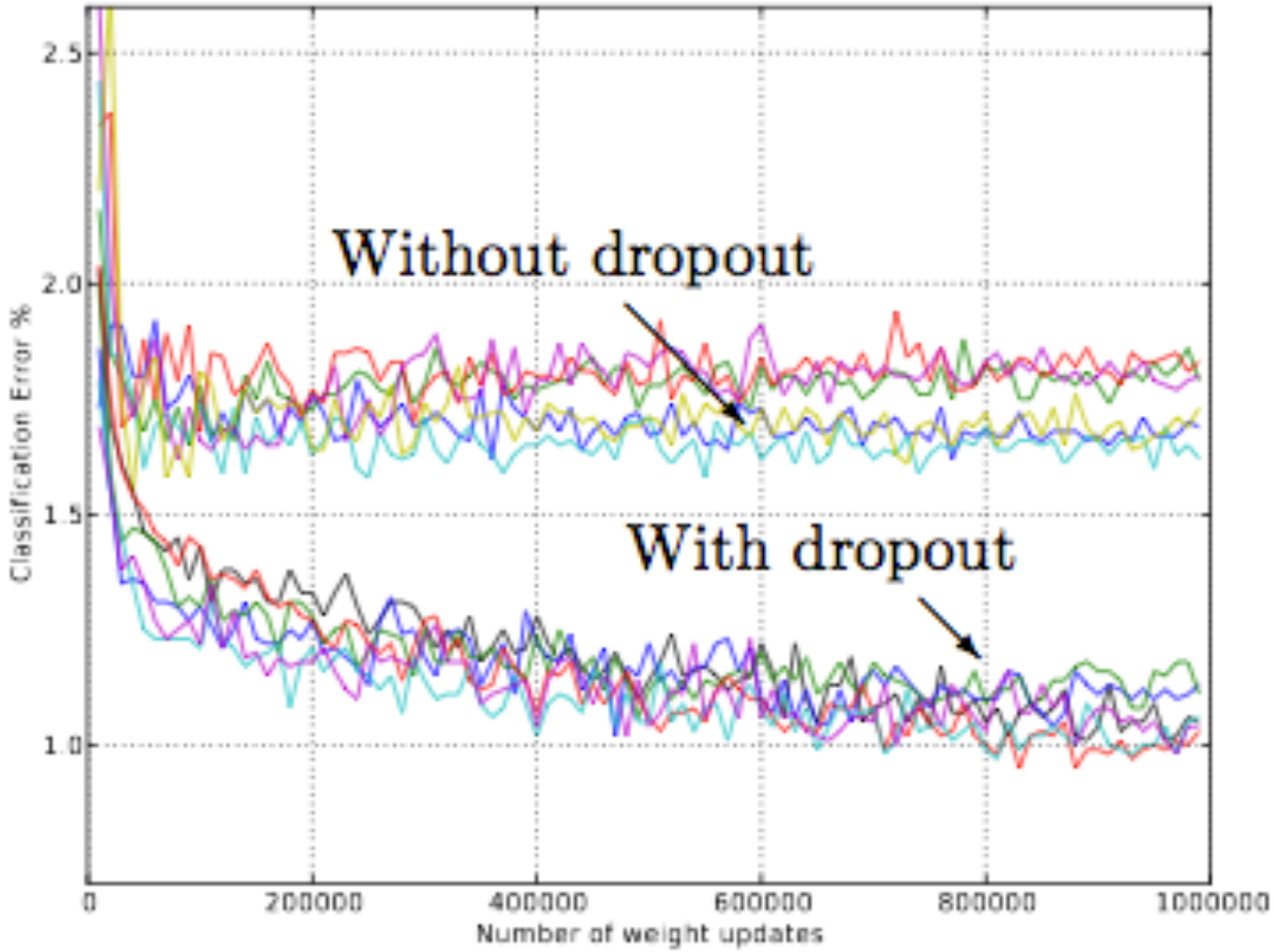
(a) Standard network



(b) Dropout network

Deep Learning

- Dropout - Srivastava et al., 2014



Deep Learning

- Dropout

Method	**Error %**
Binary Features (WDCH) (Netzer et al., 2011)	36.7
HOG (Netzer et al., 2011)	15.0
Stacked Sparse Autoencoders (Netzer et al., 2011)	10.3
KMeans (Netzer et al., 2011)	9.4
Multi-stage Conv Net with average pooling (Sermanet et al., 2012)	9.06
Multi-stage Conv Net + L2 pooling (Sermanet et al., 2012)	5.36
Multi-stage Conv Net + L4 pooling + padding (Sermanet et al., 2012)	4.90
Conv Net + max-pooling	3.95
Conv Net + max pooling + dropout in fully connected layers	3.02
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	2.80
Conv Net + max pooling + dropout in all layers	2.55
Conv Net + maxout (Goodfellow et al., 2013)	**2.47**
Human Performance	2.0

Method	**CIFAR-10**	**CIFAR-100**
Conv Net + max pooling (hand tuned)	15.60	43.48
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	15.13	42.51
Conv Net + max pooling (Snoek et al., 2012)	14.98	-
Conv Net + max pooling + dropout fully connected layers	14.32	41.26
Conv Net + max pooling + dropout in all layers	12.61	**37.20**
Conv Net + maxout (Goodfellow et al., 2013)	**11.68**	38.57

Deep Learning

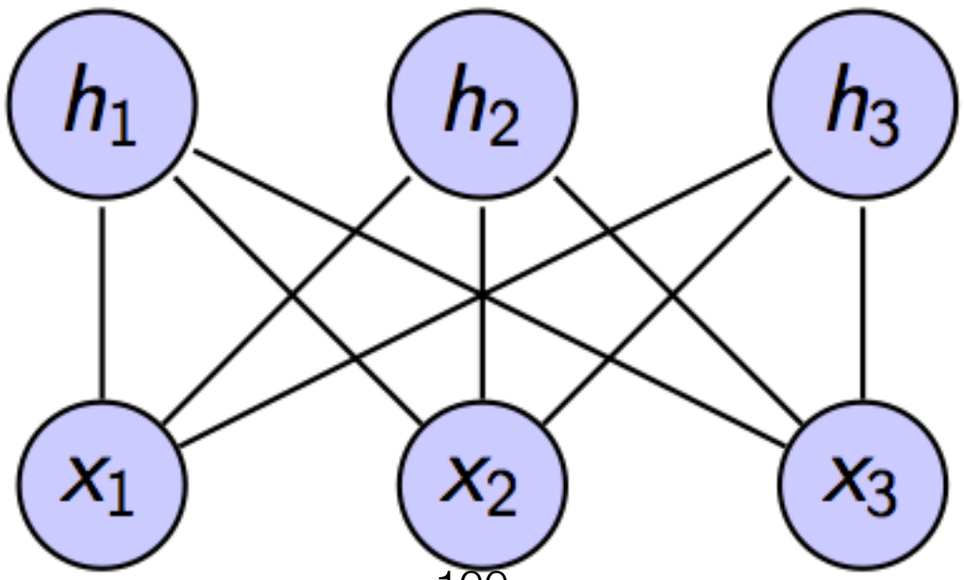
- Abordagem do Deep Learning
  - Encontrar $f : x \rightarrow y$
  - Mas ao invés de fazer isso diretamente, por que não quebrar o problema em partes?

$$x \rightarrow h \rightarrow y$$

- Como encontrar h?
  - Diferentes métodos de Deep Learning, diferentes métodos de encontrar h!!!
  - Deep Belief Net usam Restricted Boltzmann Machines (RBMs)

Deep Belief Networks

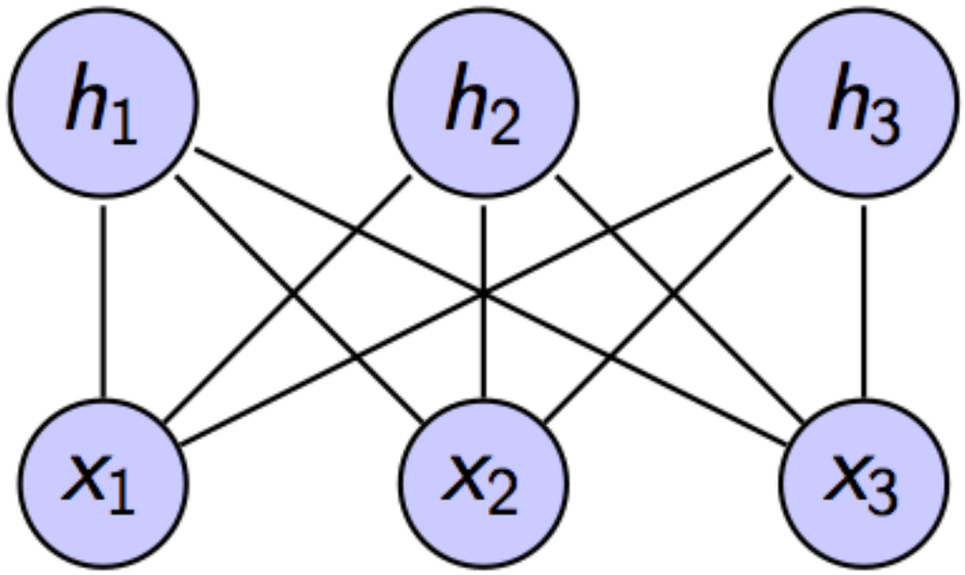
- Restricted Boltzmann Machine (RBM)
  - Modelo baseado na energia
  - $p(x,h) = \frac{1}{Z_\theta} e^{E_\theta(x,h)}$
  - onde: $E_\theta(x,h) = -x^T W h - b^T x - d^T h$
  - Assumindo que x_i e h_i são **BINÁRIOS**
  - $Z_\theta = \sum_{(x,h)} e^{-E_\theta(x,h)}$ é um fator de normalização



Deep Belief Networks

- Restricted Boltzmann Machine (RBM)

  - Exemplo:

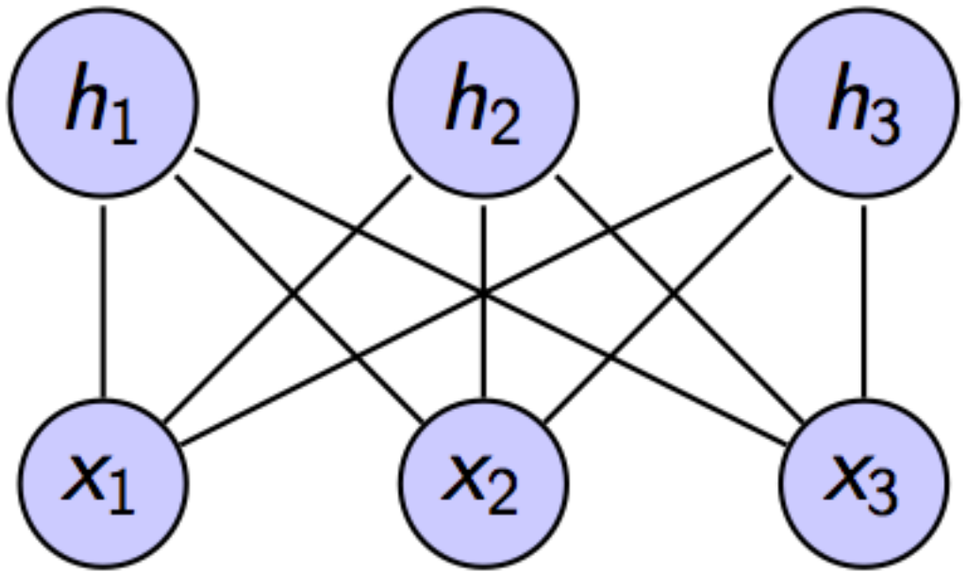


  - Suponha os pesos (h_1,x_1), (h_1,x_3) positivos, os outros iguais a zero e b = d = 0.

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)

  - Exemplo:



- Suponha os pesos (h_1,x_1), (h_1,x_3) positivos, os outros iguais a zero e b = d = 0.

- Então, esta RBM define a distribuição de [x_1,x_2,x_3,h_1,h_2,h_3] onde p(x_1=1,x_2=0,x_3=1,h_1=1,h_2=0,h_3=0) tem alta probabilidade

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)

1. Exemplo: Calculo de p(h|x)

$$p(h|x) = \frac{p(x,h)}{\sum_h p(x,h)} = \frac{e^{-E_\theta(x,h)}}{\sum_h e^{-E_\theta(x,h)}} = \frac{e^{(x^T W h + b^T x + d^T h)}}{\sum_h e^{(x^T W h + b^T x + d^T h)}}$$

$$p(h|x) = \frac{e^{(x^T W h + d^T h)} \cdot e^{b^T x}}{\sum_h \left[e^{(x^T W h + d^T h)} \cdot e^{b^T x} \right]} = \frac{\prod_j e^{(x^T W_j h_j + d_j^T h_j)}}{\sum_{h_1 \in [0,1]} \sum_{h_2 \in [0,1]} \cdots \sum_{h_j} \prod_j e^{(x^T W_j h_j + d_j^T h_j)}}$$

$$p(h|x) = \prod_j \frac{e^{(x^T W_j h_j + d_j^T h_j)}}{\sum_{h_j \in [0,1]} \left[e^{(x^T W_j h_j + d_j^T h_j)} \right]} = \prod_j p(h_j|x)$$

- Note que $p(h_j = 1|x) = \frac{e^{(x^T W_j + d_j)}}{Z_\theta} = \sigma(x^T W_j + d_j)$
- E que o cálculo de $p(x|h) = \prod_i p(x_i|h)$ é simples

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - Treinamento de uma RBM para otimizar P(x)

$$\partial_{w_{ij}} \log P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h \frac{1}{Z_w} e^{-E_w(x^m, h)}$$

$$= -\partial_{w_{ij}} \log Z_w + \partial_{w_{ij}} \log \sum_h e^{(-E_w(x^m, h))}$$

$$= -\frac{1}{\sum_h e^{(-E_w(x^m, h))}} \sum_h e^{(-E_w(x^m, h))} \partial_{w_{ij}} E_w(x^m, h) + \frac{1}{Z_w} \sum_{h,x} e^{(-E_w(x, h))} \partial_{w_{ij}} E_w(x, h)$$

$$= -\sum_h P_w(x^m, h) \left[\partial_{w_{ij}} E_w(x^m, h)\right] + \sum_{h,x} P_w(x, h) \left[\partial_{w_{ij}} E_w(x, h)\right]$$

$$= \mathbb{E}_{p(h|x=x^m)}[x_i^m \cdot h_j] - \mathbb{E}_{p(x,h)}[x_i \cdot h_j]$$

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - Treinamento de uma RBM para otimizar P(x)

$$\partial_{w_{ij}} \log P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h \frac{1}{Z_w} e^{-E_w(x^m, h)}$$

$$= -\partial_{w_{ij}} \log Z_w + \partial_{w_{ij}} \log \sum_h e^{(-E_w(x^m, h))}$$

$$= -\frac{1}{\sum_h e^{(-E_w(x^m,h))}} \sum_h e^{(-E_w(x^m,h))} \partial_{w_{ij}} E_w(x^m, h) + \frac{1}{Z_w} \sum_{h,x} e^{(-E_w(x,h))} \partial_{w_{ij}} E_w(x, h)$$

$$= -\sum_h P_w(x^m, h) \left[\partial_{w_{ij}} E_w(x^m, h)\right] + \sum_{h,x} P_w(x, h) \left[\partial_{w_{ij}} E_w(x, h)\right]$$

$$= \mathbb{E}_{p(h|x=x^m)}[x_i^m \cdot h_j] - \mathbb{E}_{p(x,h)}[x_i \cdot h_j]$$

Expectativa nos dados

Expectativa no modelo

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - Treinamento de uma RBM para otimizar P(x)

$$\partial_{w_{ij}} \log P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h \frac{1}{Z_w} e^{-E_w(x^m, h)}$$

$$= -\partial_{w_{ij}} \log Z_w + \partial_{w_{ij}} \log \sum_h e^{(-E_w(x^m, h))}$$

$$= -\frac{1}{\sum_h e^{(-E_w(x^m,h))}} \sum_h e^{(-E_w(x^m,h))} \partial_{w_{ij}} E_w(x^m, h) + \frac{1}{Z_w} \sum_{h,x} e^{(-E_w(x,h))} \partial_{w_{ij}} E_w(x, h)$$

$$= -\sum_h P_w(x^m, h) \left[\partial_{w_{ij}} E_w(x^m, h)\right] + \sum_{h,x} P_w(x, h) \left[\partial_{w_{ij}} E_w(x, h)\right]$$

$$= \mathbb{E}_{p(h|x=x^m)}[x_i^m \cdot h_j] - \mathbb{E}_{p(x,h)}[x_i \cdot h_j]$$

Expectativa nos dados

Expectativa no modelo

Direção de atualização do peso w_{ij}

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - Treinamento de uma RBM para otimizar P(x)

$$\partial_{w_{ij}} \log P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h P_w(x = x^m) = \partial_{w_{ij}} \log \sum_h \frac{1}{Z_w} e^{-E_w(x^m, h)}$$

$$= -\partial_{w_{ij}} \log Z_w + \partial_{w_{ij}} \log \sum_h e^{(-E_w(x^m, h))}$$

$$= -\frac{1}{\sum_h e^{(-E_w(x^m,h))}} \sum_h e^{(-E_w(x^m,h))} \partial_{w_{ij}} E_w(x^m, h) + \frac{1}{Z_w} \sum_{h,x} e^{(-E_w(x,h))} \partial_{w_{ij}} E_w(x, h)$$

$$= -\sum_h P_w(x^m, h) \left[\partial_{w_{ij}} E_w(x^m, h)\right] + \sum_{h,x} P_w(x, h) \left[\partial_{w_{ij}} E_w(x, h)\right]$$

$$= \mathbb{E}_{p(h|x=x^m)}[x_i^m \cdot h_j] - \mathbb{E}_{p(x,h)}[x_i \cdot h_j]$$

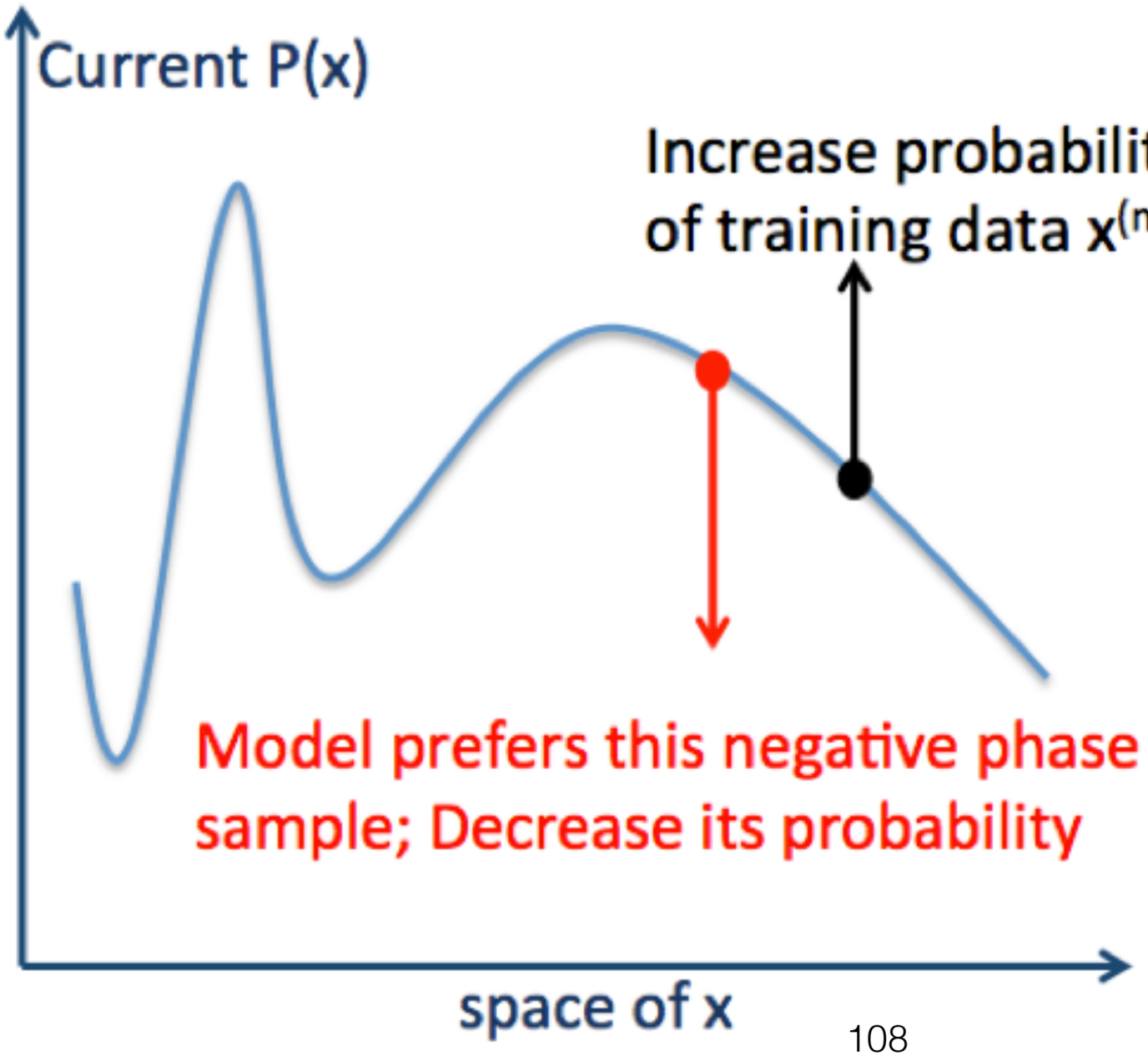
Termo que aumenta a probabilidade de x^m

Termo que reduz a probabilidade dos demais termos gerados pelo modelo

Direção de atualização do peso w_{ij}

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - Treinamento de uma RBM para otimizar P(x)



Análise detalhada feita em [Carreira-Perpinan and Hinton, 2005]

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)

  - Treinamento de uma RBM para otimizar P(x)

$$\mathbb{E}_{p(h|x=x^m)}[x_i^m \cdot h_j] - \mathbb{E}_{p(x,h)}[x_i \cdot h_j]$$

Alto custo computacional

- Solução: *Contrastive Divergence Algorithm*

- Algoritmo de cálculo rápido embora apresente uma tendência.

