INTRODUÇÃO À APRENDIZAGEM PROFUNDA

Eduardo Bezerra (CEFET/RJ)
ebezerra@cefet-rj.br

Sobre mim

- Professor do CEFET/RJ (Rio de Janeiro) desde 2005.
- Docente permanente no recém-criado Programa de Pós-Graduação em Ciência da Computação (PPCIC).
  - inscrições abertas para 2017!
- Doutor em Engenharia de Sistemas e Computação pela COPPE/UFRJ (2006).
- Coautor do livro Data Mining: Conceitos, técnicas, algoritmos, orientações e aplicações, em sua 2a edição, Elsevier/Brasil.
- Áreas de interesse: Inteligência Artificial, Mineração de Dados e Recuperação de Informações.

Sobre o minicurso

- 3,5 horas (14:00h às 16:00h; 16:30h às 18:00h)
- Uma “introdução” mesmo...
- Prioriza o “entendimento conceitual”.
- Relativamente pouco conteúdo sobre aspectos
  - teóricos
  - de implementação

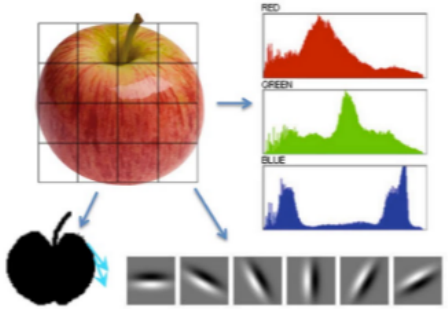
Visão geral

- Considerações Iniciais
- Conceitos Básicos de Redes Neurais Artificiais
- Redes Autocodificadoras
- Redes Convolucionais
- Redes Recorrentes
- Técnicas para Treinamento de Redes Profundas
- Considerações Finais

Considerações Iniciais

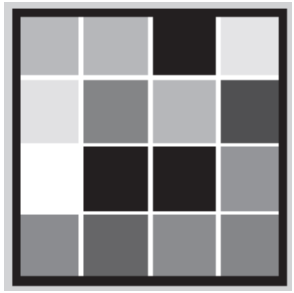
O que é Aprendizagem Profunda?

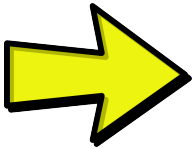
- Problema de treinar redes neurais artificiais que realizam o aprendizado de características de forma hierárquica.
- Características nos níveis mais altos da hierarquia são formadas pela combinação de características de mais baixo nível.

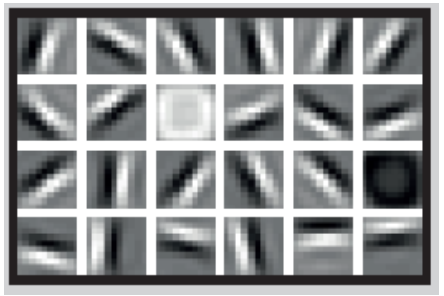


Créditos: Victor Lavreko

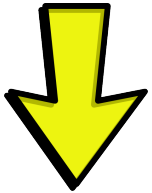
Hierarquia de características

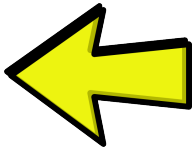


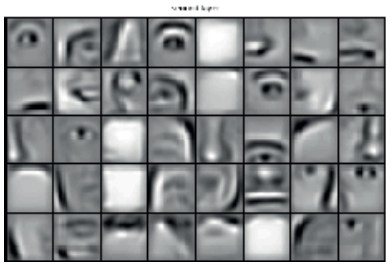




Composição de funções é um dos pilares da aprendizagem profunda.

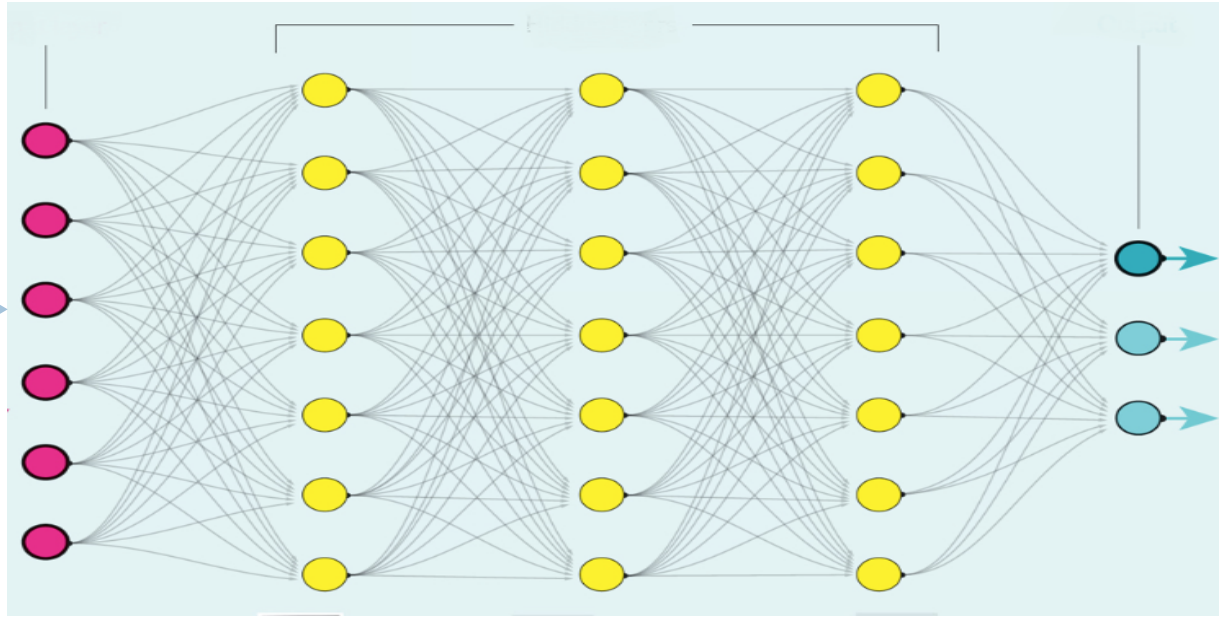




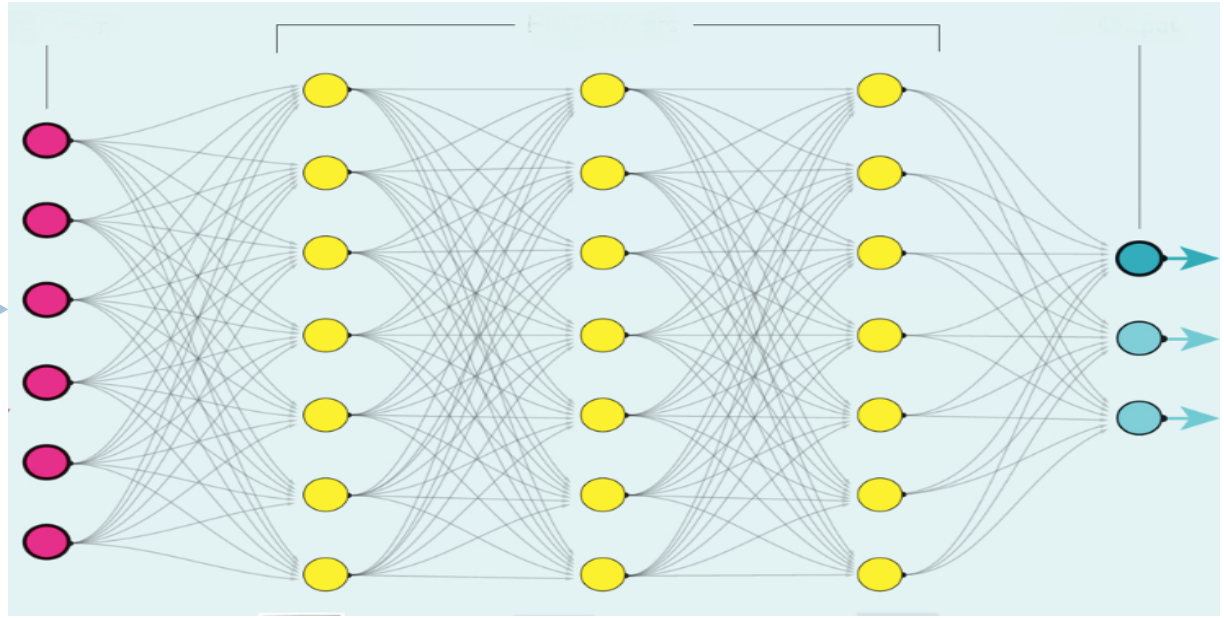


Fonte: Andrew Ng apud The Learning Machines (Nature, 2014)

Composição de características



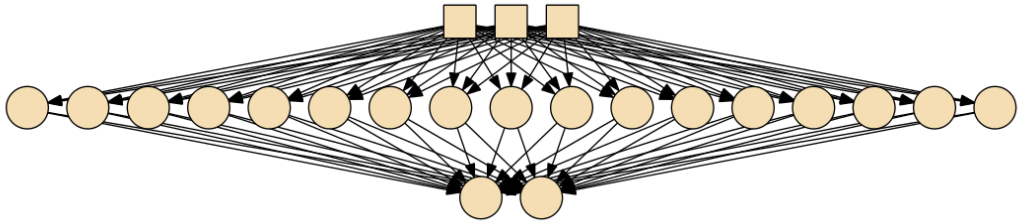
Composição de características



Jayme

Por que Aprendizagem Profunda?

- Teorema da Aproximação Universal (1991)
  - "Com uma única camada oculta, é possível aproximar qualquer função contínua limitada, contanto que se possa definir a quantidade suficiente de neurônios"

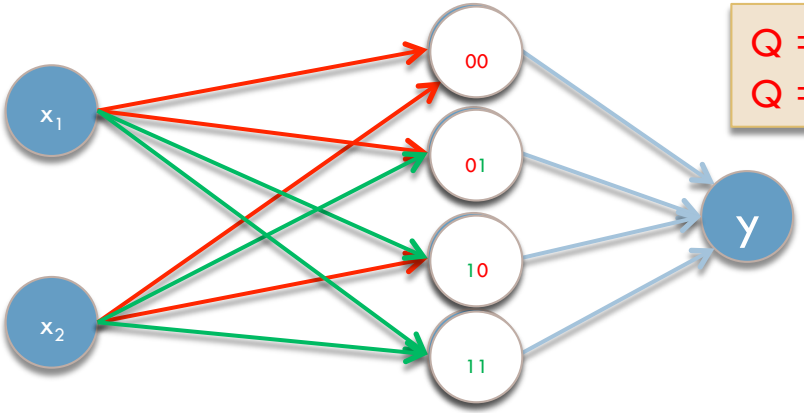


Então, por que utilizar de redes profundas?!

Approximation capabilities of multilayer feedforward networks (1991)

Por que Aprendizagem Profunda?

- Teoria **não** dá garantias sobre o *limite superior* da quantidade Q de neurônios na camada oculta.
  - Dependendo do problema, Q pode crescer de forma **exponencial** com o tamanho da entrada.



Q = 4 → entrada de tamanho 2
Q = ? → entrada de tamanho 100

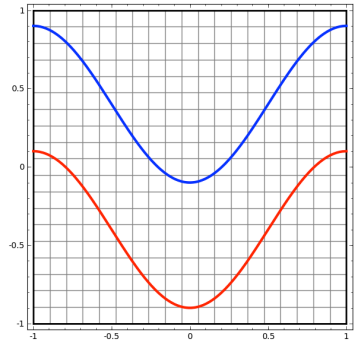
Por que Aprendizagem Profunda?

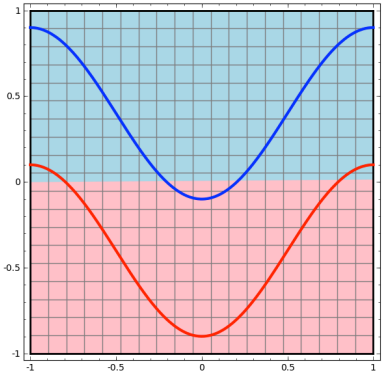
- Com mais camadas ocultas, funções podem ser representadas por uma quantidade de unidades ocultas que cresce de forma polinomial com o tamanho da entrada.
- **Expressividade:** é possível modelar uma função altamente não linear formada de uma hierarquia de funções não lineares mais simples.

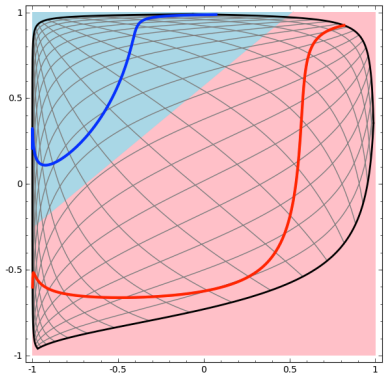
On the expressive power of deep neural networks, 2016;
Exponential expressivity in deep neural networks through transient chaos, 2016
On the Expressive Power of Deep Architectures, 2011.
Larochelle et al. Exploring strategies for training deep neural networks. J. Mach. Learn. Res., 10:1–40, 2009.

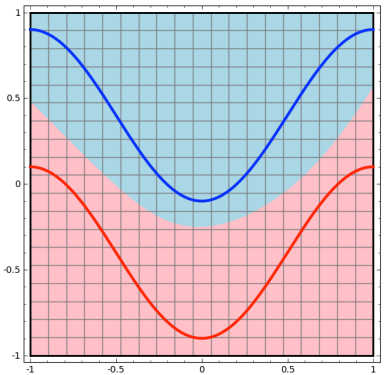
Por que Aprendizagem Profunda?

Expressividade









Fonte: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Por que Aprendizagem Profunda?

- Quanto mais dados, melhor!

"[…] what we're seeing consistently is that the bigger you can run these models, the better they perform. If you train one of these algorithms on one computer, you know, it will do pretty well. If you train them on 10, it will do even better. If you train on 100, even better. And we found that when we trained it on 16,000 CPU cores, […], that was the best model we were able to train."

"Now, why do we need so many processors? […] The point was to have a software, maybe a little simulated baby brain."

Andrew Ng

http://www.npr.org/2012/06/26/155792609/a-massive-google-network-learns-to-identify

Por que Aprendizagem Profunda?

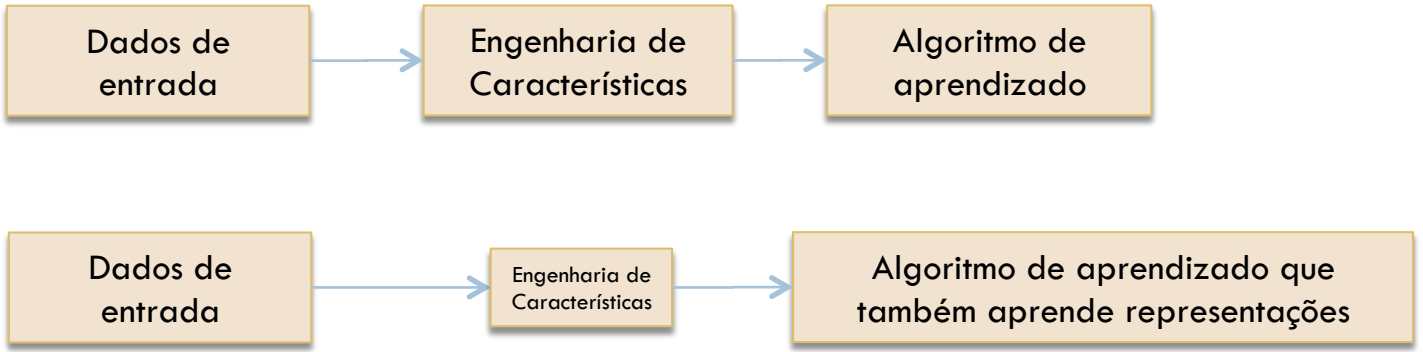
- Quanto mais dados, melhor!

"What was wrong in the 80's is that we didn't have enough data and we didn't have enough computer power"

Geoffrey Hinton

Por que Aprendizagem Profunda?