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - *Contrastive Divergence Algorithm*
  - Amostragem de Gibbs (amostras iterativas de x e h)
    - Algoritmo

1. Seja x^m o ponto de treinamento e $W = [w_{ij}]$ os pesos atuais
2. Amostra h'_j de $p(h_j|x=x^m)$
3. Amostra x'_i de $p(x_i|h=h_j)$
4. Amostra h''_j de $p(h_j|x=x'_i)$
5. Faz a Atualização dos pesos

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - *Contrastive Divergence Algorithm*
  - Amostragem de Gibbs (amostras iterativas de x e h)
    - Algoritmo

1. Seja x^m o ponto de treinamento e $W = [w_{ij}]$ os pesos atuais
2. Amostra h'_j de $p(h_j|x=x^m)$ $h'_j \leftarrow p(h_j|x = x^m) = \sigma\left(\sum_i w_{ij} x_i^m + d_j\right)$
3. Amostra x'_i de $p(x_i|h=h_j)$
4. Amostra h''_j de $p(h_j|x=x'_i)$
5. Faz a Atualização dos pesos

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - *Contrastive Divergence Algorithm*
  - Amostragem de Gibbs (amostras iterativas de x e h)
    - Algoritmo

1. Seja x^m o ponto de treinamento e $W = [w_{ij}]$ os pesos atuais
2. Amostra h'_j de $p(h_j|x=x^m)$
3. Amostra x'_i de $p(x_i|h=h_j)$ $\quad x'_i \leftarrow p(x_i|h=h') = \sigma\left(\sum_j w_{ij} h'_j + b_i\right)$
4. Amostra h''_j de $p(h_j|x=x'_i)$
5. Faz a Atualização dos pesos

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - *Contrastive Divergence Algorithm*
  - Amostragem de Gibbs (amostras iterativas de x e h)
    - Algoritmo

1. Seja x^m o ponto de treinamento e $W = [w_{ij}]$ os pesos atuais
2. Amostra h'_j de $p(h_j|x=x^m)$
3. Amostra x'_i de $p(x_i|h=h_j)$
4. Amostra h''_j de $p(h_j|x=x'_i)$ $h''_j \leftarrow p(h_j|x = x^m) = \sigma\left(\sum_i w_{ij}x'_i + d_j\right)$
5. Faz a Atualização dos pesos

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - *Contrastive Divergence Algorithm*
  - Amostragem de Gibbs (amostras iterativas de x e h)
    - Algoritmo

1. Seja x^m o ponto de treinamento e $W = [w_{ij}]$ os pesos atuais
2. Amostra h'_j de $p(h_j|x=x^m)$
3. Amostra x'_i de $p(x_i|h=h_j)$
4. Amostra h''_j de $p(h_j|x=x'_i)$
5. Faz a Atualização dos pesos

$$w_{ij} \leftarrow w_{ij} + \gamma(x_i^m \cdot h'_j - x'_i \cdot h''_j)$$

Deep Belief Networks

- Restricted Boltzmann Machine (RBM)
  - *Contrastive Divergence Algorithm*
  - Amostragem de Gibbs (amostras iterativas de x e h)
    - Algoritmo

1. Seja x^m o ponto de treinamento e $W = [w_{ij}]$ os pesos atuais
2. Amostra h'_j de $p(h_j|x=x^m)$
3. Amostra x'_i de $p(x_i|h=h_j)$
4. Amostra h''_j de $p(h_j|x=x'_i)$
5. Faz a Atualização dos pesos

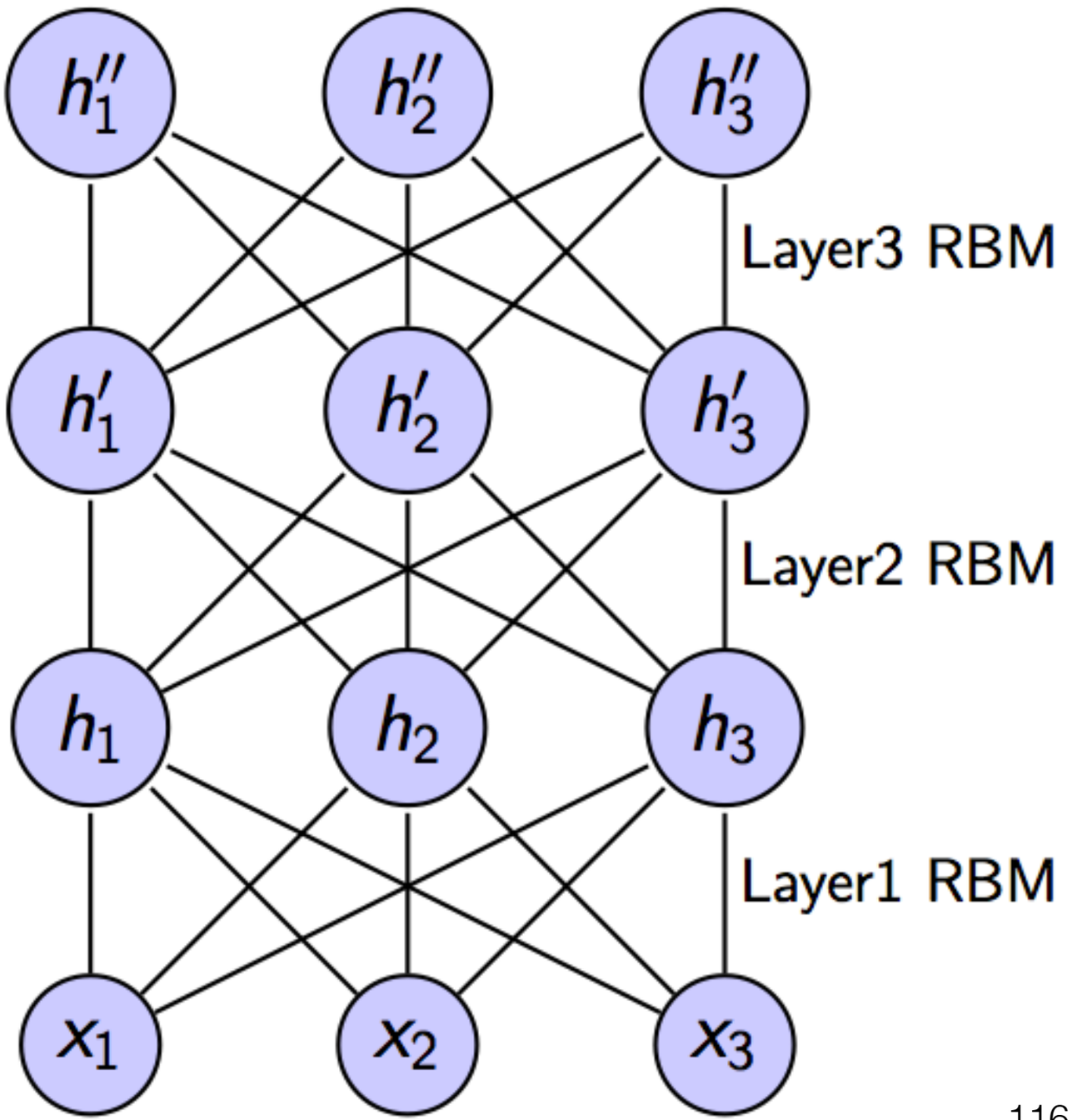
Atualização correta

Erro do modelo atual

$$w_{ij} \leftarrow w_{ij} + \gamma(x_i^m \cdot h'_j - x'_i \cdot h''_j)$$

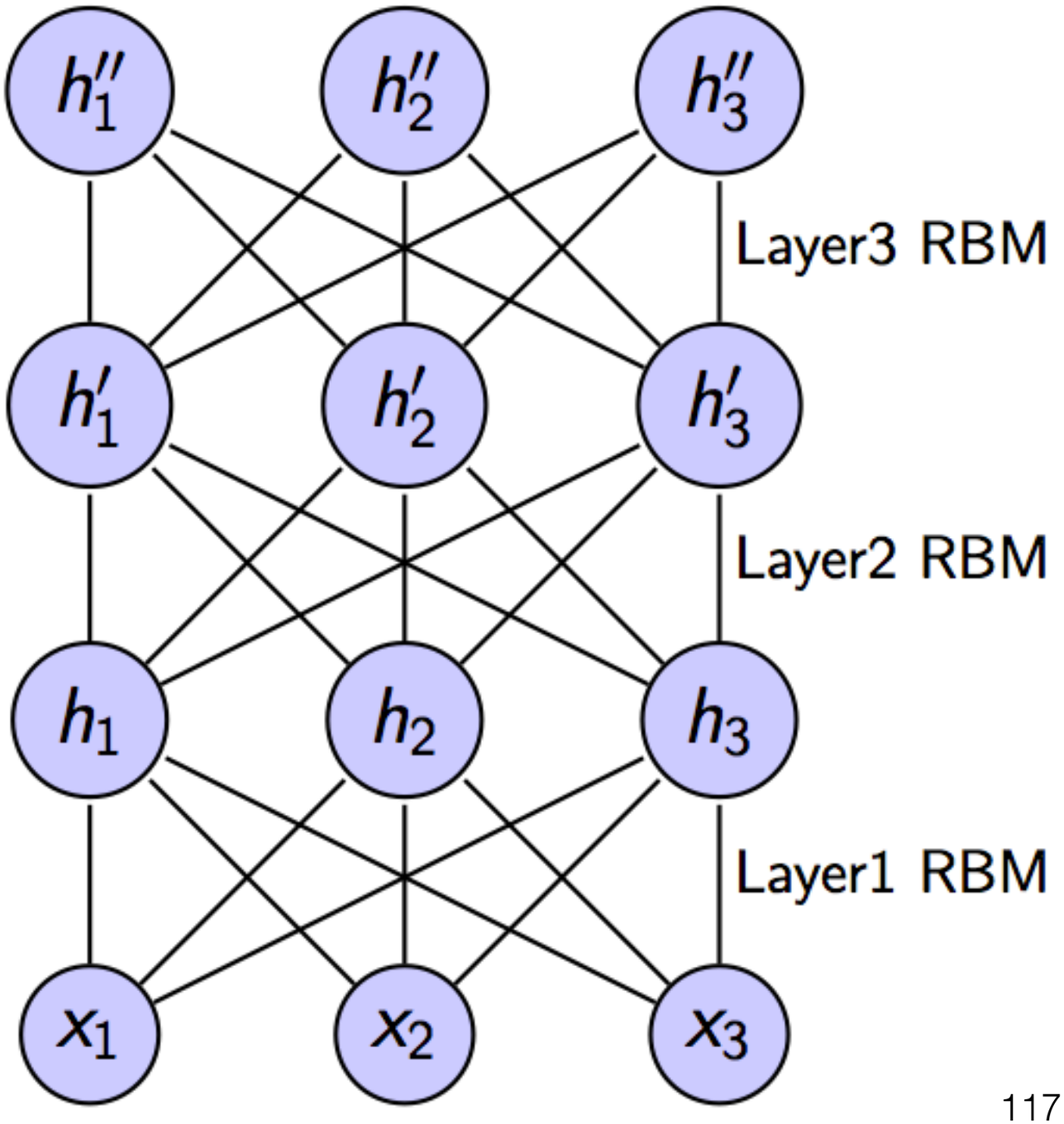
Deep Belief Networks

- Uma vez que as RBMs foram treinadas, teremos a Deep Belief Net com o cascateamento de várias RBMs



Deep Belief Networks

- Uma vez que as RBMs foram treinadas, teremos a Deep Belief Net com o cascateamento de várias RBMs

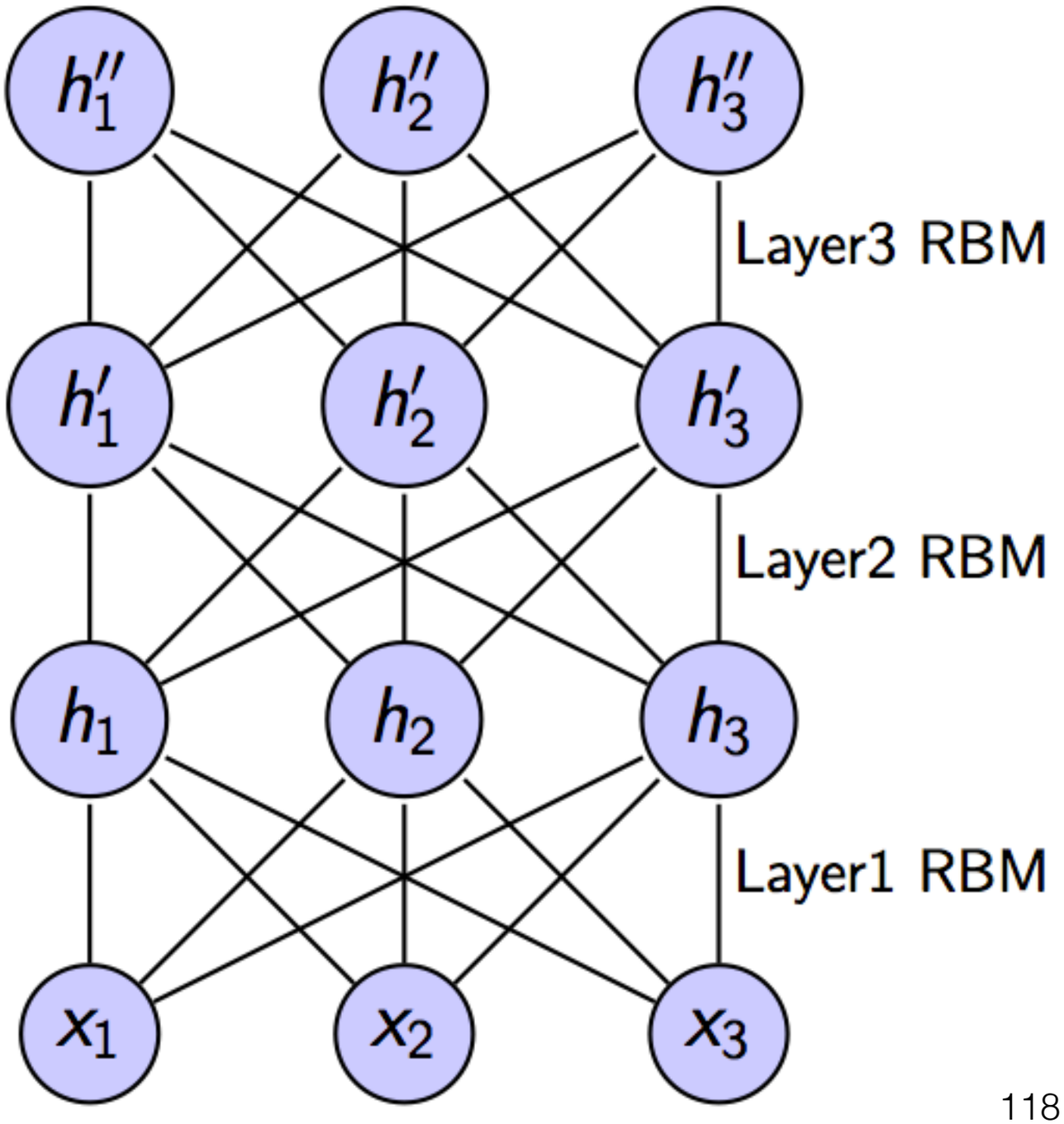


DBN define um modelo gerador p(x) com formato:

$$p(x) = \sum_{h,h',h''} p(x|h)p(x|h')p(h',h'')$$

Deep Belief Networks

- Uma vez que as RBMs foram treinadas, teremos a Deep Belief Net com o cascateamento de várias RBMs

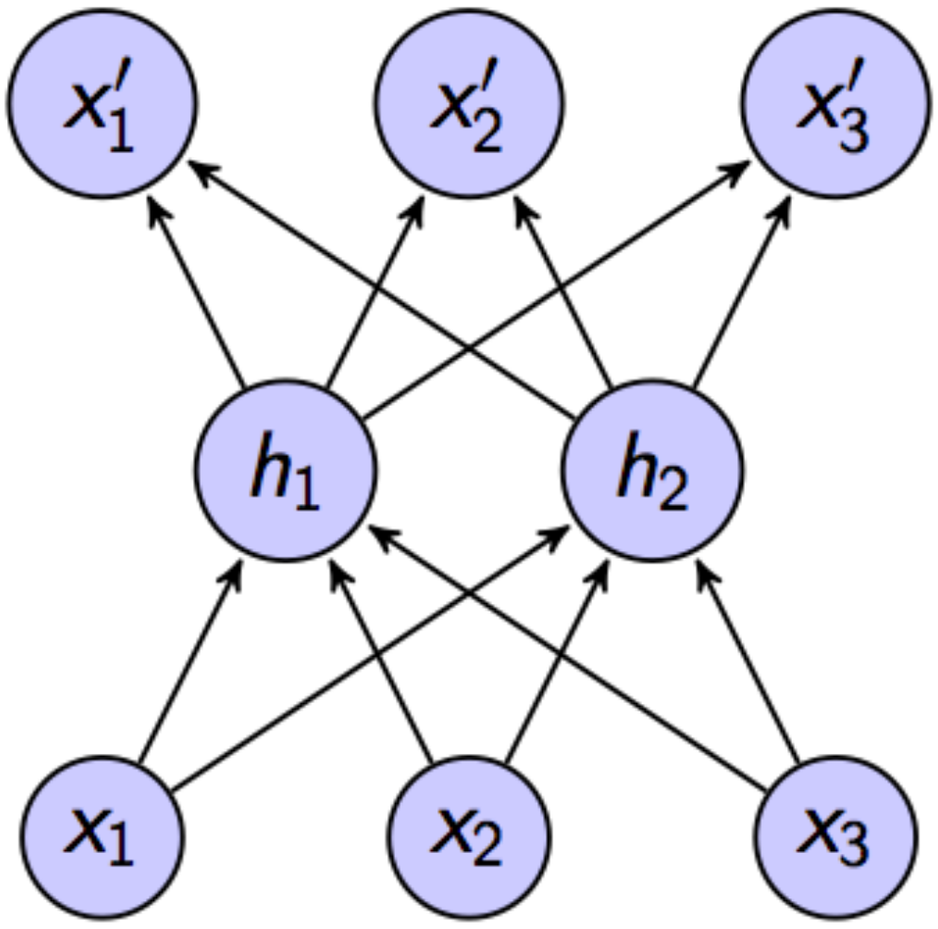


DBN define um modelo gerador p(x) com formato:

$$p(x) = \sum_{h,h',h''} p(x|h)p(x|h')p(h',h'')$$

Uma outra aplicação é a inicialização de Deep NN.

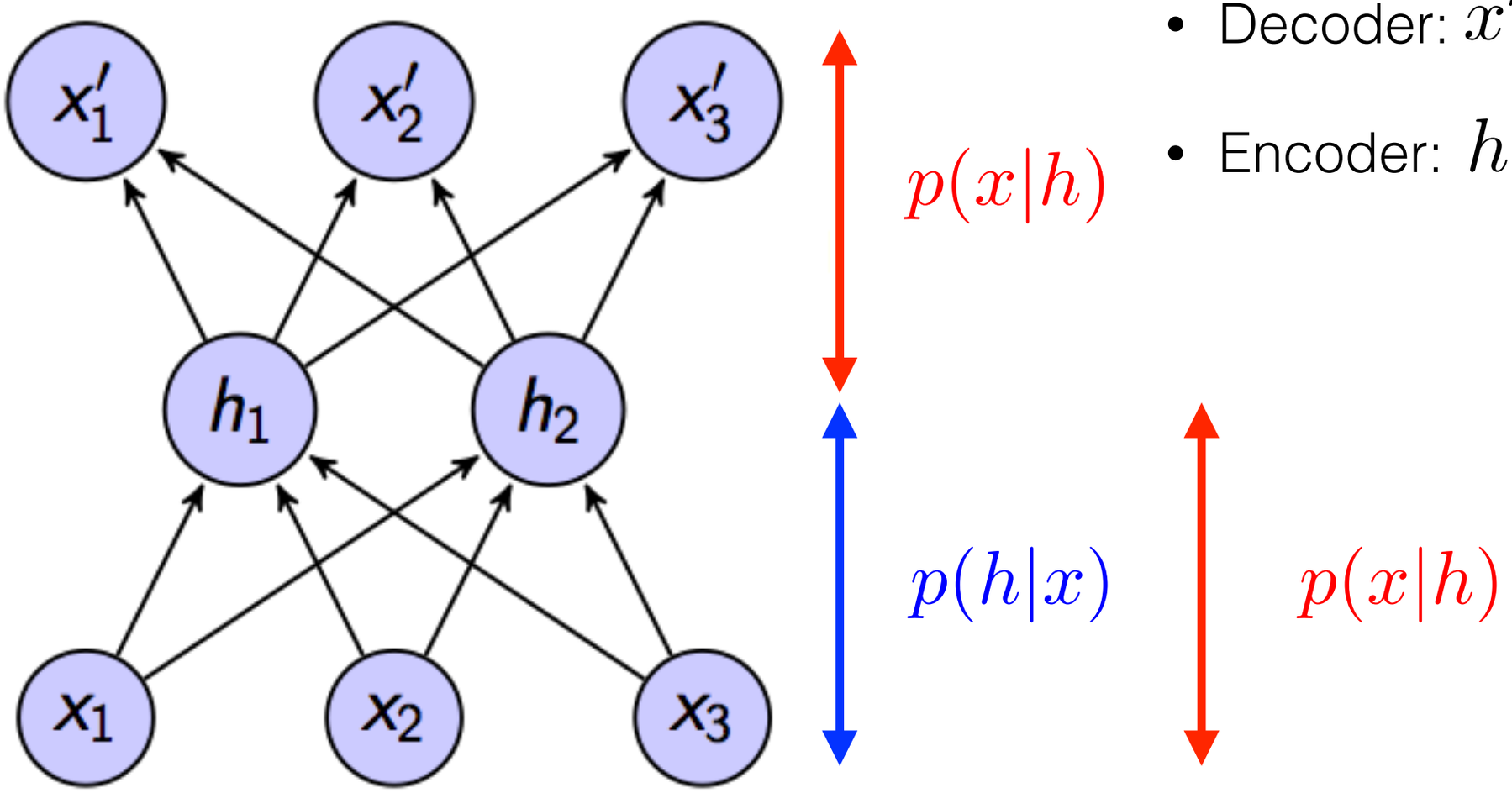
Auto-Encoders



- Decoder: $x' = \sigma(W'h + d)$
- Encoder: $h = \sigma(Wx + b)$

- Definições:
  - $Custo = \sum_m ||x^m - DECODER(ENCODER(x^m))||^2$
  - Reconstrução: $x' = \sigma(W'\sigma(Wx + b) + d)$

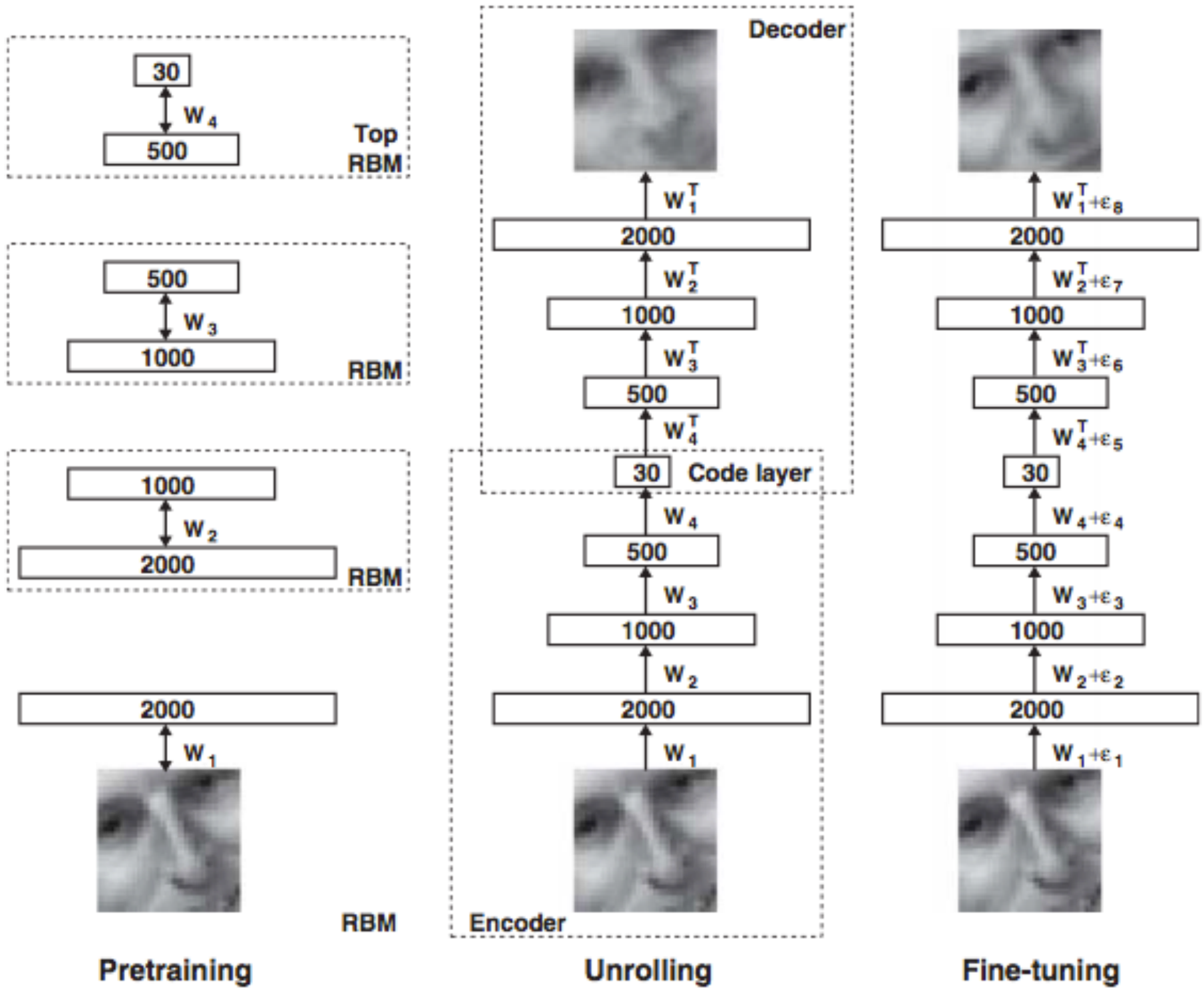
Auto-Encoders



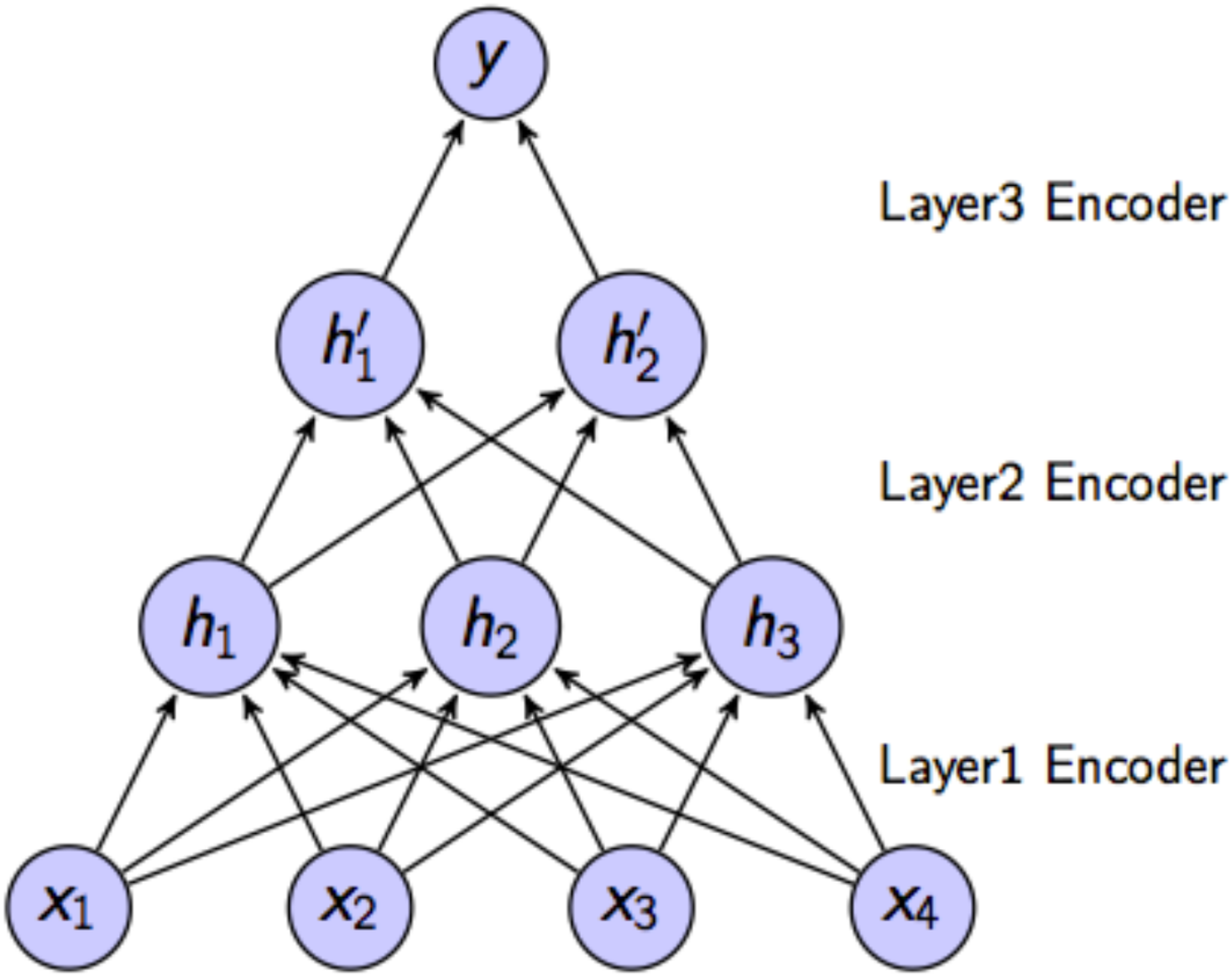
- Decoder: $x' = \sigma(W'h + d)$
- Encoder: $h = \sigma(Wx + b)$

- Então a resolução ficaria bem próxima ao que foi feito para as RBMs

Auto-Encoders

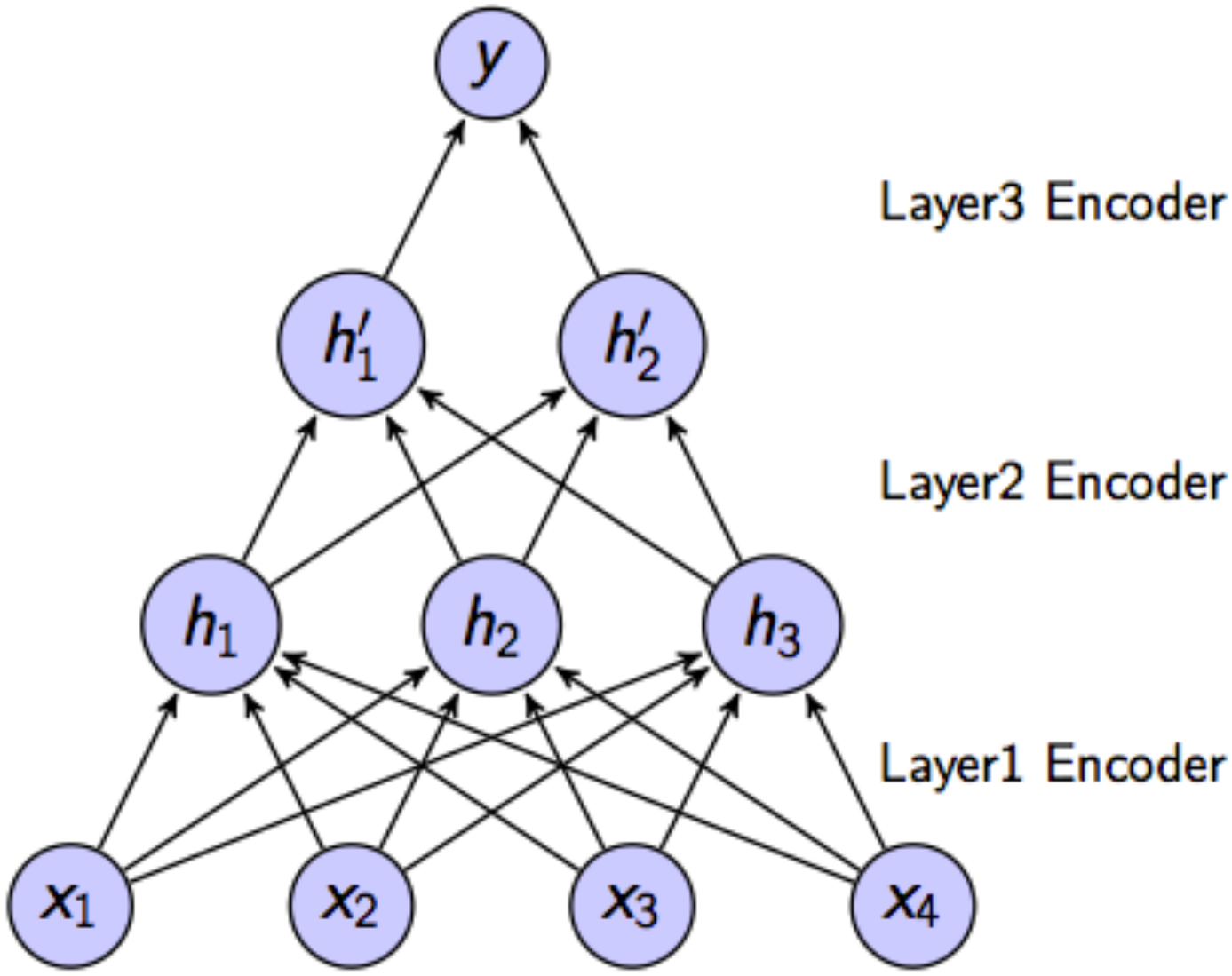


Auto-Encoders



- Diferentemente das RBMs, Auto-Encoders são determinísticos
- $h = \sigma(Wx + b)$ e não $p(h = \{0, 1\}) = \sigma(Wx + b)$

Auto-Encoders

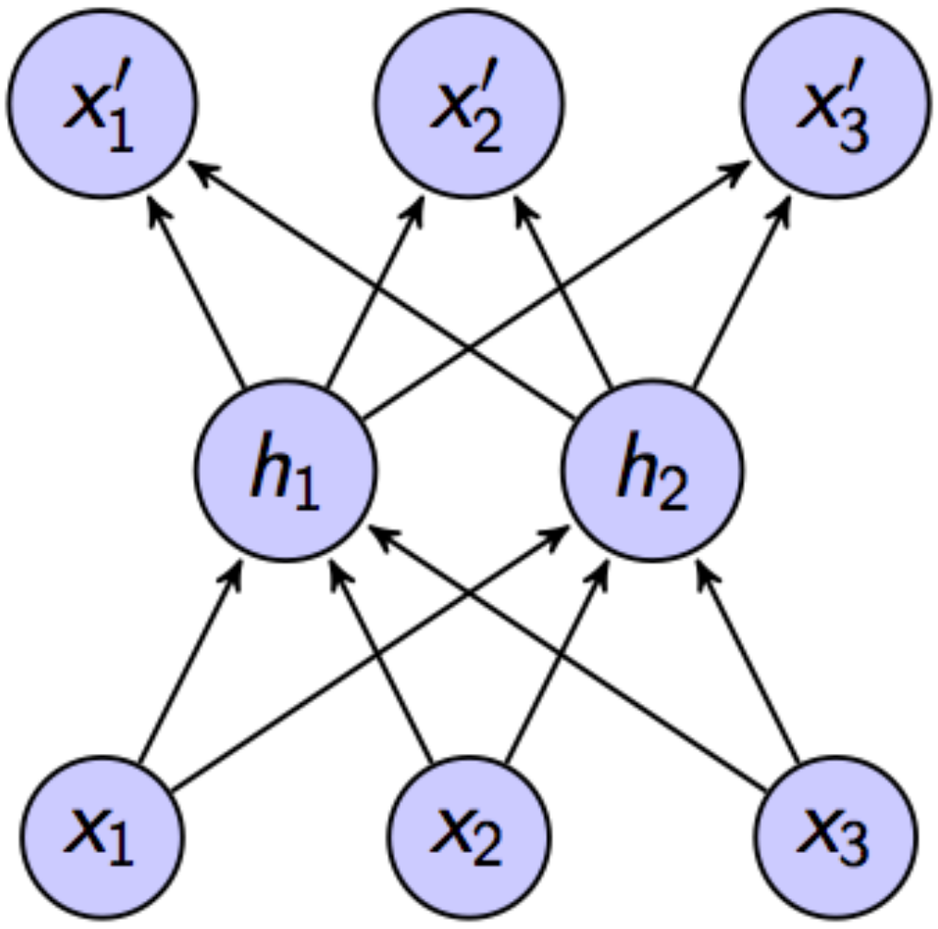


- + Rápido para treinar quando comparado com RBM

Auto-Encoders

- Variações de aplicação para os Auto-Encoders
  - Compressão de dados - h possui dimensão menor do que x
  - Extração das Componentes Principais (Bourlard e Kamp, 1988) fazendo sigma uma função linear
  - Extração de Componentes Esparsos e/ou outras representações *over-complete* (Ranzato et al., 2006) - h possui dimensão maior do x
  - Forçando h a possuir valores binários, a resposta do auto-encoder é uma função geradora de códigos *hash* (Salakhutdinov e Hinton, 2007)
  - Durante o processo de treinamento do auto-encoder pela adição de ruído, pode-se explorar o seu conhecimento a respeito do ruído gerando assim um *denoising auto-encoder (*Vincent et al., 2010)