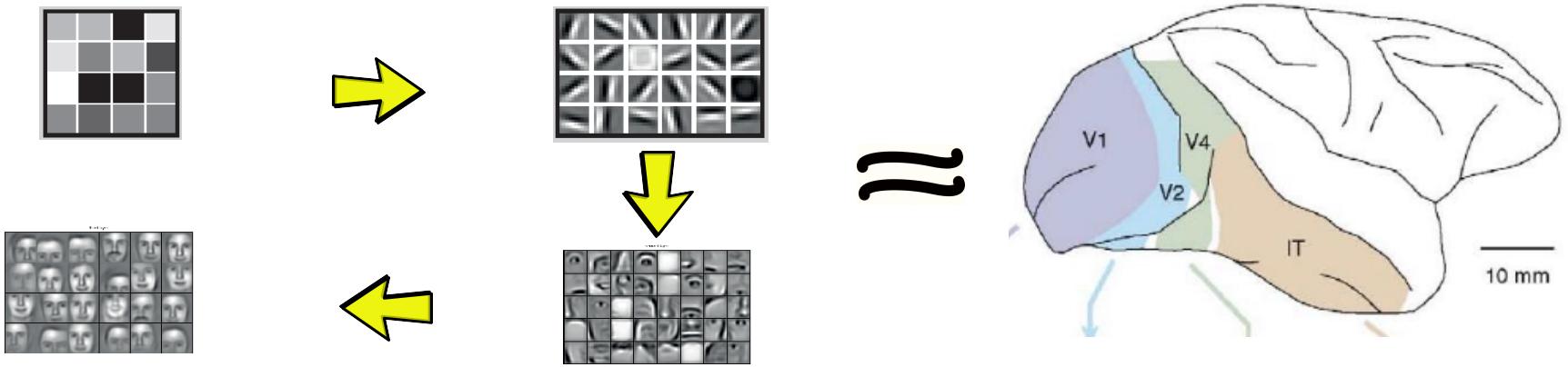
- Aprendizagem de representações automatizada, ou pelo menos simplificada.



Engenharia de Características (*Feature Engineering*): usar conhecimento específico de domínio ou métodos automáticos para gerar, extrair ou alterar características dos dados de entrada.

Por que Aprendizagem Profunda?

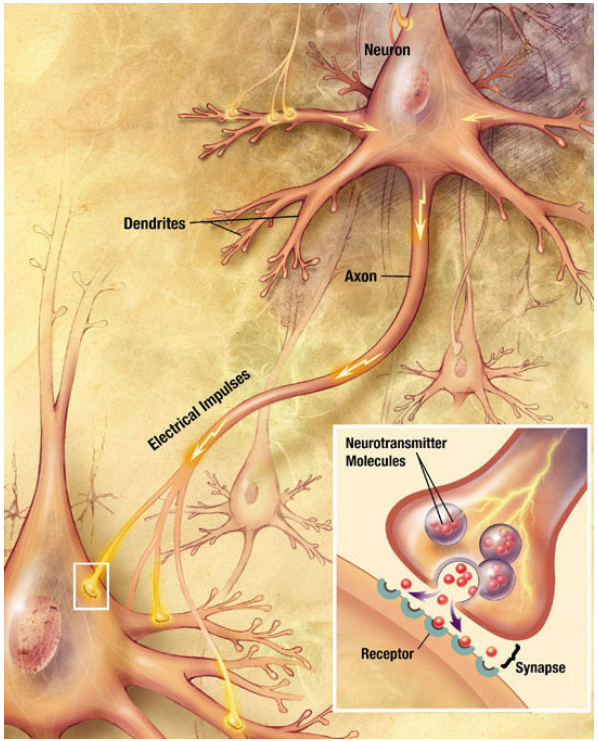
- Biologicamente plausível!



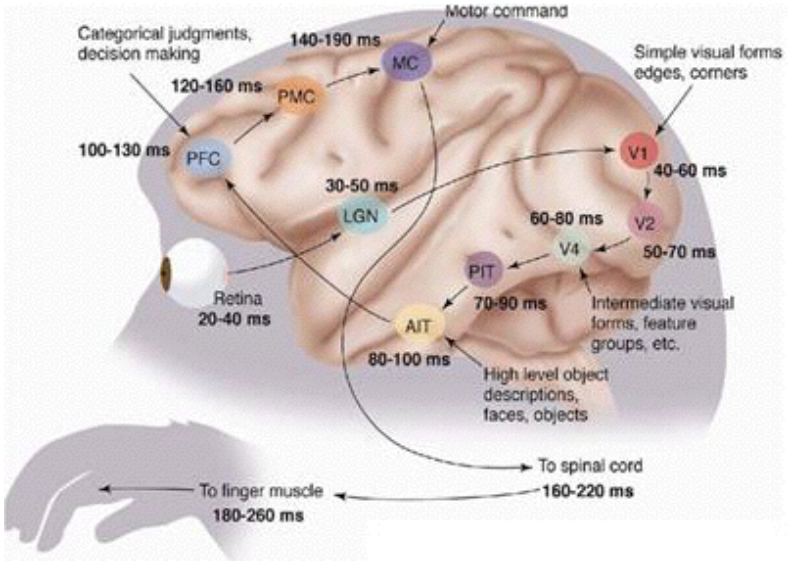
Cérebros são engenheiros de características!

Conceitos Básicos de Redes Neurais Artificiais

(Modelo de um) neurônio real

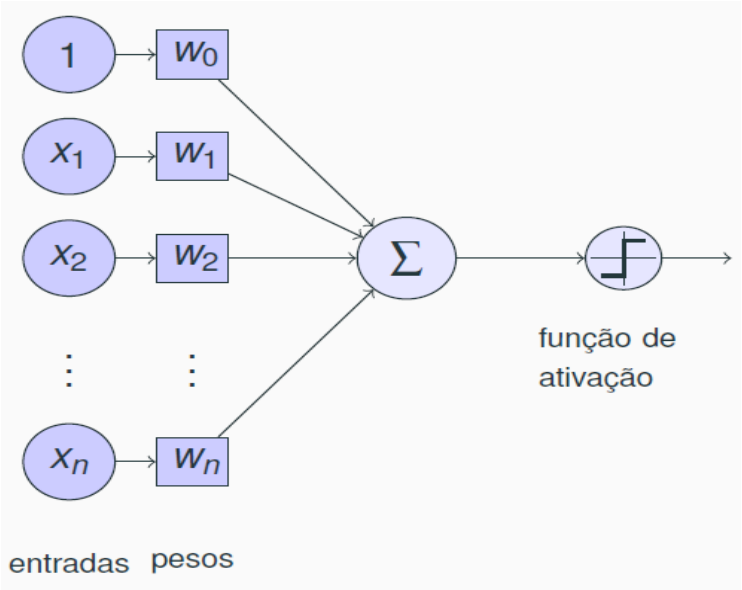


Fonte: https://wikimedia.org



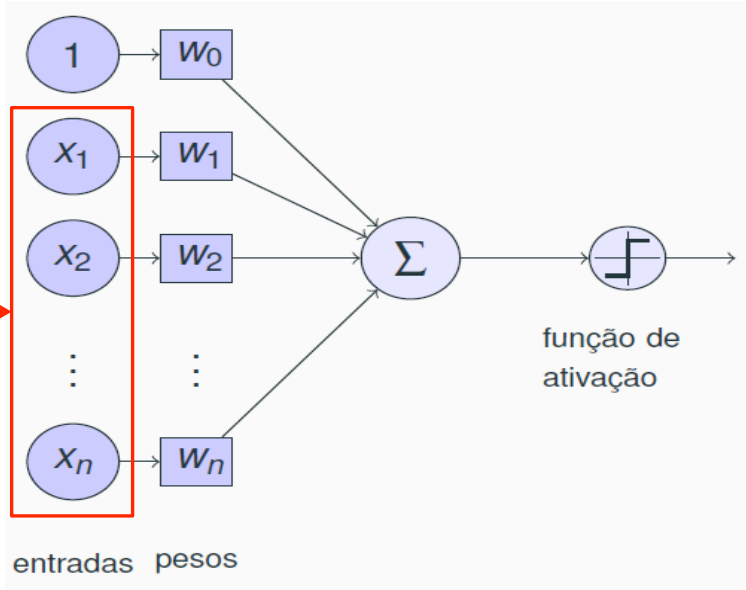
Créditos: SIMON THORPE

Neurônio artificial



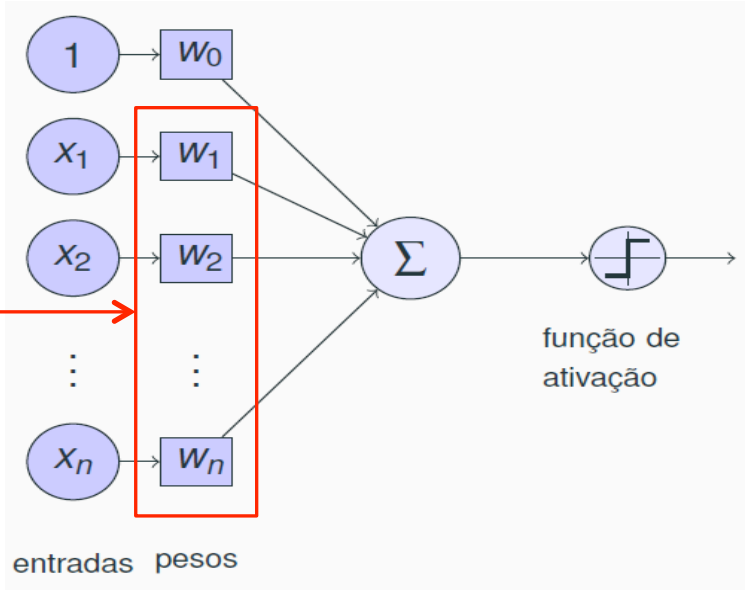
Neurônio artificial - pré-ativação

$$a(\mathbf{x}) = b + \sum_i w_i x_i$$



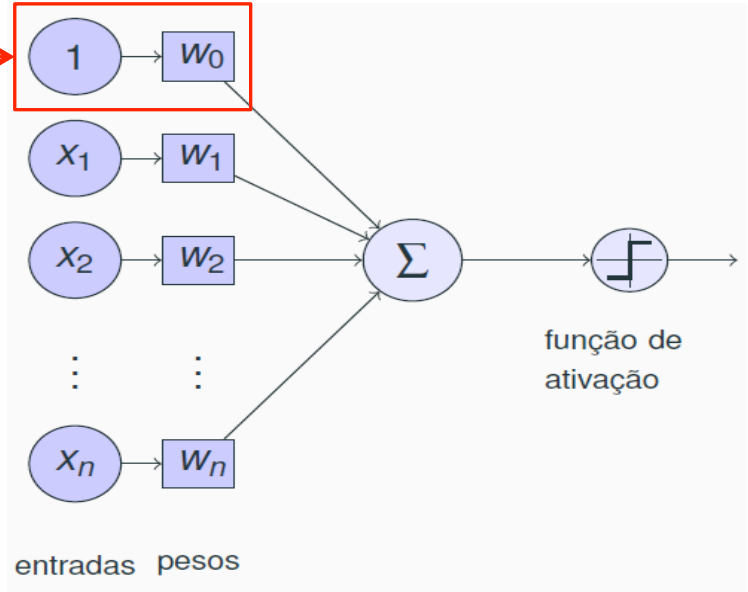
Neurônio artificial - pré-ativação

$$a(\mathbf{x}) = b + \sum_i w_i x_i$$



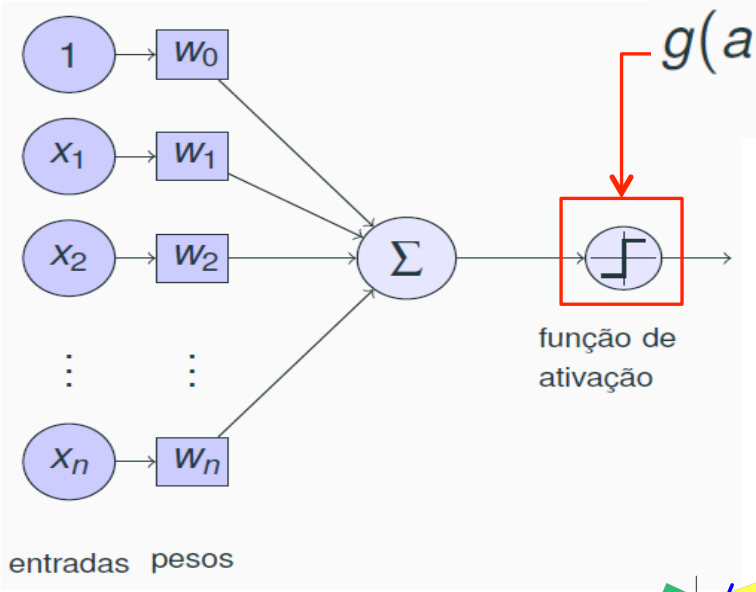
Neurônio artificial - pré-ativação

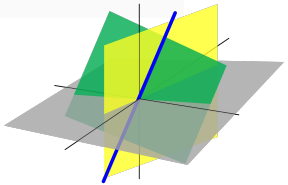
$$a(\mathbf{x}) = b + \sum_i w_i x_i$$



Neurônio artificial - ativação (não-linearidade)

$$g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

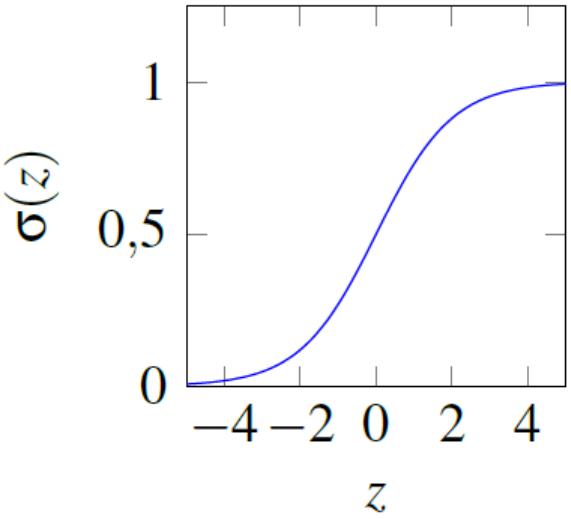




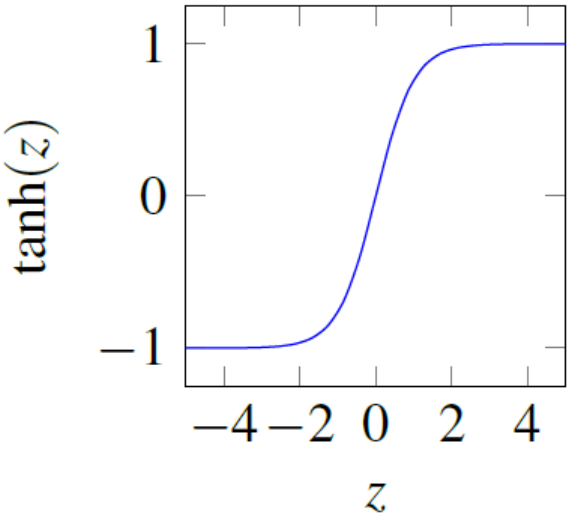
"A composição de transformações lineares também é uma transformação linear"

Não linearidades são necessárias para que a rede aprenda representações complexas dos dados.

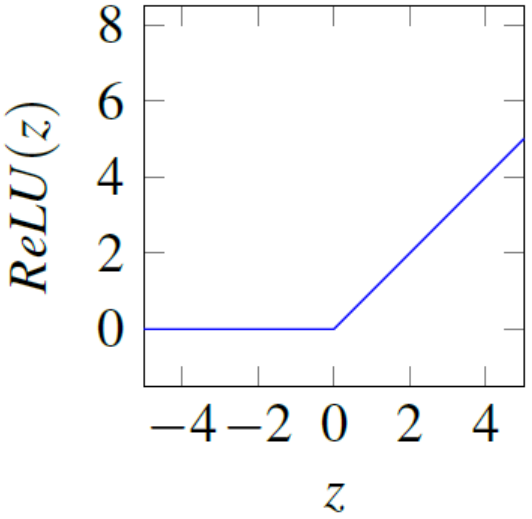
Neurônio artificial - ativação (não-linearidade)



(a) Logística.