Auto-Encoders



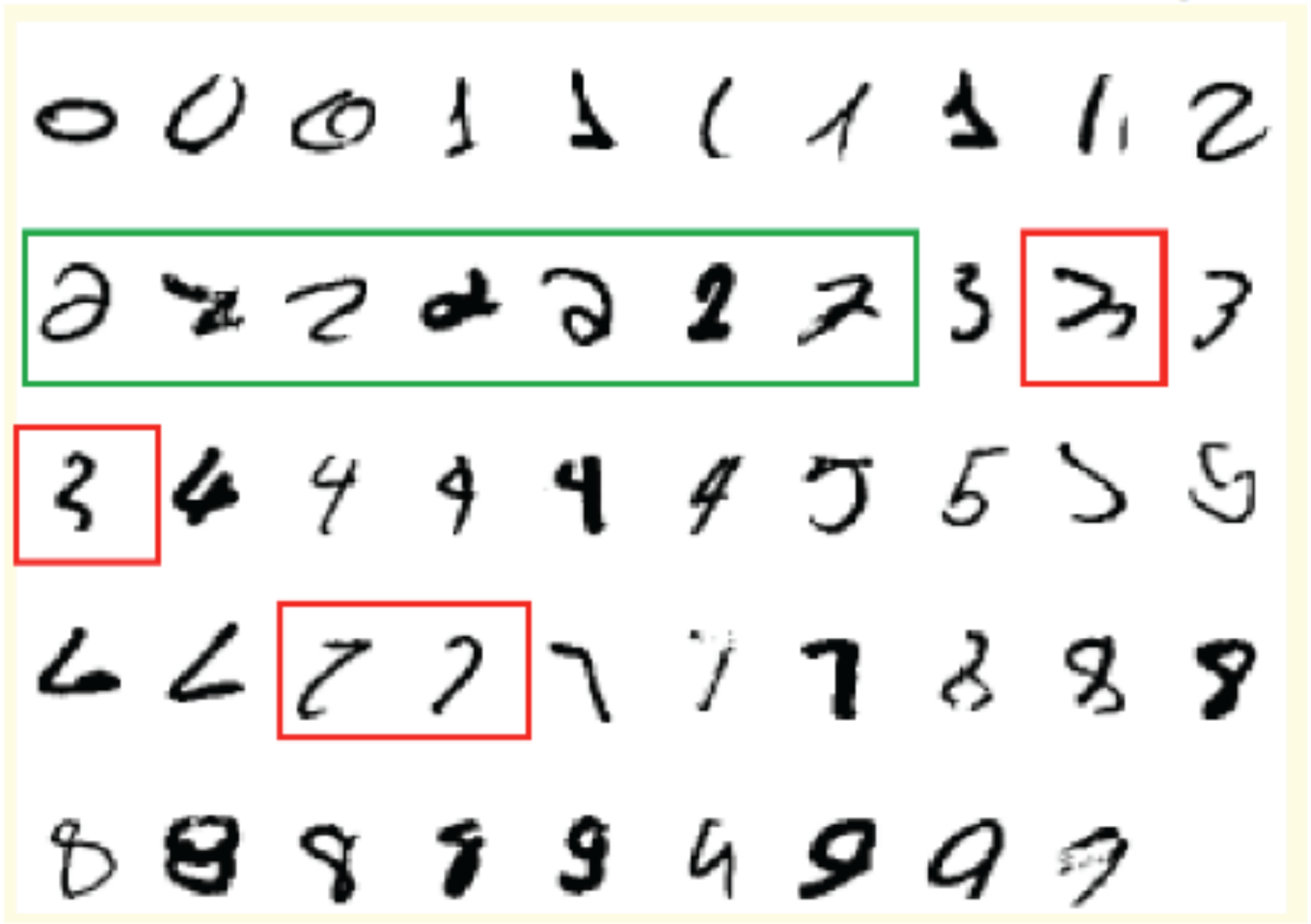
- Decoder: $x' = \sigma(W'h + d)$
- Encoder: $h = \sigma(W\hat{x} + b)$

$$\hat{x} = x + noise$$

- Definições:
  - $Custo = \sum_m ||x^m - DECODER(ENCODER(\hat{x^m}))||^2$
  - Reconstrução: $x' = \sigma(W'\sigma(Wx + b) + d)$

Auto-Encoders

- Exemplo: Rotação, escala e deslocamento aleatórios em imagens de caracteres e a adição de ruídos guassianos e "sal com pimenta" (*salt-and-pepper noise*)



- Um "2" deve ser classificado como um "2" independente do ruído adicionado a ele e qualquer outra mudança feita a imagem.

Auto-Encoders

- Auto-encoders são uma alternativa mais simples (na questão do treinamento) do que RBMs e DBNs.

  - Como o treinamento não entra no viés probabilístico, podemos implementá-lo com *backpropagation* e seus derivados

- Auto-encoders aprendem a “comprimir” e a “reconstruir” o conjunto de dados de entrada, ou seja, “cria” uma nova representação, focando primeiro na representação, ou seja, em p(x)

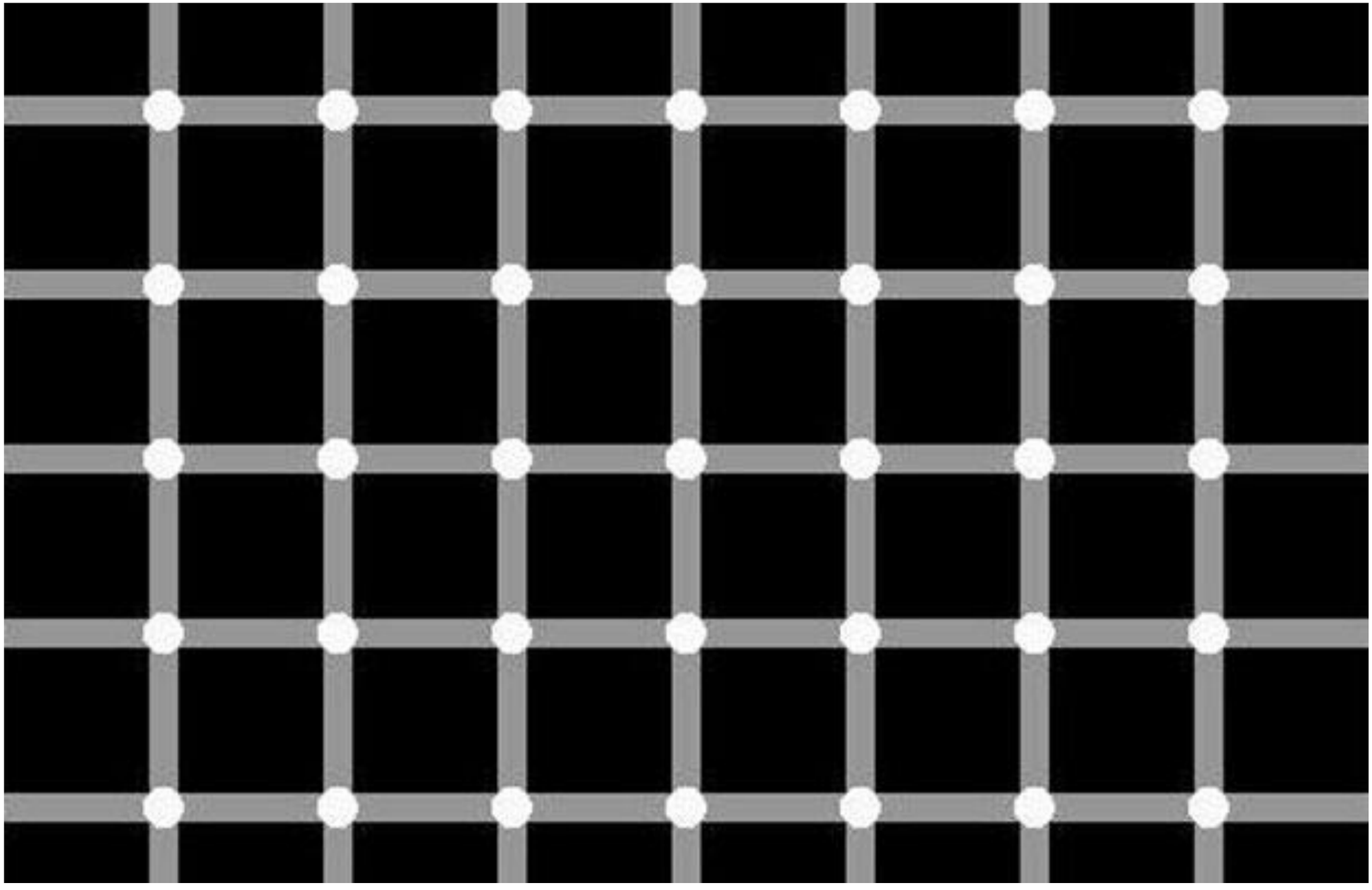
- Possuem diversas variações, algumas com a possibilidade de incorporar informações periféricas a representação extraída.

Redes Neurais Convolucionais

- Inspiradas no comportamento do cortex visual
- Redes do tipo feedforward
- Visa a utilização **mínima** de pré-processamento
- Extração de representações dos dados que acessem informações invariantes a rotação, translação e outras distorções da imagem.
- Como fazer isso? Distorcendo a imagem!

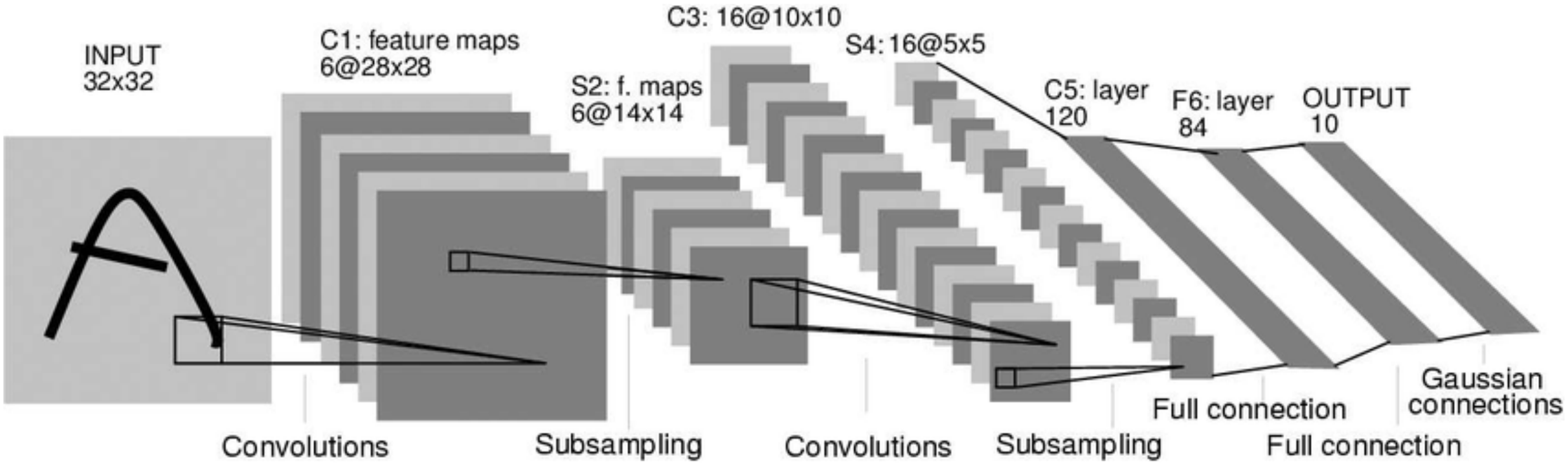
Redes Neurais Convolucionais

- Inspiradas no comportamento do cortex visual



Redes Neurais Convolucionais

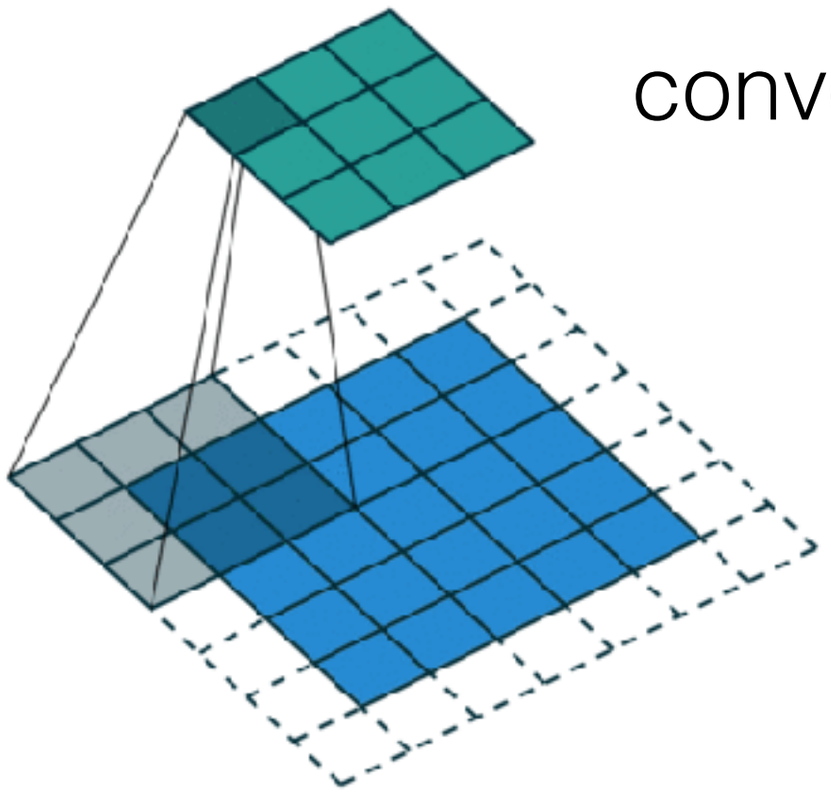
- Redes do tipo feedforward (LeNet)



- Neste caso, 32x32 entradas = 1024 entradas
- Supondo um modelo totalmente conectado com 10 neurônios: 10250 parâmetros
- É viável?

Redes Neurais Convolucionais

- Necessidade de redução extrema de parâmetros de treinamento



convolução

Aqui, 5x5 (25 entradas) acessadas por diferentes filtros 3x3 (9 saídas)

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

max pooling

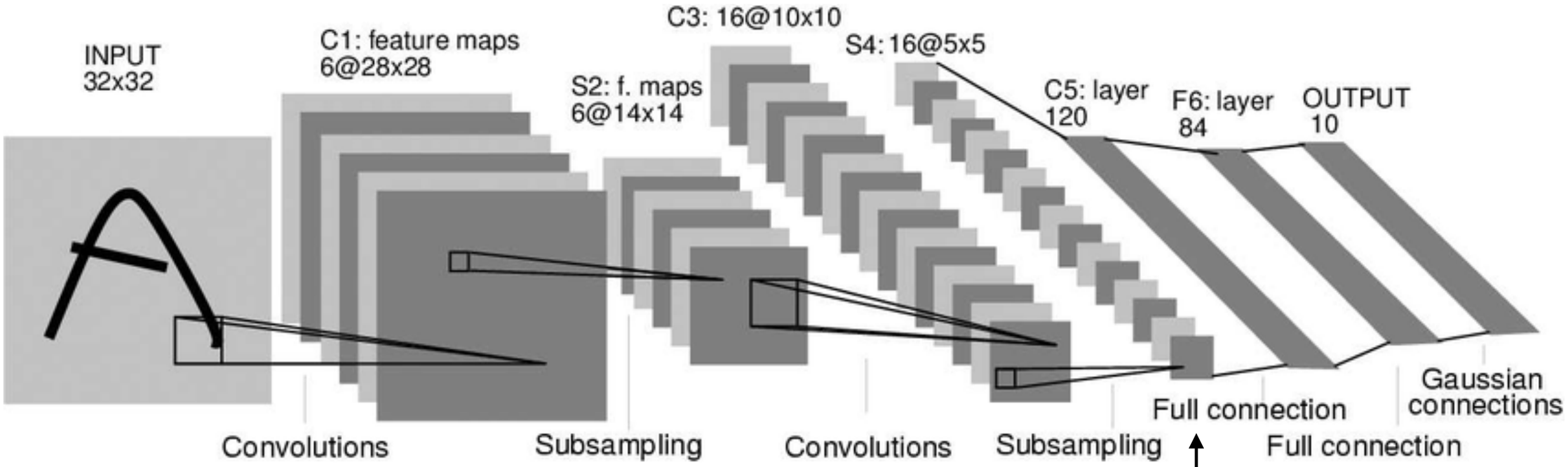
1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

mean pooling

Redes Neurais Convolucionais

- Necessidade de redução extrema de parâmetros de treinamento

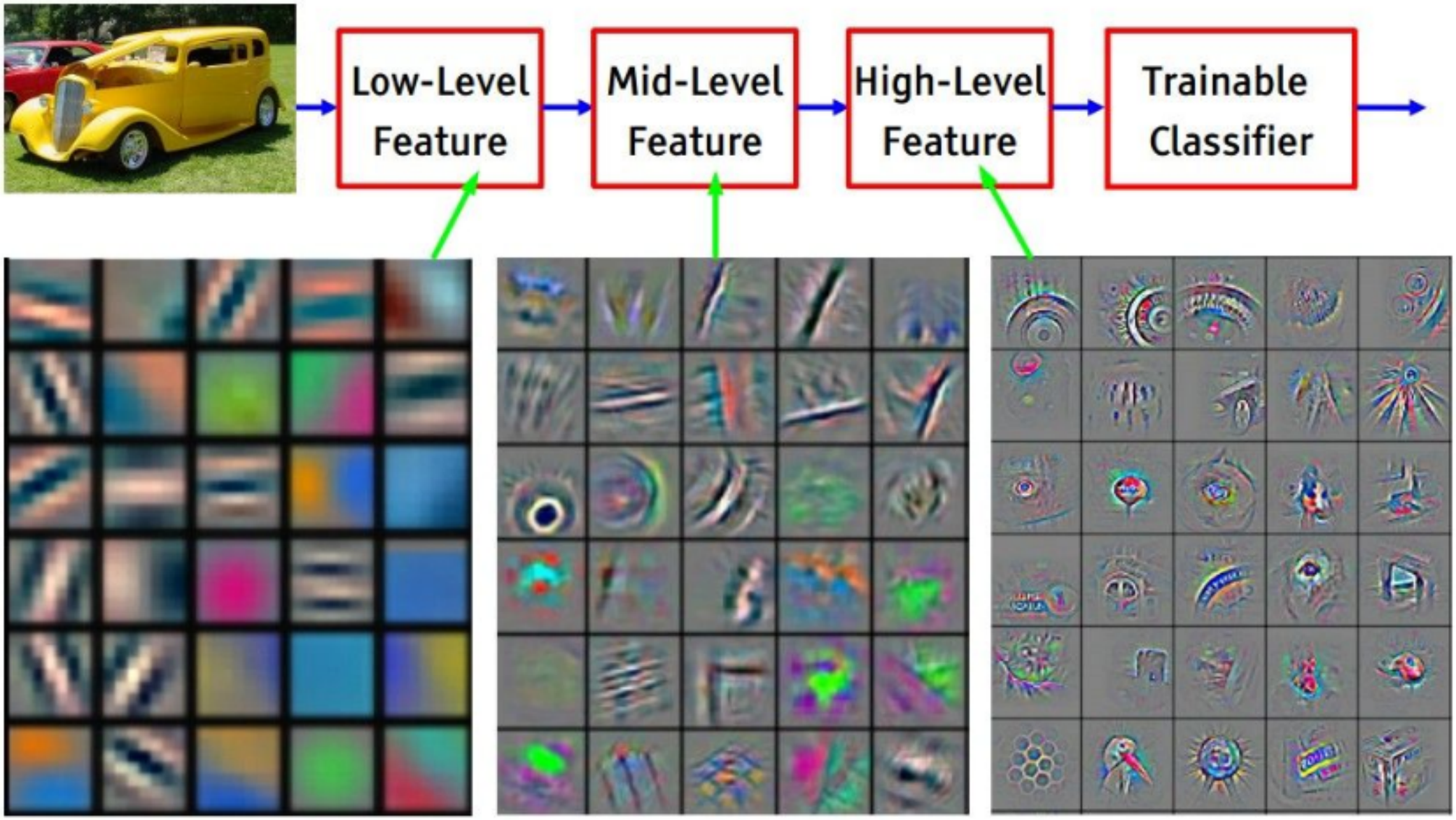


- Taxa de Redução em 0,40 (400/1024)
- Invariância a rotação e translação

Tomada de decisão em 400 dimensões

Redes Neurais Convolucionais

- Necessidade de redução extrema de parâmetros de treinamento



- ImageNet (Zeiler & Fergus, 2013) depois do treinamento

Redes Neurais Convolucionais

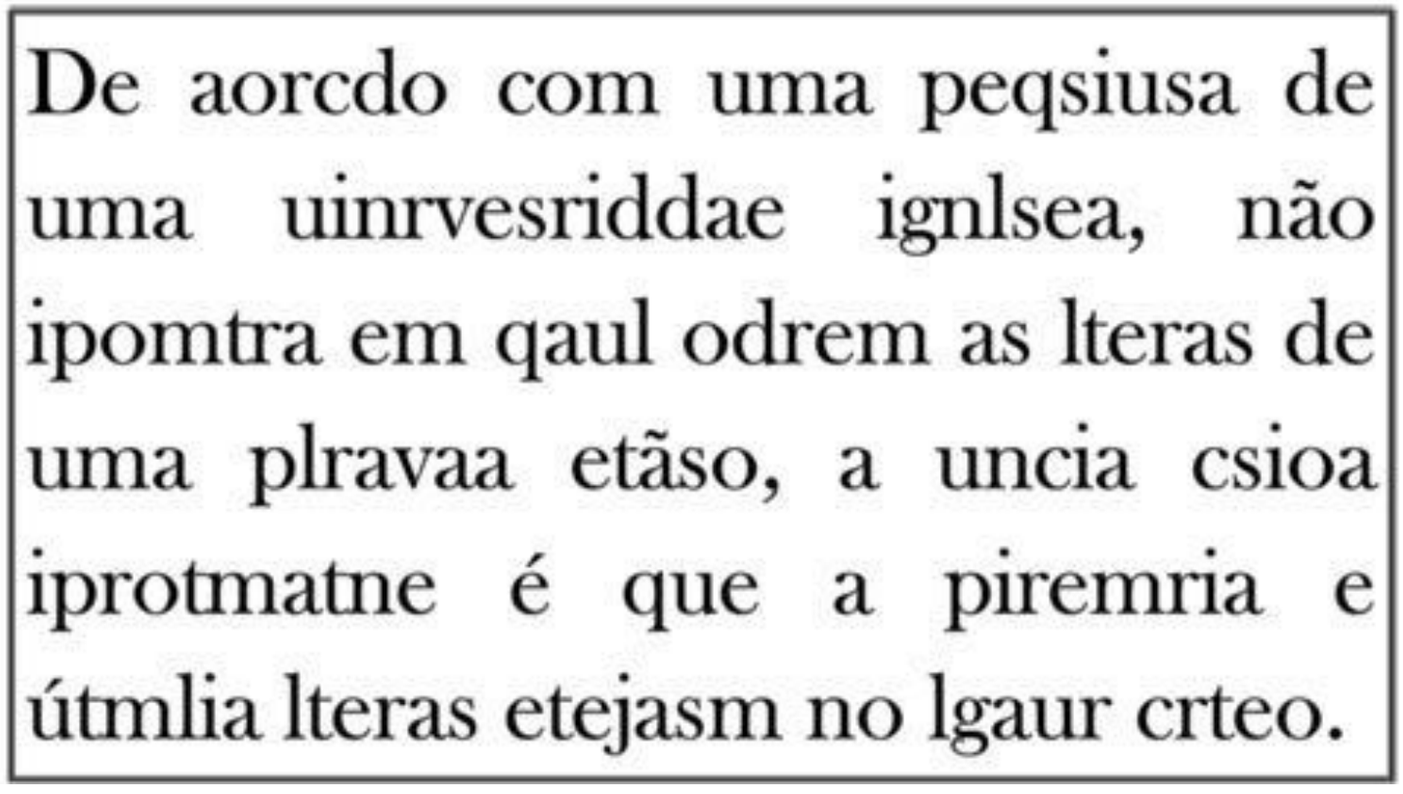
- Como treinar esse “monstro”? Backpropagation “normal”
  - Camada de classificação - BP
  - Camada de *pooling* - BP (em caso de ausência de sobreposição)
  - Camada de Convolução - BP
- Necessidade extrema de técnicas para evitar o overfit - muitos parâmetros!

Redes Neurais Recursivas

- Até aqui: uma entrada é totalmente independente das outras.
- Saída dependente somente da entrada correspondente...

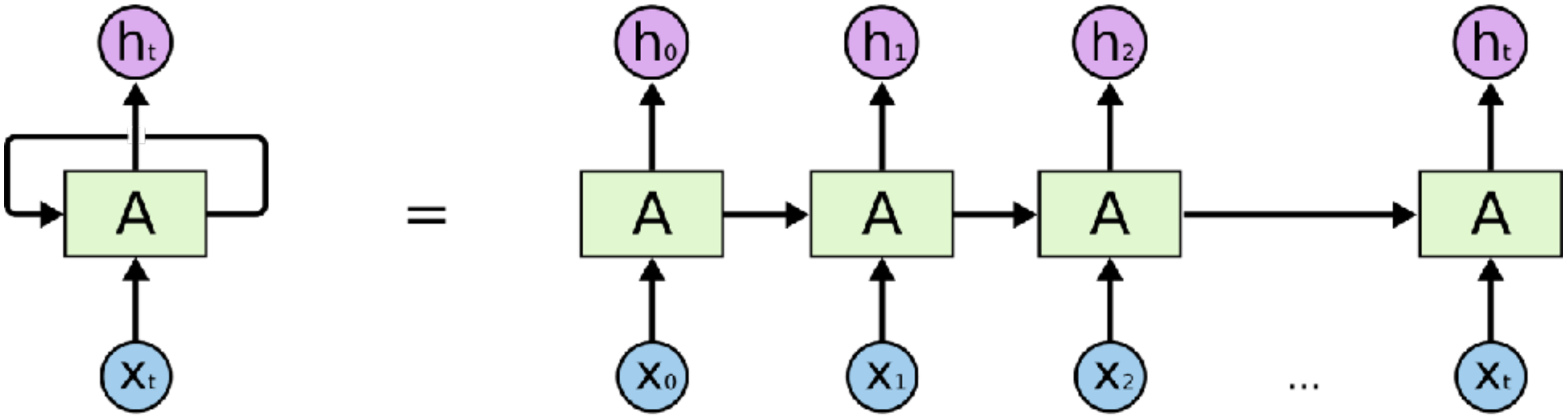
$$f : x \rightarrow y$$

- Isso é verdade?



Redes Neurais Recursivas

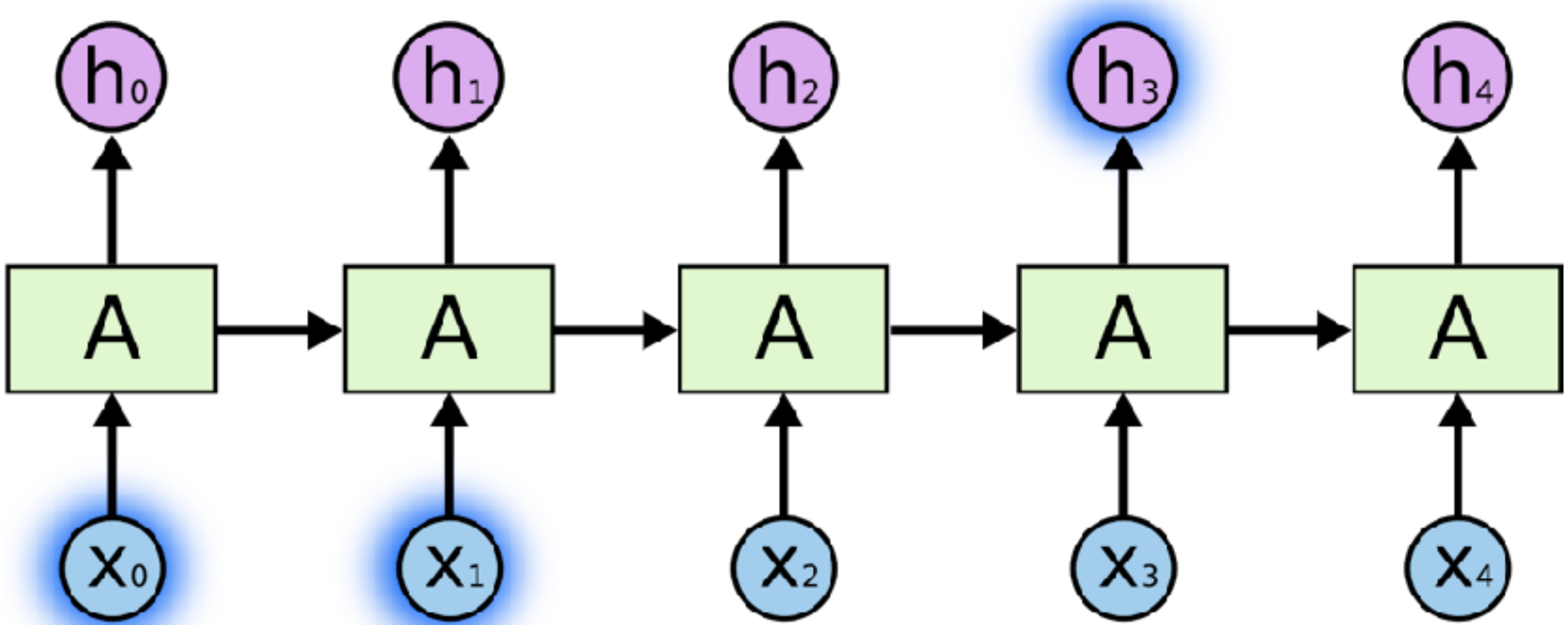
- Para redes neurais recursivas, as entradas influenciam mais do que apenas as suas saídas correspondentes



- Ou seja, de maneira mais geral, uma rede neural recursiva depende do contexto onde esta se aplica

Redes Neurais Recursivas

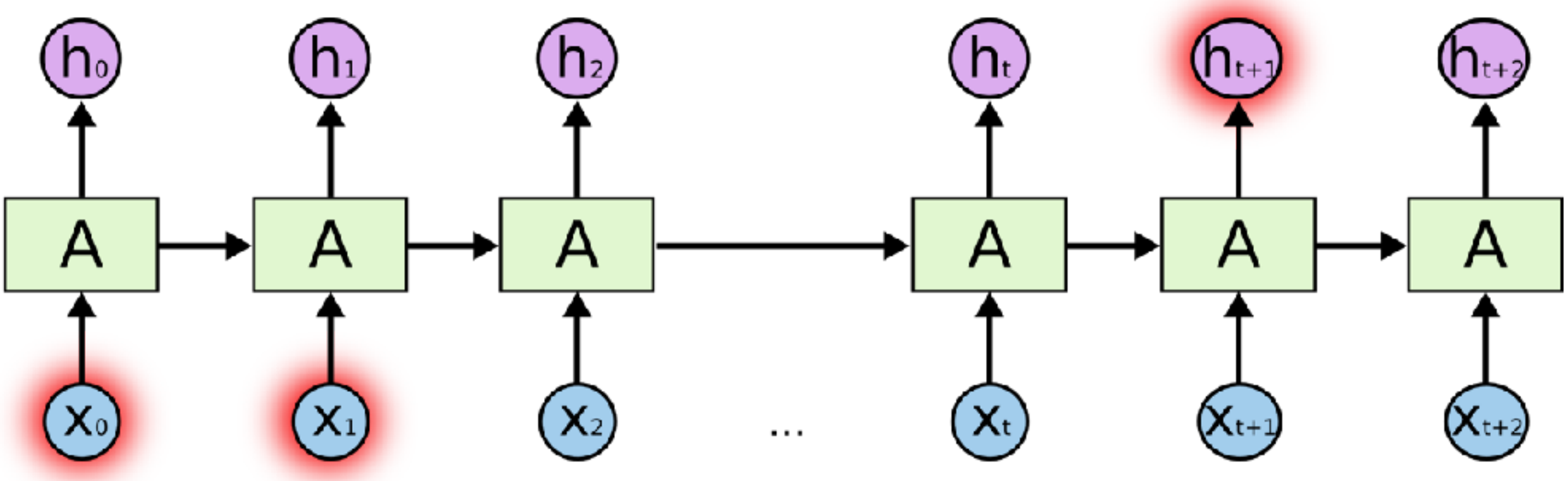
- Assim sendo, tudo é uma questão de contexto
- Complete a frase: as nuvens estão no _____



- Contexto: céu, informação: nuvens
- Memória curta, dentro da frase

Redes Neurais Recursivas

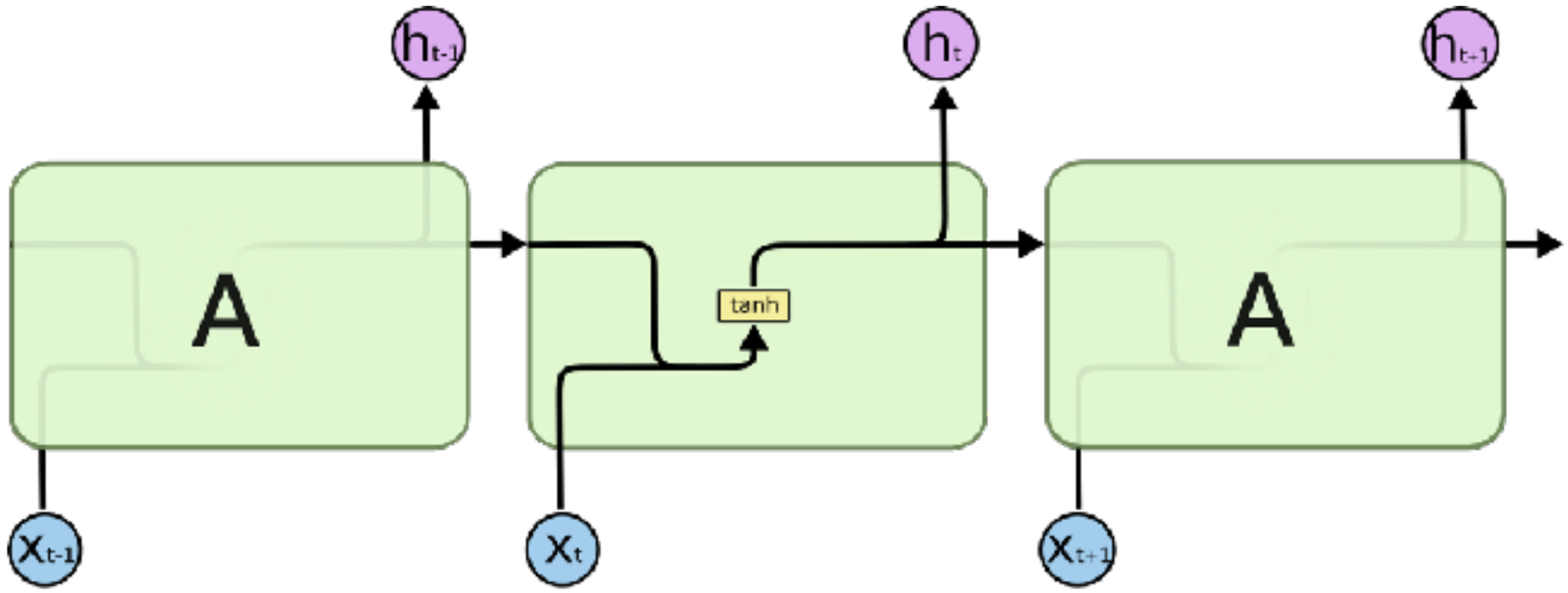
- Assim sendo, tudo é uma questão de contexto
- Complete a frase: Eu sou suíço, logo falo _____



- Contexto: Suíça, informação: linguagem
- Memória de tempo longo, informação não dada

Redes Neurais Recursivas

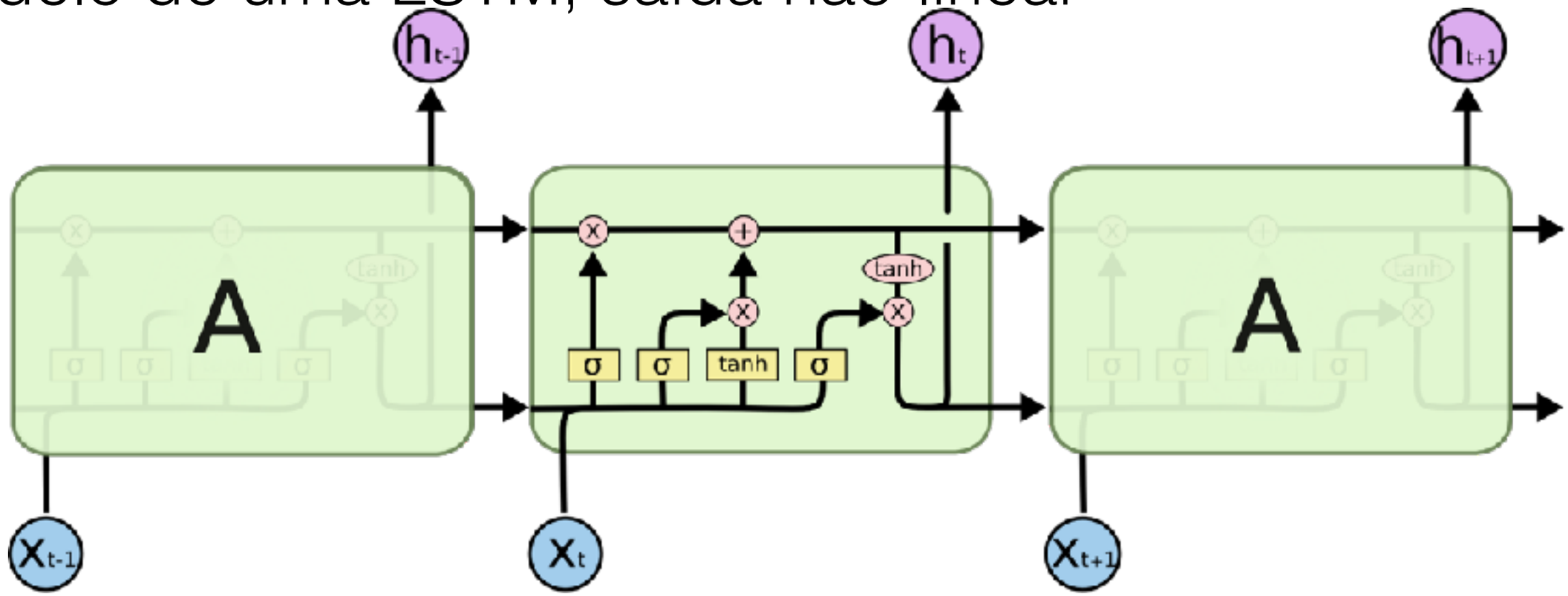
- No contexto de RNN, 90% dos trabalhos se desenvolvem em LSTM (*long short term memory*)
- Modelo básico de uma RNN com saída não-linear