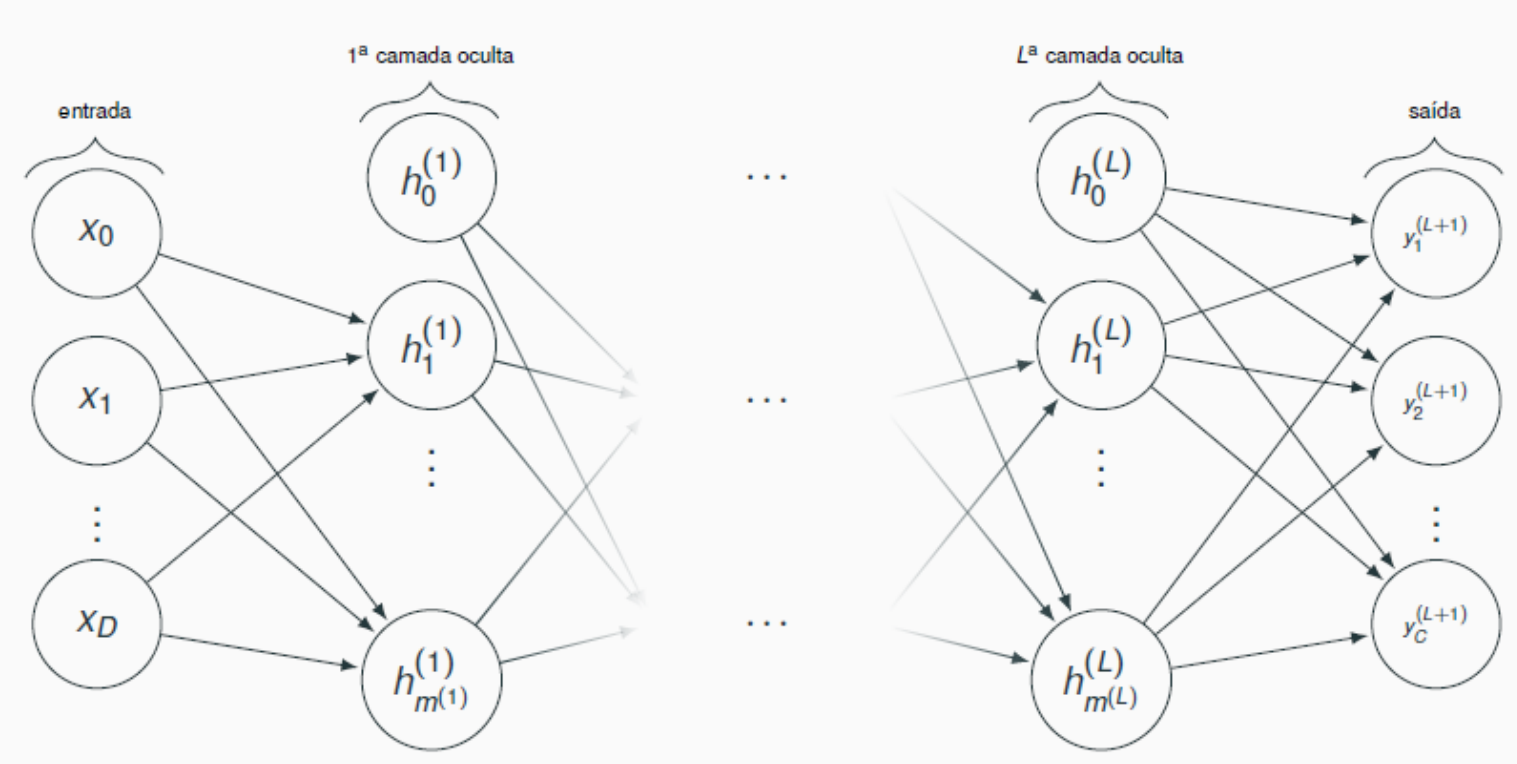


(b) Hiperbólica tangente.



(c) Retificada linear.

Rede neural artificial

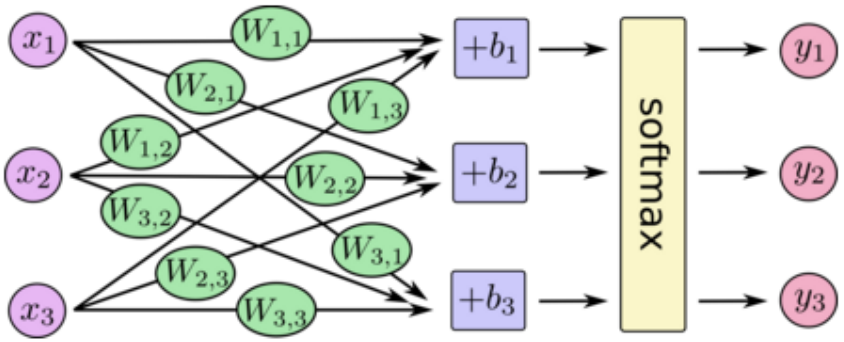


Rede de propagação adiante (feedforward NN)

softmax

$$y_i = o\left(a_i^{(L+1)}\right) = \frac{e^{a_i^{(L+1)}}}{\sum_{j=1}^{C} e^{a_j^{(L+1)}}}$$

- Quando o propósito de uma RNA é realizar a tarefa de classificação, é comum usar a função softmax.



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{pmatrix}$$

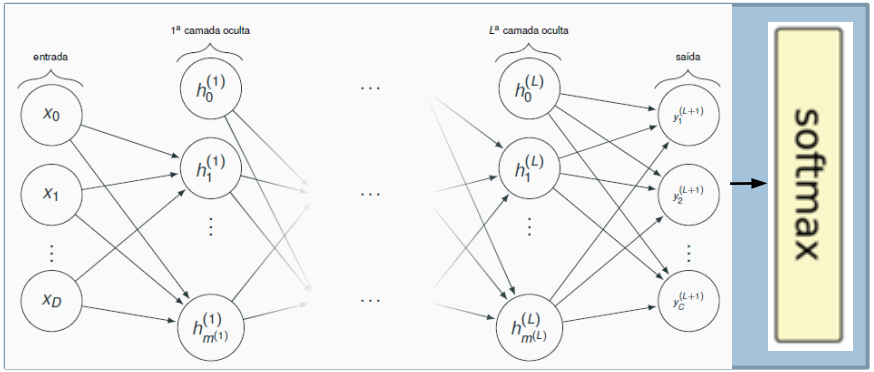
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

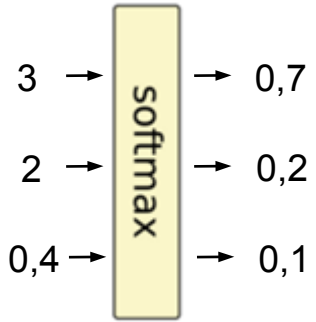
Fonte: https://www.tensorflow.org/versions/r0.9/tutorials/mnist/beginners/index.html

softmax

$$y_i = o\left(a_i^{(L+1)}\right) = \frac{e^{a_i^{(L+1)}}}{\sum_{j=1}^{C} e^{a_j^{(L+1)}}}$$

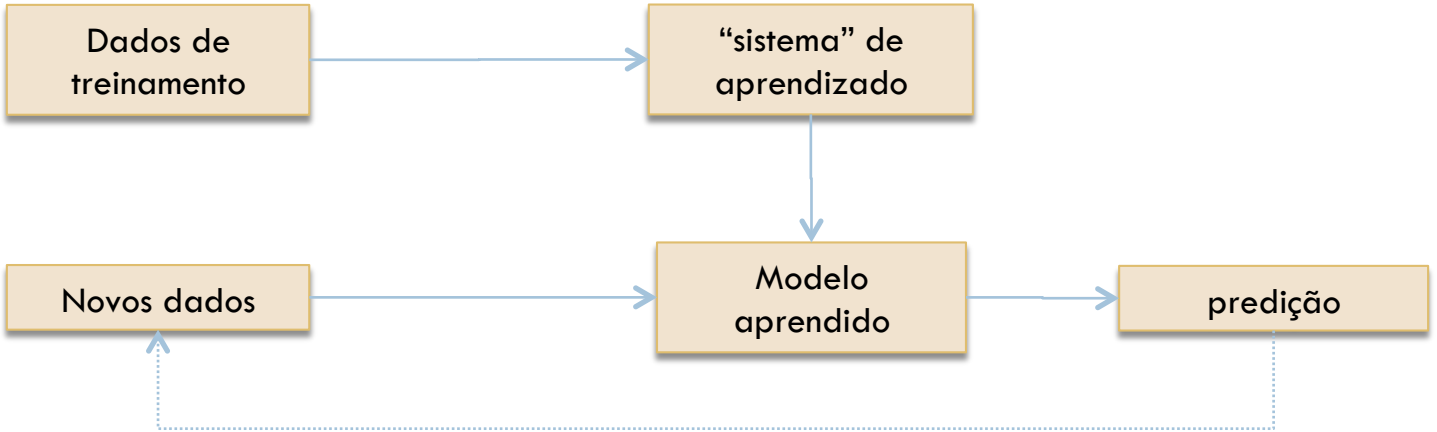
- Permite interpretar os valores da camada de saída como uma distribuição de probabilidades.





Treinamento Supervisionado

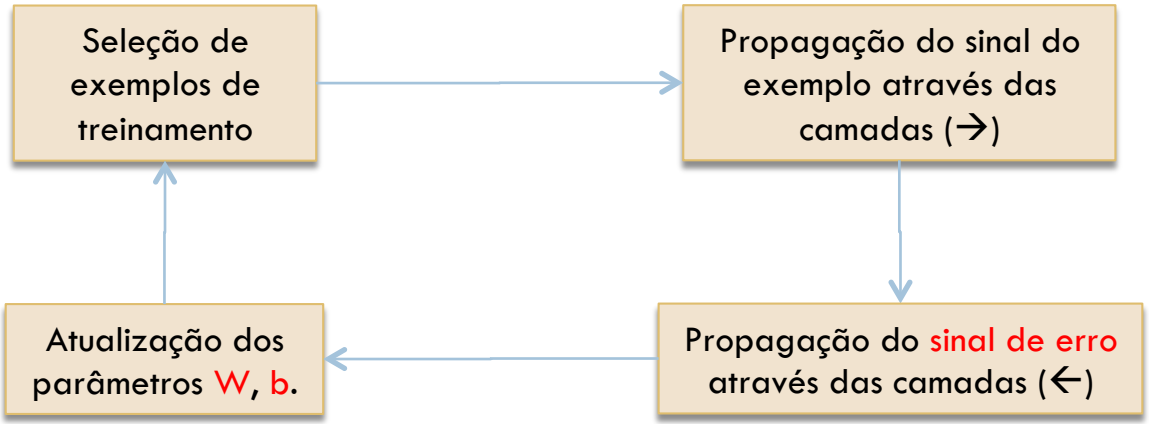
Aprendizagem de máquinas (em uma página!)



$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + \mathbf{b})$$

Treinamento em redes neurais

- Sinal de erro:
  - mede a diferença entre as previsões da rede e os valores desejados.
  - usado para gradativamente alterar os parâmetros, de modo que as previsões sejam mais precisas.



Treinamento = problema de otimização

- Dado um conjunto de treinamento da forma

$$\{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) : 1 \leq t \leq T\}$$

o treinamento corresponde a utilizar esse conjunto para ajustar os parâmetros da rede, de forma que o erro de treinamento seja minimizado.

- Dessa forma, o treinamento supervisionado é convertido em um problema de otimização.

Função de custo J

- Função de custo: erro cometido pela rede.
- Θ é conjunto de todos os pesos e bias.

$$\mathbf{J}(\theta) = \underbrace{\frac{1}{T}\sum_{t=1}^{T} J(f(\mathbf{x}^{(t)};\theta),\mathbf{y}^{(t)})}_{\text{custo dos dados}} + \underbrace{\frac{\lambda}{2T}\sum_{k}\sum_{l} W_{k,l}^{2}}_{\text{custo de regularização}}$$

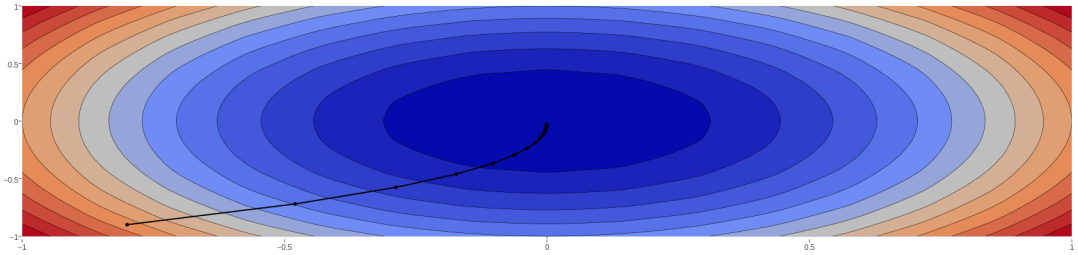
Intuição: se o vetor de pesos ficar muito grande, pequenas alterações gradativas a esse vetor não irão alterá-lo significativamente

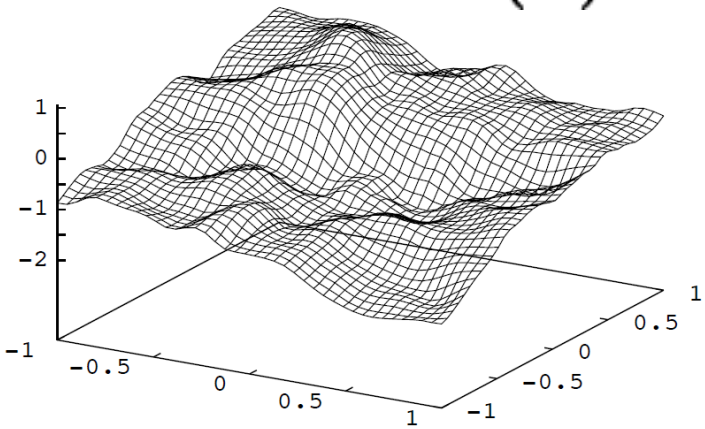
Gradiente Descendente Estocástico

(Stochastic Gradient Descent, SGD)

- Esse algoritmo permite percorrer o espaço de busca dos parâmetros de uma rede neural.

$J(\theta)$

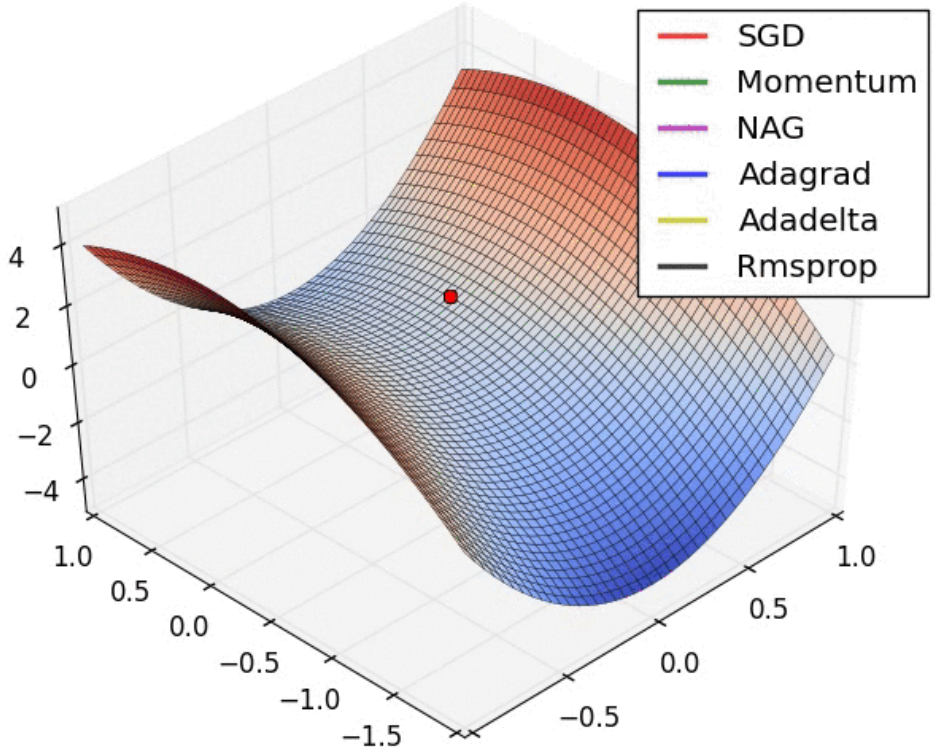




Fonte: David McKay, ITPRNN

SGD funciona pare redes profundas!

Alternativas ao SGD



Fonte: http://sebastianruder.com/optimizing-gradient-descent/

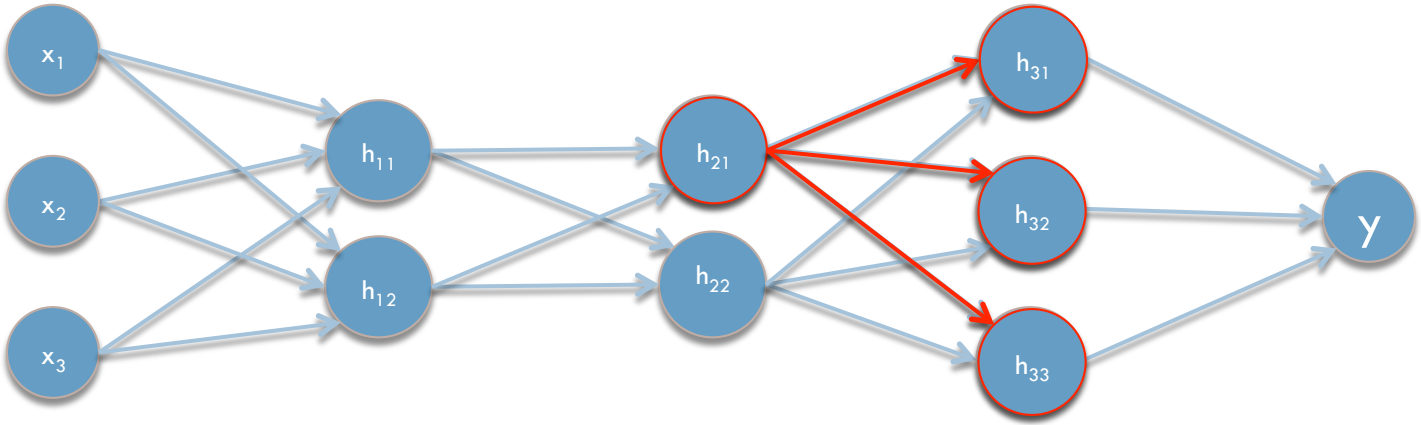
Retropropagação do erro *(error back-propagation)*

- Algoritmo redescoberto muitas vezes...
- Como **não** funciona: perturbação nos pesos...
- Como funciona: se a saída produzida para $x^{(t)}$ é diferente de $y^{(t)}$, então é necessário determinar o grau de responsabilidade de cada parâmetro da rede
  - para, em seguida, alterar esses parâmetros com o propósito de reduzir o erro produzido.