- A saída do estado atual é utilizada para alimentar o estado posterior

Redes Neurais Recursivas

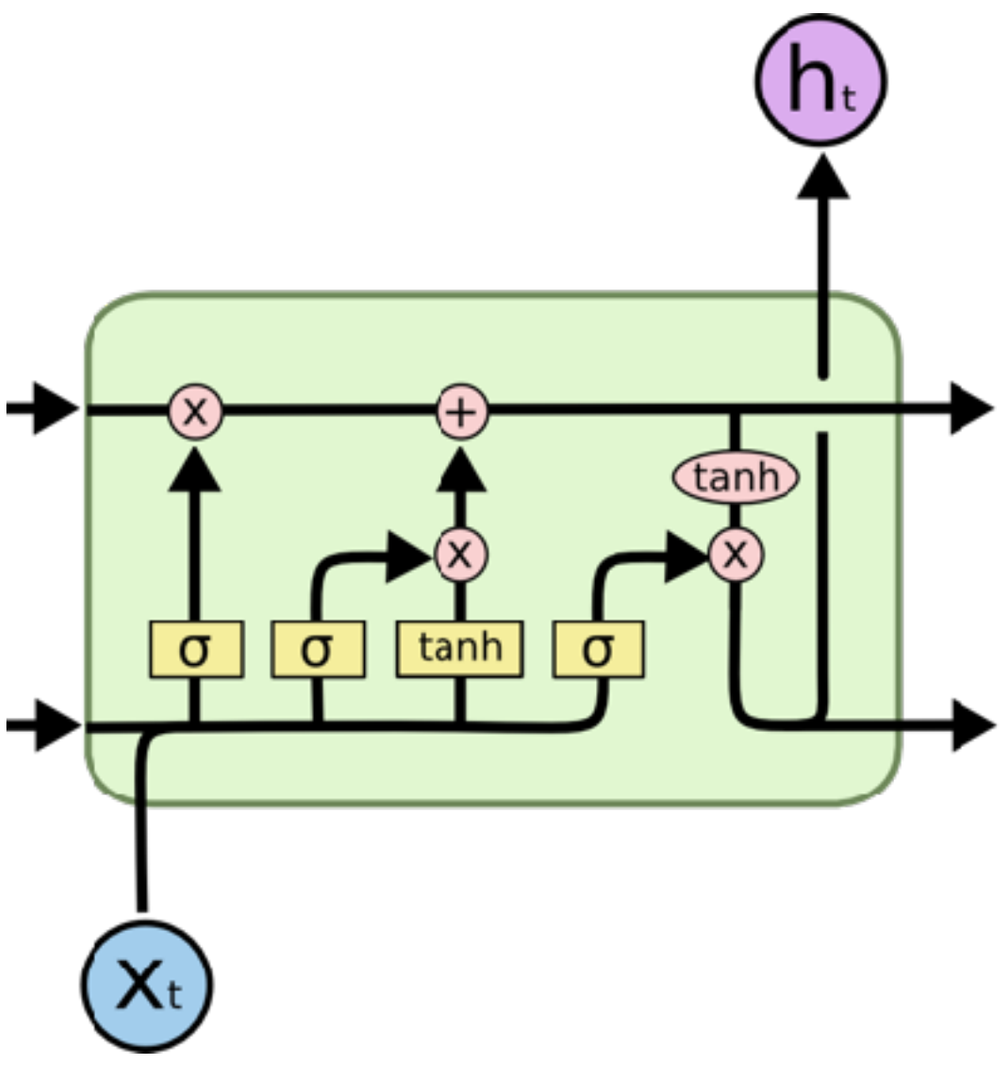
- No contexto de RNN, 90% dos trabalhos se desenvolvem em LSTM (*long short term memory*)
- Modelo de uma LSTM, saída não-linear



- A saída do estado atual é utilizada como entrada de uma composição não-linear para estado posterior

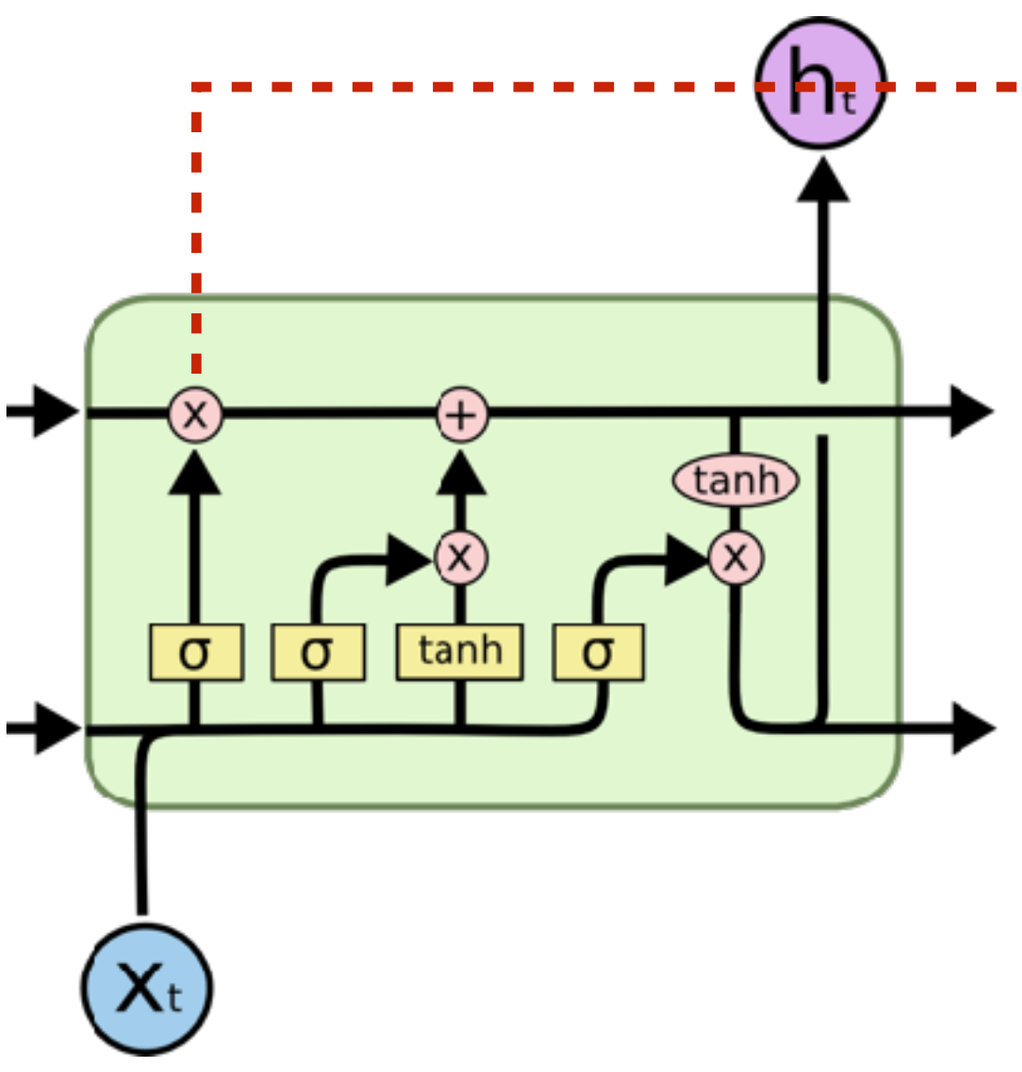
Redes Neurais Recursivas

- No modelo LSTM, temos o chamado *Cell-State*, que é um corredor de informação que podemos chamar de **contexto**
- Este contexto é atualizado por uma composição do contexto (C), das entradas atuais (x) e das saídas anteriores (h)



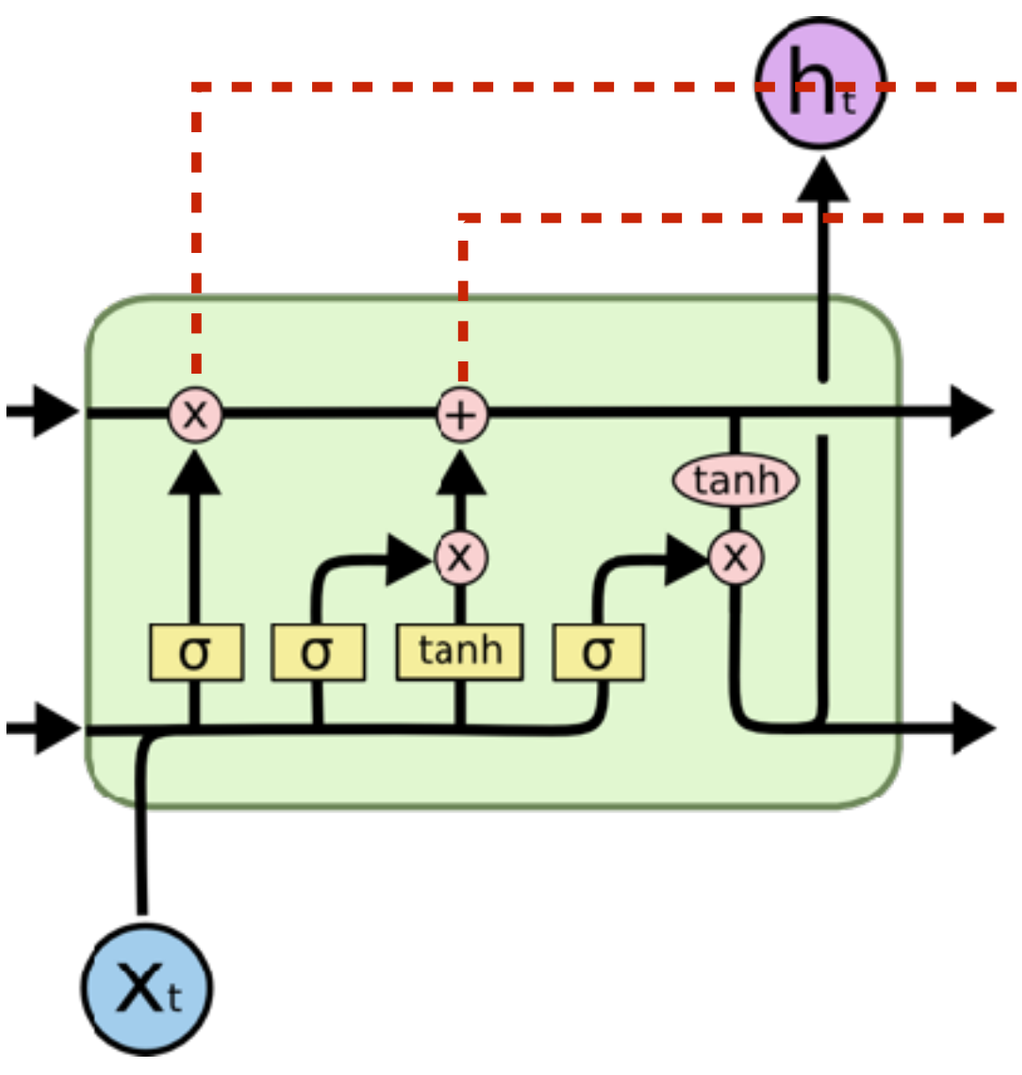
Redes Neurais Recursivas

- No modelo LSTM, temos o chamado *Cell-State*, que é um corredor de informação que podemos chamar de **contexto**
- Este contexto é atualizado por uma composição do contexto (C), das entradas atuais (x) e das saídas anteriores (h)



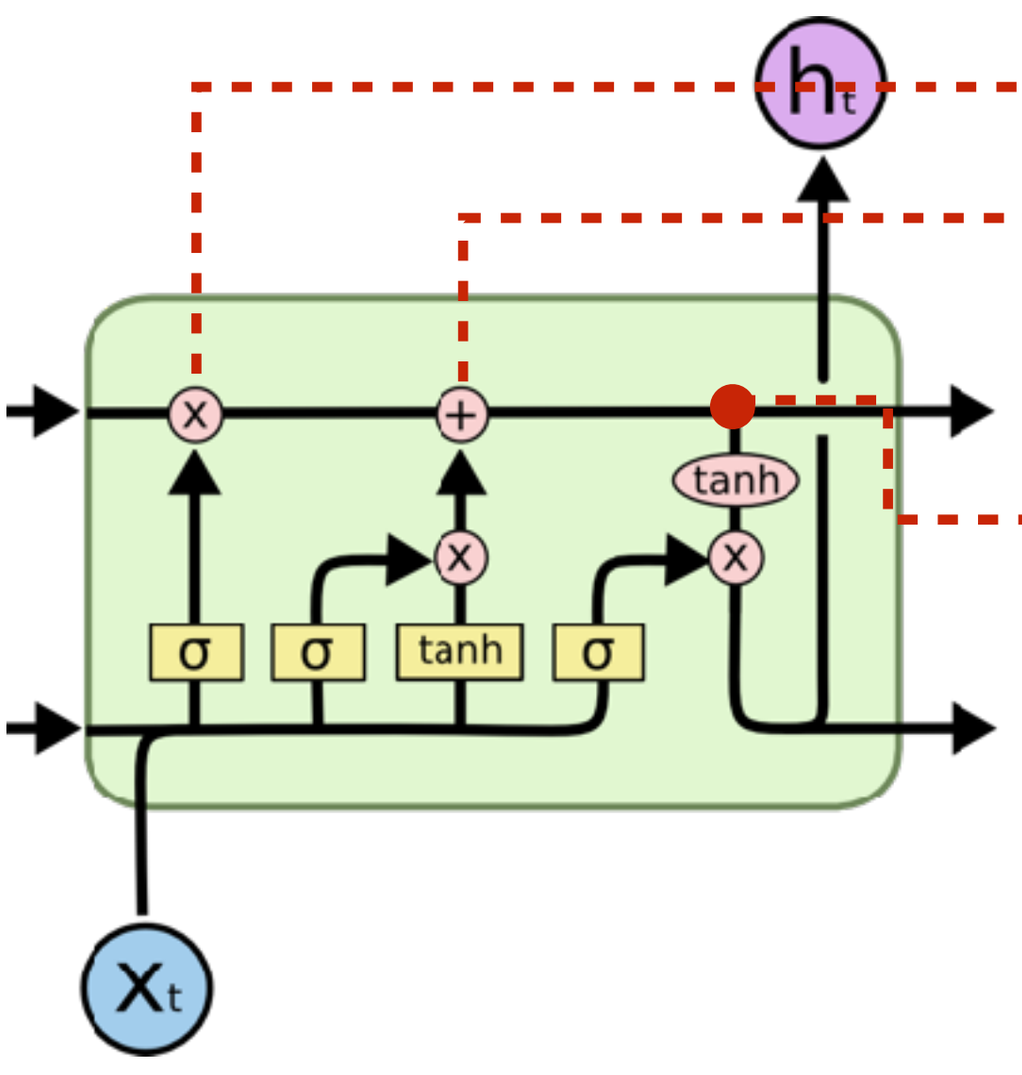
Redes Neurais Recursivas

- No modelo LSTM, temos o chamado *Cell-State*, que é um corredor de informação que podemos chamar de **contexto**
- Este contexto é atualizado por uma composição do contexto (C), das entradas atuais (x) e das saídas anteriores (h)



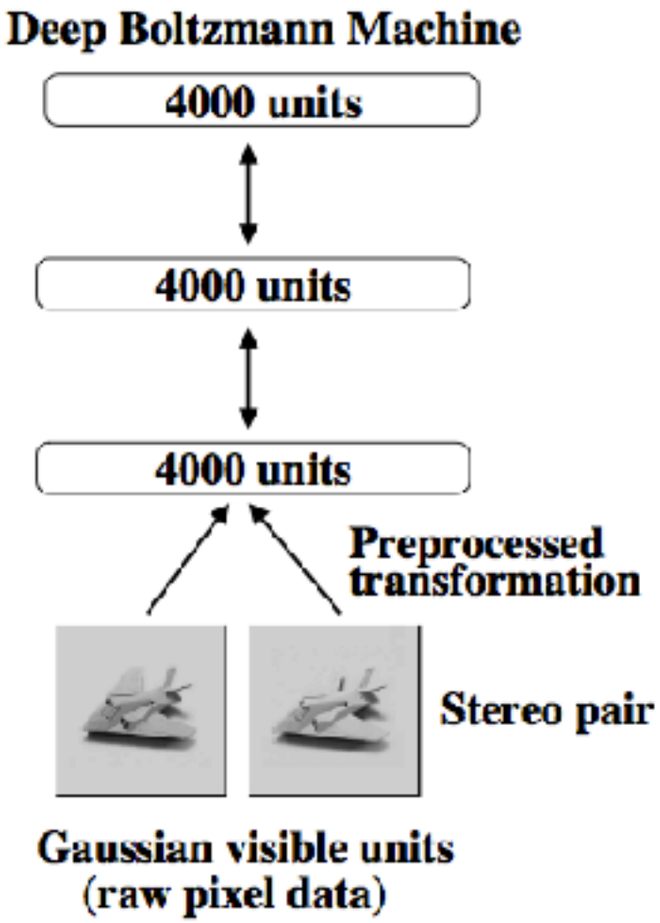
Redes Neurais Recursivas

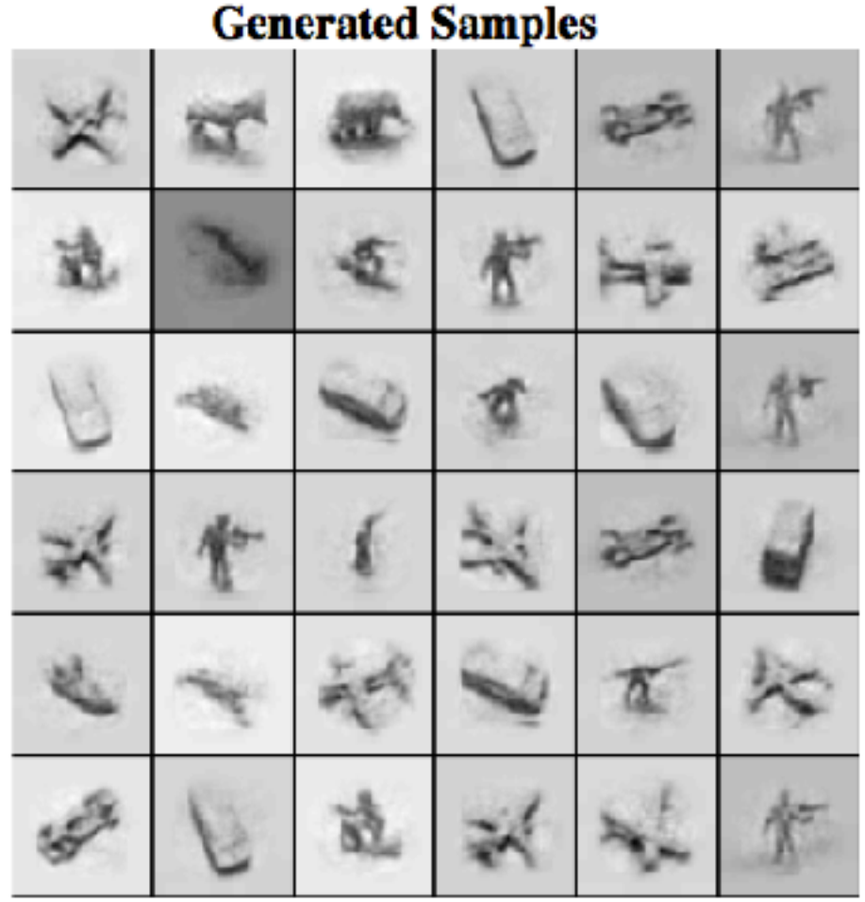
- No modelo LSTM, temos o chamado *Cell-State*, que é um corredor de informação que podemos chamar de **contexto**
- Este contexto é atualizado por uma composição do contexto (C), das entradas atuais (x) e das saídas anteriores (h)