Humelhart et al. Learning representations by back-propagating errors, 1986.

Retropropagação do erro *(error back-propagation)*

- Nas unidade ocultas, usa a regra da cadeia para computar o quão rápido o erro muda quando mudamos a ativação de uma unidade oculta.



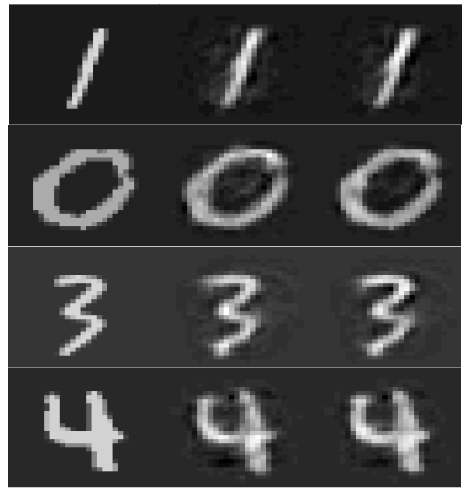
Retropropagação do erro *(error back-propagation)*

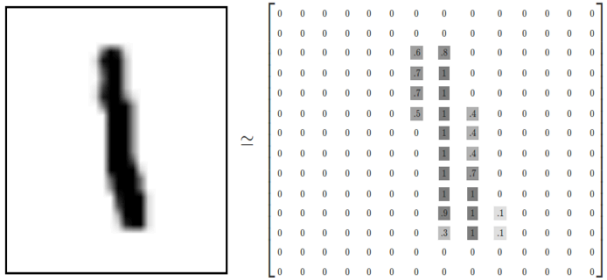
- passo 1: propagar o sinal de entrada por todas as camadas até a saída.
- passo 2: atualizar os pesos de modo que a saída produzida se aproxime da saída correta, minimizando o erro para cada neurônio de saída e para a rede como um todo.

$$\Delta w[t] = -\gamma \frac{\partial J}{\partial w[t]} = -\gamma \nabla J(w[t])$$

MNIST *(handwritten digits dataset)*

- Conjunto de dados com 70k exemplos
  - Treinamento: 60k
  - Teste: 10k
- Cada exemplo: 28 x 28 pixels
- Melhores desempenhos:
  - classificador linear : 12% error
  - (Gaussian) SVM 1.4% error
  - ConvNets <1% error





[http://yann.lecun.com/exdb/mnist/]

Redes Autocodificadoras

Redes autocodificadoras (*autoencoders*)

- Podem aprender a estrutura subjacente ao conjunto de dados de forma não supervisionada.
  - Permite o aprendizado de representações mais concisas.
  - Características identificadas são úteis para uso posterior em tarefas de aprendizado supervisionado.

Transformações realizadas

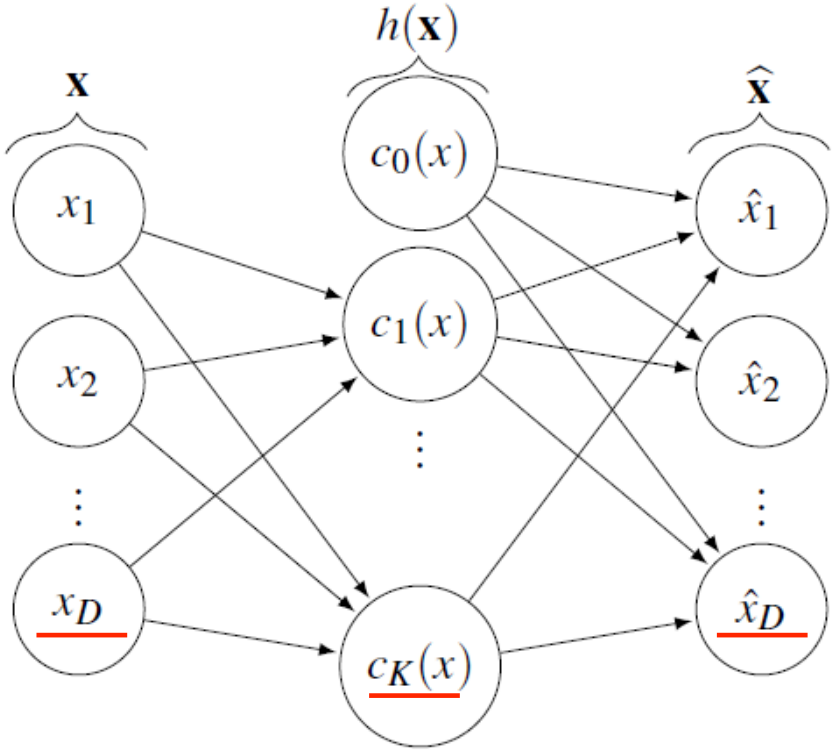
- Transformações são aplicadas na entrada de acordo com dois tipos de funções.
  - A função de extração de características (*encoder*) mapeia o conjunto de treinamento para uma **representação latente**.

$$h : \mathbb{R}^D \rightarrow \mathbb{R}^K$$

  - A função de reconstrução (*decoder*) mapeia a representação produzida por *h* de volta para o espaço original.

$$r : \mathbb{R}^K \rightarrow \mathbb{R}^D$$

Esquema da arquitetura



Treinamento

- Uma autocodificadora é treinada para reproduzir na sua saída a própria entrada.
- O objetivo do treinamento é definir parâmetros em h e r tal que o erro de reconstrução seja minimizado.
- Pode ser treinada usando backprop + SGD.
  - com o cuidado de substituir os valores-alvo desejados pela própria entrada **x**.

$$L(\mathbf{X}) = \sum_{\mathbf{x}^{(i)} \in \mathbf{X}} \ell(\mathbf{x}^{(i)}, r(h(\mathbf{x}^{(i)})))$$

Treinamento

- Possibilidade: pesos atados (*tied weights*)
  - Há pares de matrizes em posições simétricas na rede, uma é a transposta da outra!
  - Essa decisão...
    - ...resulta em menos parâmetros para otimizar;
    - ...previne soluções degeneradas.

subcompletas x supercompletas

- Se a representação latente em uma autocodificadora tem dimensão K:
  - K < D → undercomplete autoencoder;
  - K > D → overcomplete autoencoder.
- A escolha de K determina
  1. a quantidade de unidades da camada intermediária central,
  2. que tipo de informação a autocodificadora pode aprender acerca da distribuição de entrada.

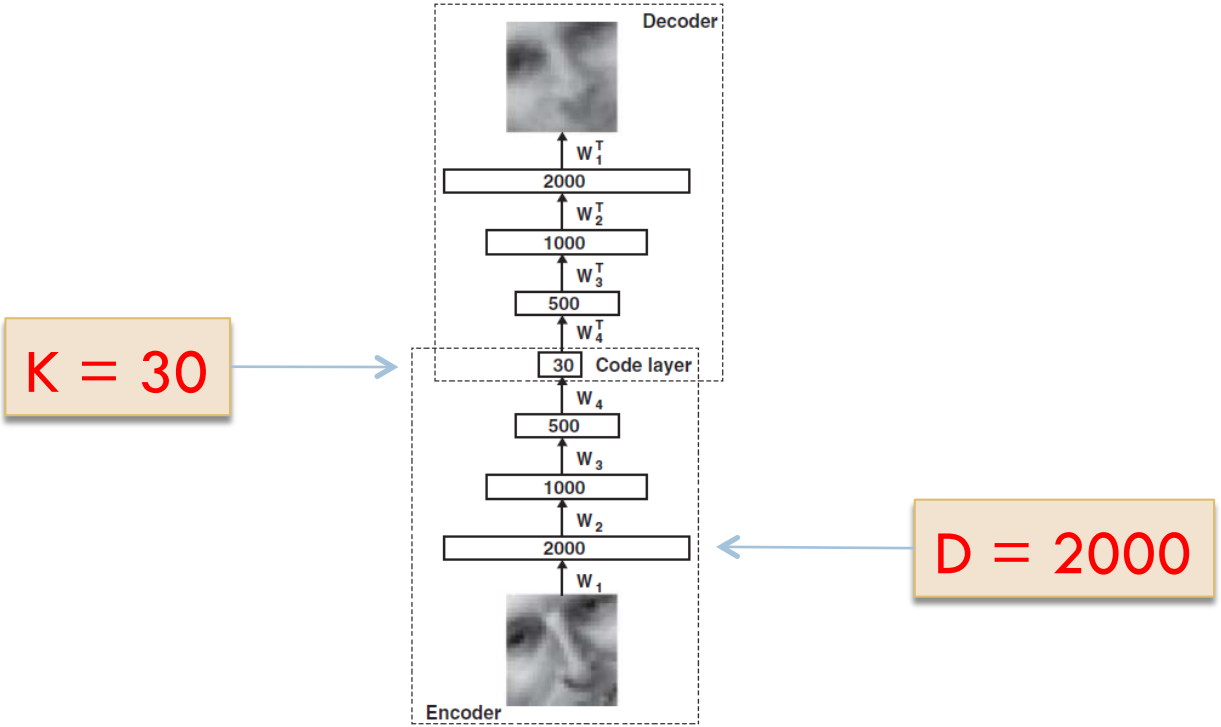
Caso K < D (*bottleneck*)