Aplicações

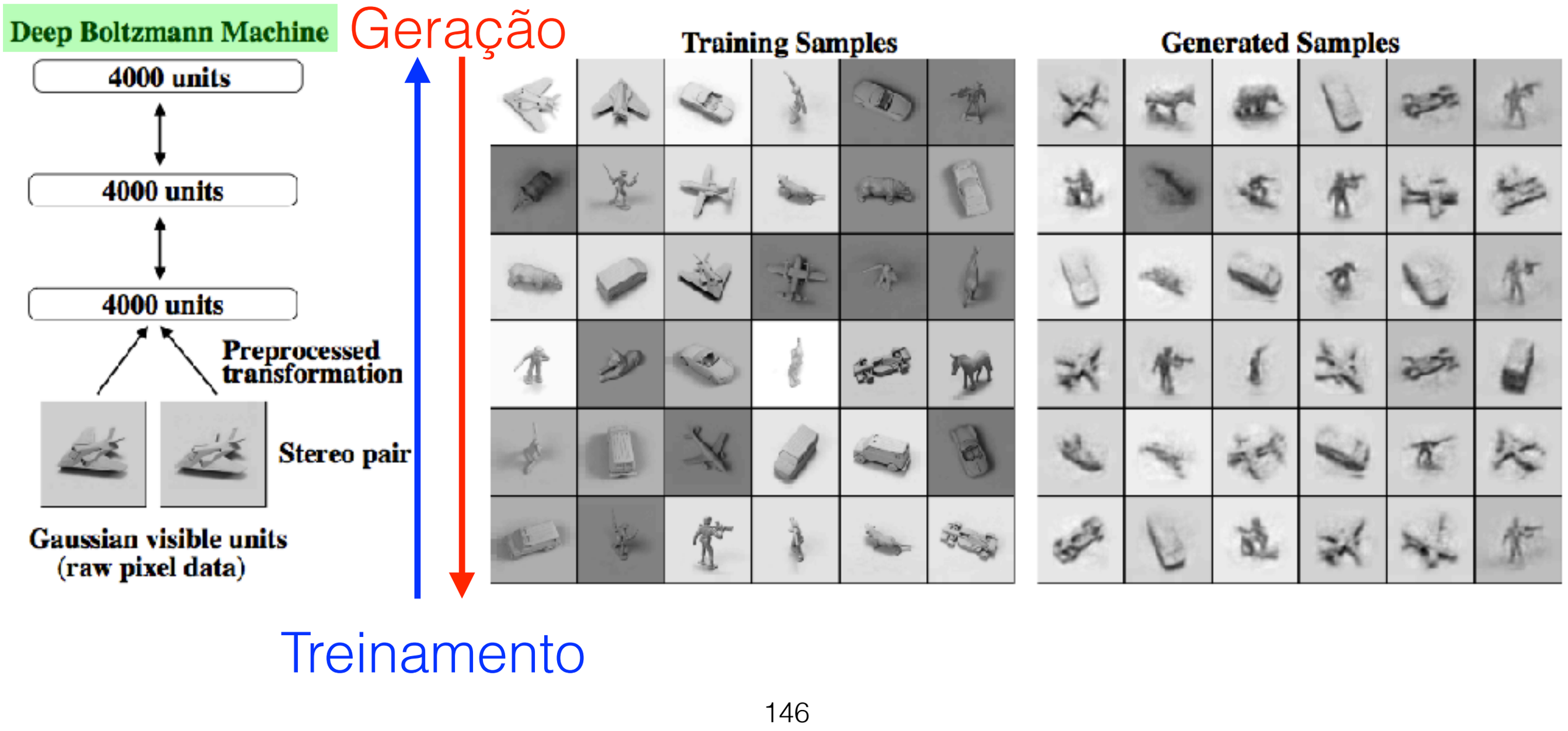
- Deep Belief Networks - um exemplo de modelo gerador é descrito em (Salakhutdinov and Hinton, 2009) que pode gerar imagens aleatórias (imagens 96x96 - 9216 dimensões, reduzidas para 4488 por compressão de





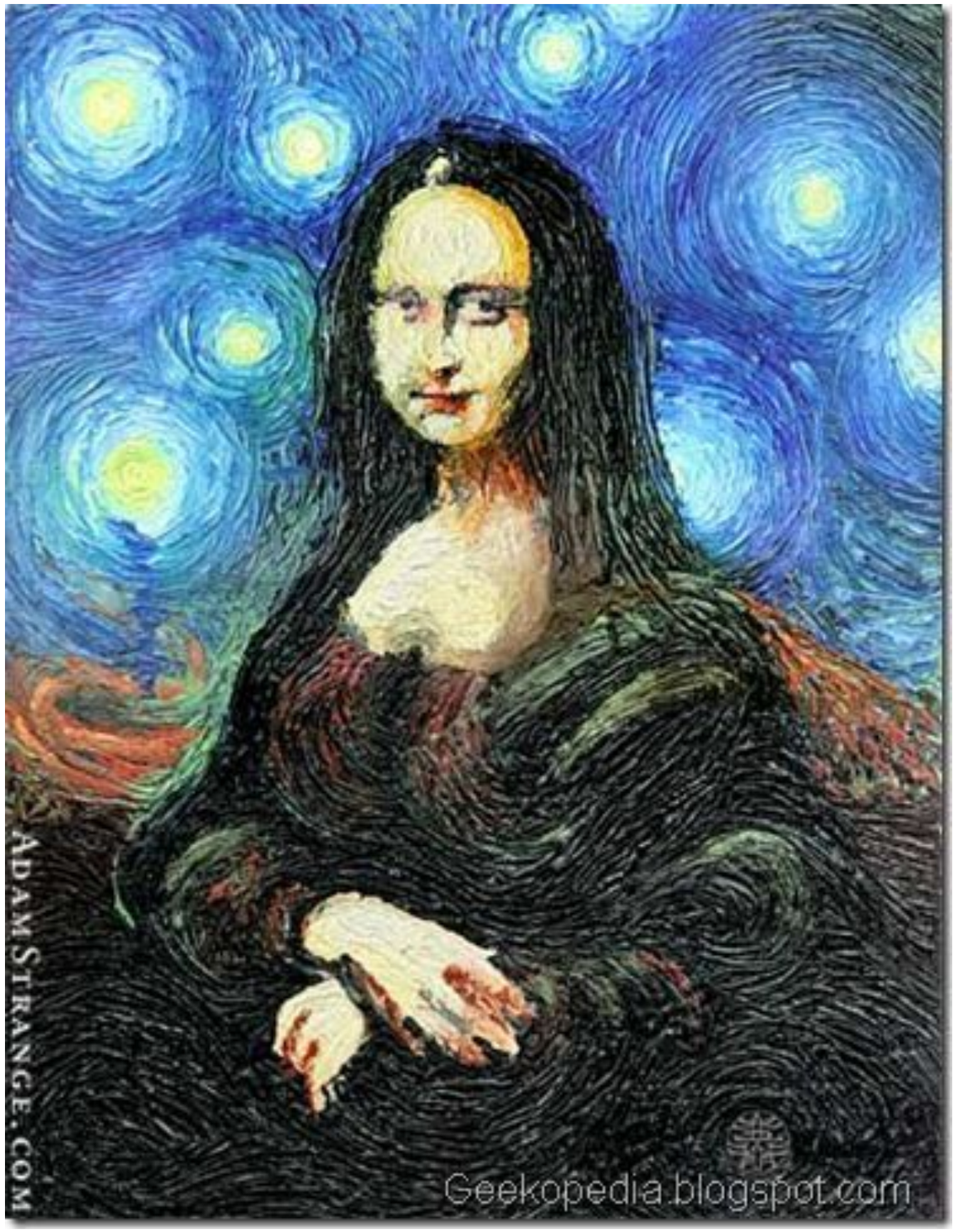
Aplicações

- Deep Belief Networks - um exemplo de modelo gerador é descrito em (Salakhutdinov and Hinton, 2009) que pode gerar imagens aleatórias (imagens 96x96 - 9216 dimensões, reduzidas para 4488 por compressão de



Aplicações

- Filtro de imagens



Aplicações

- Auto-Encoders: em (Hinton and Salakhutdinov, 2006) propõe uma compactação com Deep NN

Dado Compactado com PCA (30 dim) MSE: 135

Dado Compactado com DNN (30 dim) MSE: 126

Dado Original

Aplicações

- Auto-Encoders: em (Hinton and Salakhutdinov, 2006) propõe uma compactação com Deep NN

Dado Reconstruído com PCA Padrão MSE: 13.87

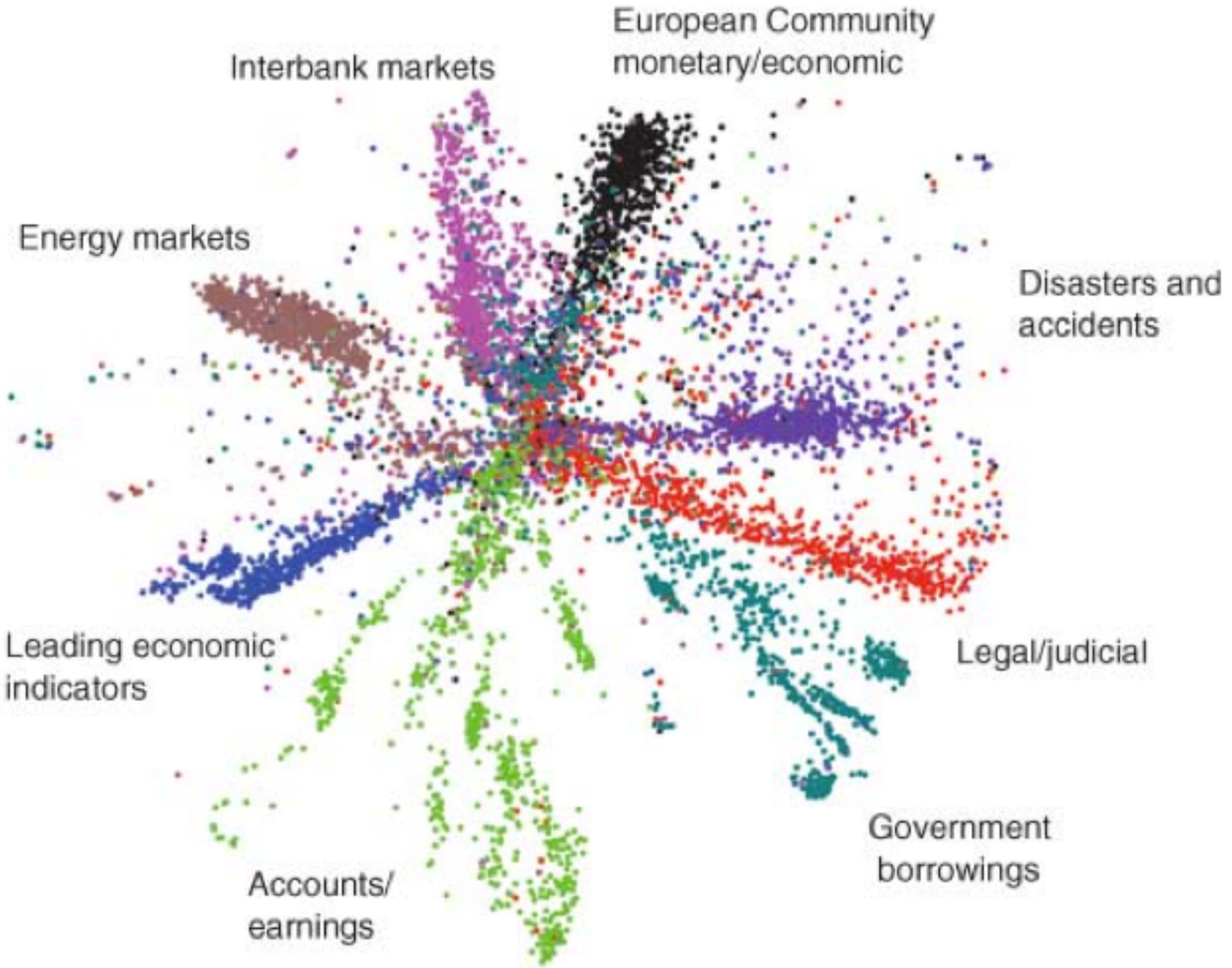
Dado Recons. com PCA extraída com NN MSE: 8.01

Dado Recons. com PCA extraída com DNN MSE: 3.0

Dado Original

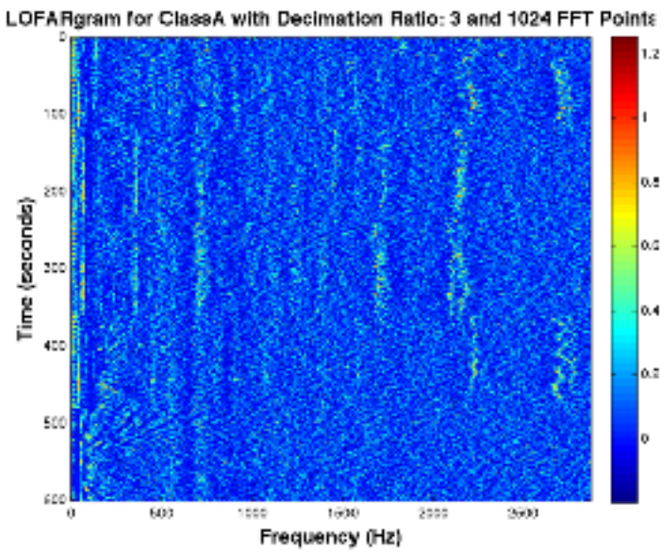
Aplicações

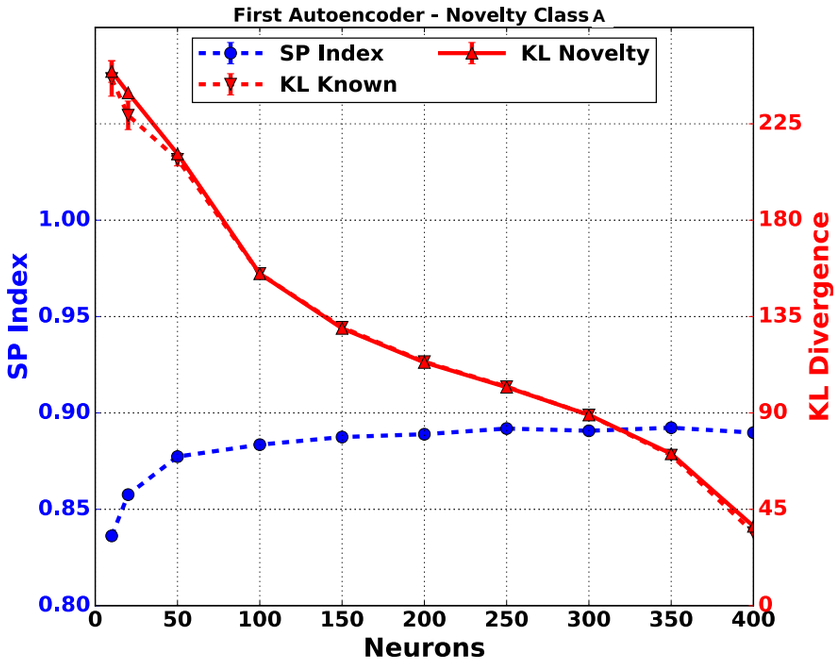
- Auto-Encoders: em (Hinton and Salakhutdinov, 2006) propõe uma compactação com Deep NN

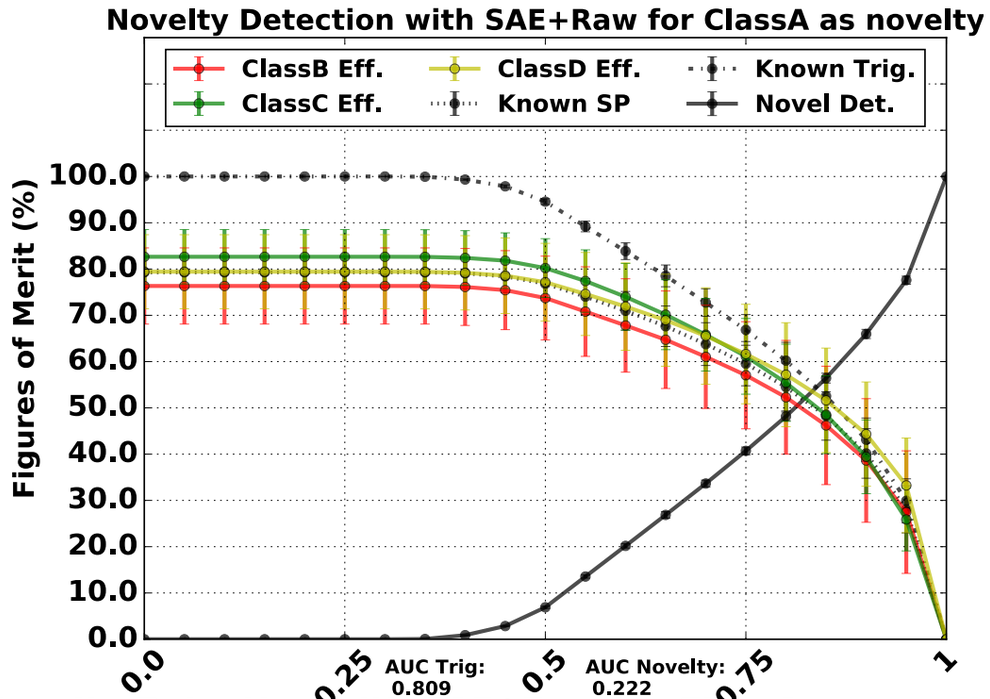


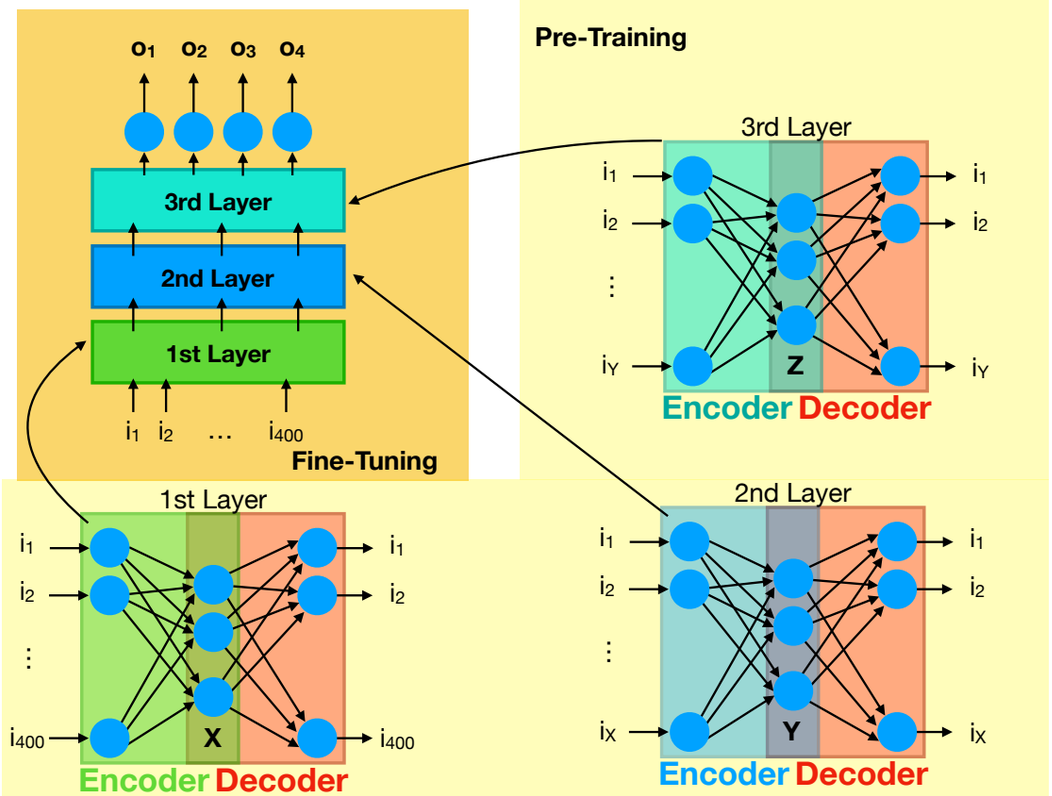
Aplicações

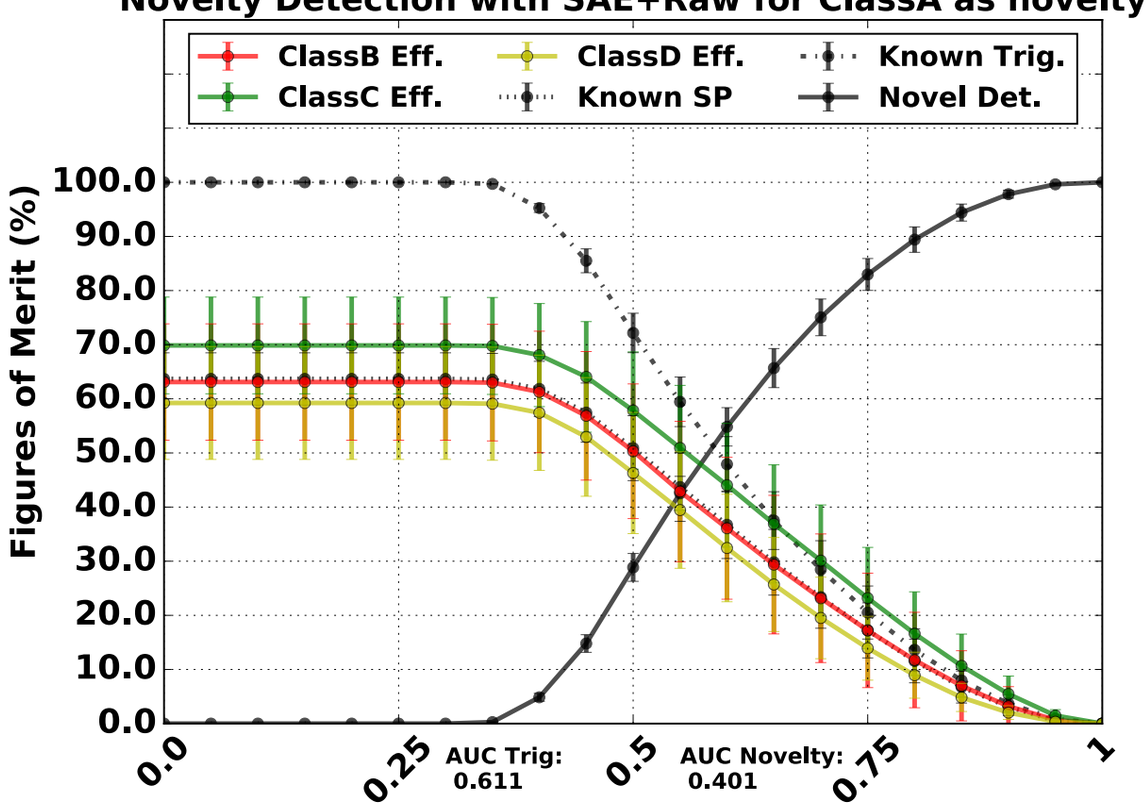
- Auto-Encoders: Detecção de novidade para sistemas de sonar passivo





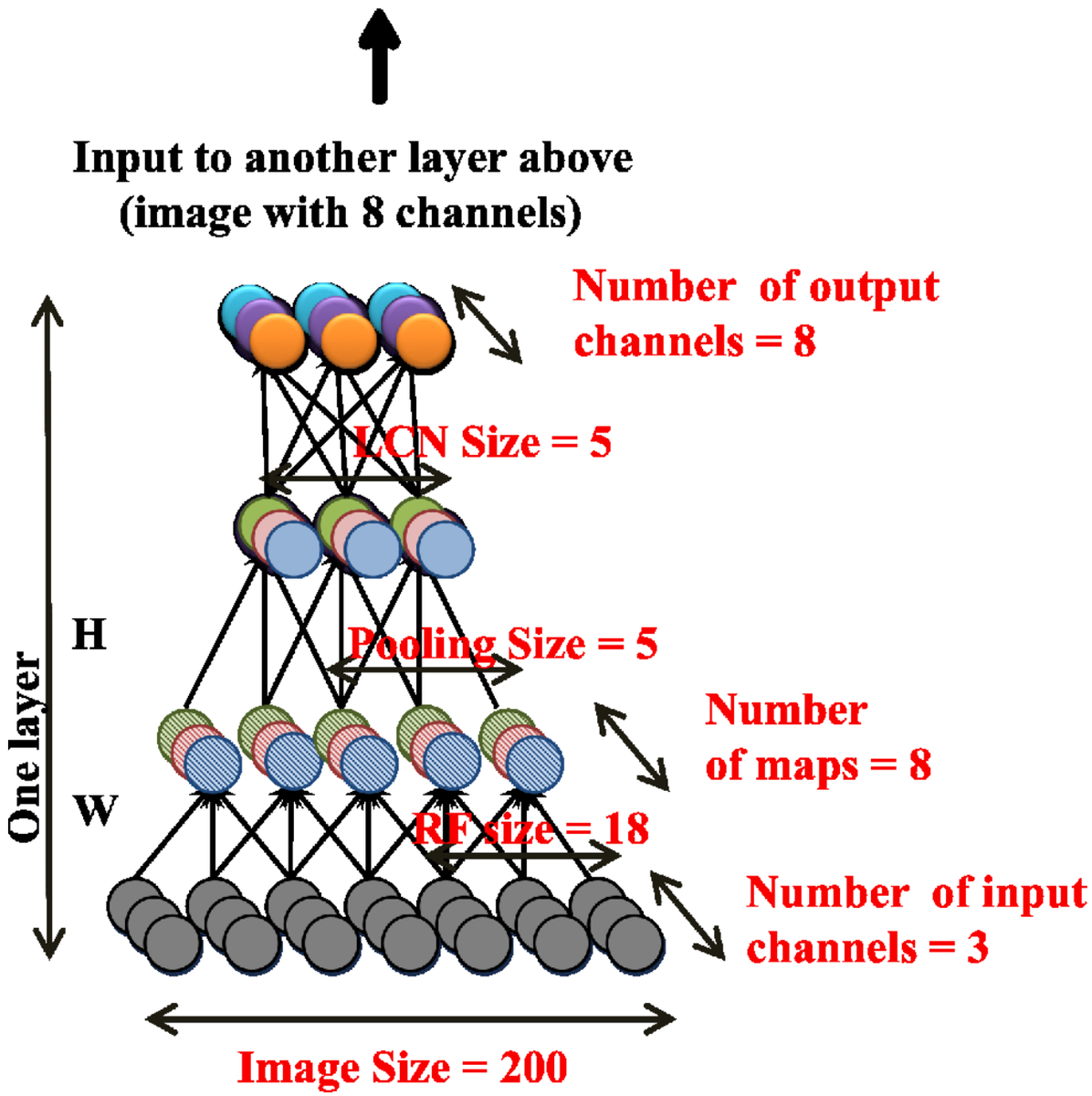






Aplicações

- CNN: Google para classificação de imagens
- 1000 máquinas (16000 núcleos de GPU) x 1 semana

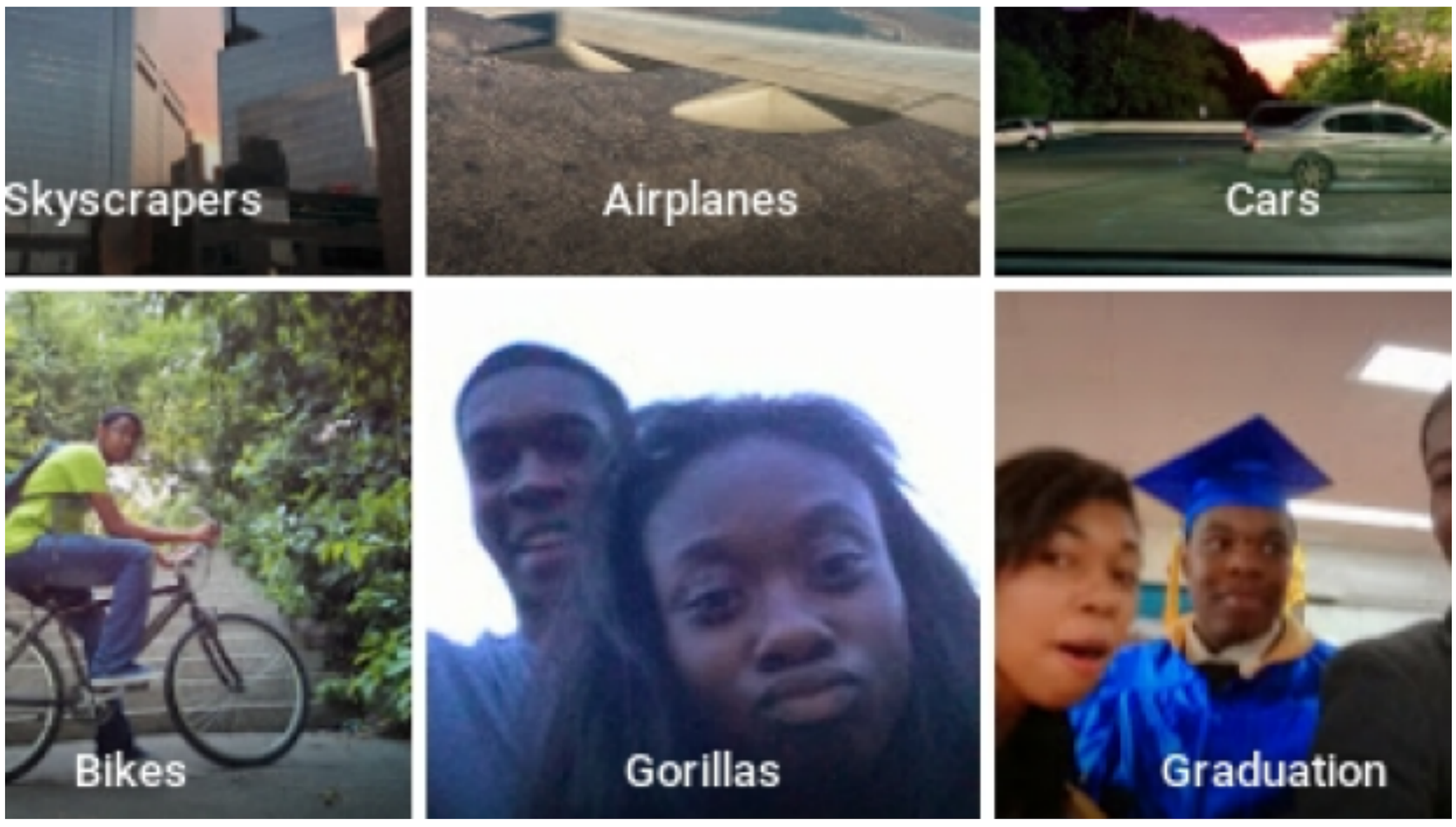


- 1 bilhão de parâmetros a ajustar
- a estrutura ao lado se repete por outras 3x
- Entradas: 200x200
- 10k classes / 22k classes

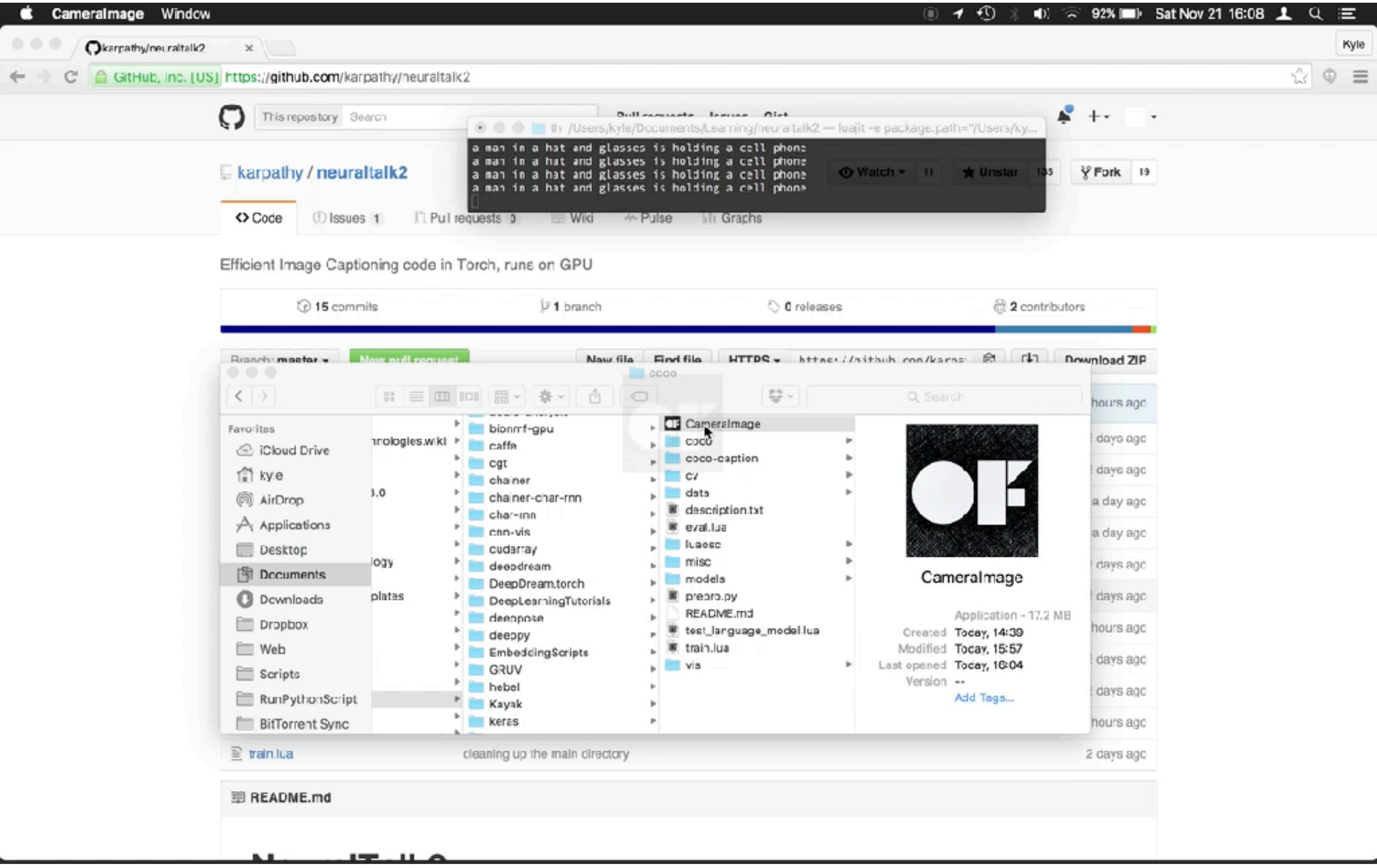
Aplicações

- CNN: Google para classificação de imagens
- 1000 máquinas (16000 núcleos de GPU) x 1 semana

Dataset version	2009 (~9M images, ~10K categories)	2011 (~14M images, ~22K categories)
State-of-the-art	16.7% (Sanchez & Perronnin, 2011)	9.3% (Weston et al., 2011)
Our method	16.1% (without unsupervised pretraining) **19.2%** (with unsupervised pretraining)	13.6% (without unsupervised pretraining) **15.8%** (with unsupervised pretraining)



Aplicações


CameraImage Window
Sat Nov 21 16:08
karpathy/neuraltalk2
GitHub, Inc. [US] https://github.com/karpathy/neuraltalk2
Kyle
This repository Search
a man in a hat and glasses is holding a cell phone
a man in a hat and glasses is holding a cell phone
a man in a hat and glasses is holding a cell phone
a man in a hat and glasses is holding a cell phone
karpathy / neuraltalk2
Watch 11
Unstar 135
Fork 19
Code
Issues 1
Pull requests 0
Wiki
Pulse
Graphs
Efficient Image Captioning code in Torch, runs on GPU
15 commits
1 branch
0 releases
2 contributors
Branch: master
New pull request
New file
Find file
HTTPS
Download ZIP
Favorites
iCloud Drive
kyle
AirDrop
Applications
Desktop
Documents
Downloads
Dropbox
Web
Scripts
RunPythonScript
BitTorrent Sync
CameraImage
coco
coco-caption
cv
data
description.txt
eval.lua
lua
misc
models
prepro.py
README.md
test_language_model.lua
train.lua
vis
Application - 17.2 MB
Created Today, 14:39
Modified Today, 15:57
Last opened Today, 16:04
Version --
Add Tags...
train.lua
cleaning up the main directory
2 days ago
README.md

Aplicações

- LSTM

Conclusões

- Esta apresentação visou mostrar a área de deep learning de maneira rápida
- Esta área de pesquisa tem sido amplamente divulgada em revistas acadêmicas e em revistas de cultura geral
- Foi considerada em diversos meios como: “uma tecnologia que pode quebrar paradigmas”
- Até 2006, a área de redes neurais com mais de 3 camadas foi deixada de lado
- *Layer-wise pre-training* foi o estopim que re-aqueceu a área de Deep Learning (Hinton)
- Desde então, a área se consolidou e obteve destaque

Conclusões

- De maneira geral, Deep Learning tem sido aplicada a diversas áreas e tem obtido sucesso em sua maioria.
- DBN têm sido utilizadas para gerar novos dados mantendo a estrutura estatística do processo gerador
- SAE têm sido utilizados para compactar dados de altas dimensões e acessar estatística de alta ordem
- CNN têm sido utilizadas para acessar informações espaciais em dados com muitas dimensões
- RNN têm sido utilizadas para acessar informações temporais em dados de diversas naturezas.

Fim...ou não?!