- Objetivo: aprender uma representação compacta do conjunto de dados.
- Exemplo
  - Considere
    - um conjunto de dados de treinamento T exemplos.
    - que, em cada exemplo, o esquema de codificação usado seja *um bit por estado*.
  - Nesse caso, a autocodificadora deveria codificar os exemplos por meio de uma codificação binária, com $\log_2 T$ bits por exemplo.

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

one-hot encoding

Caso K < D *(bottleneck)*



Reducing the Dimensionality of Data with Neural Networks (Science, 2006)

Caso K > D

- Objetivo: encontrar características da entrada para posteriormente serem apresentadas a um classificador linear (SVM, k-NN, etc).
- Problema potencial no treinamento: autocodificadora apenas copia os D bits da entrada para D unidades na camada intermediária.
  - aprende f(x) = x.
  - deixando de usar K-D unidades nessa camada.

Autocodificadora com filtragem de ruído

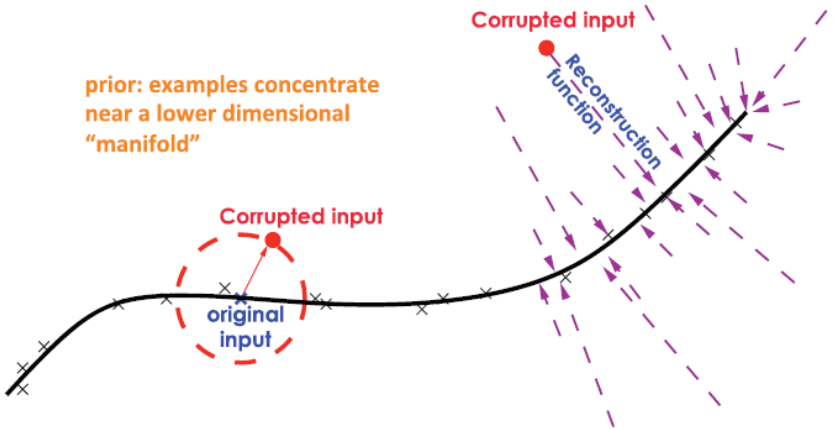
(denoising autoencoder)

- Ideia básica: fazer com que a representação aprendida seja robusta a ruídos nos dados de entrada.
- Aplicar um processo probabilístico em cada exemplo de treinamento **x** antes de apresentá-lo à rede.
- Alternativas
  a) com probabilidade p, atribuir zero a cada componente de **x**.
  b) perturbar cada componente de **x** por meio de um ruído gaussiano aditivo.

Representation Learning - A Review and New Perspectives (2013)

Autocodificadora com filtragem de ruído

(denoising autoencoder)



Representation Learning - A Review and New Perspectives (2013)

Autocodificadora contrativa

(contractive autoencoder)

- Ideia básica: adicionar uma penalização à função de perda para penalizar representações indesejadas.

$$-\sum_{k=1}^{d}[x_k \log z_k + (1-x_k)\log(1-z_k)] + \left\|\frac{\partial h(x)}{\partial x}\right\|^2$$

penalização

"mantenha boas representações" + "descarte todas as representações" = "mantenha apenas boas representações"

Contractive Auto-Encoders: Explicit Invariance during Feature Extraction (2011)

Autocodificadora esparsa

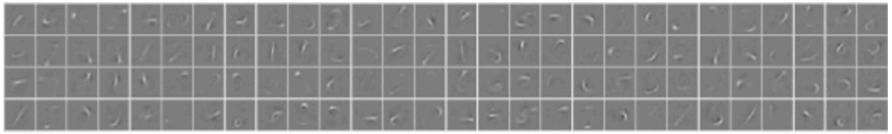
(sparse autoenconder)

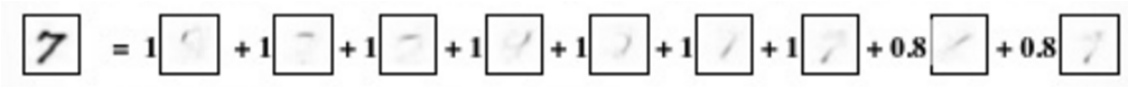
- Objetivo: fazer com que apenas uma pequena quantidade de unidades da camada oculta seja ativada para cada padrão de entrada.
- A esparsidade pode ser obtida
  - por meio de termos adicionais na função de perda durante o treinamento,
  - mantendo apenas as k unidades mais ativas e tornando todas as demais unidades iguais a zero manualmente.

Autocodificadora esparsa

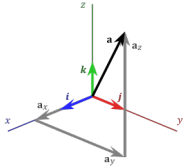
(sparse autoenconder)

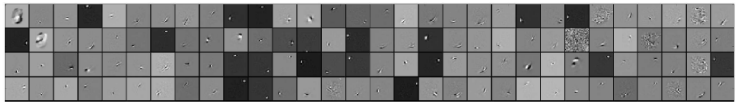
- Motivação biológica: visual córtex (V1)



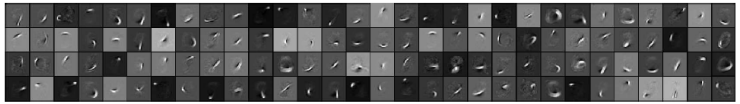


Exemplo: imagens de 28x28 pixels podem ser representadas por uma qtd. pequena de **códigos** a partir de uma **base.**

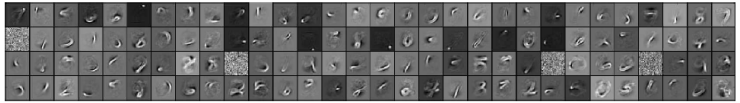




(a) $k = 70$



(b) $k = 40$

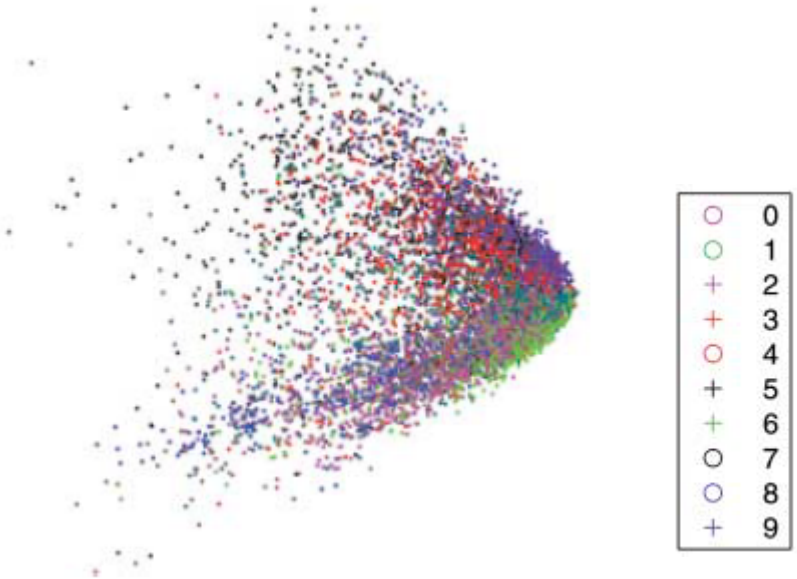


(c) $k = 25$

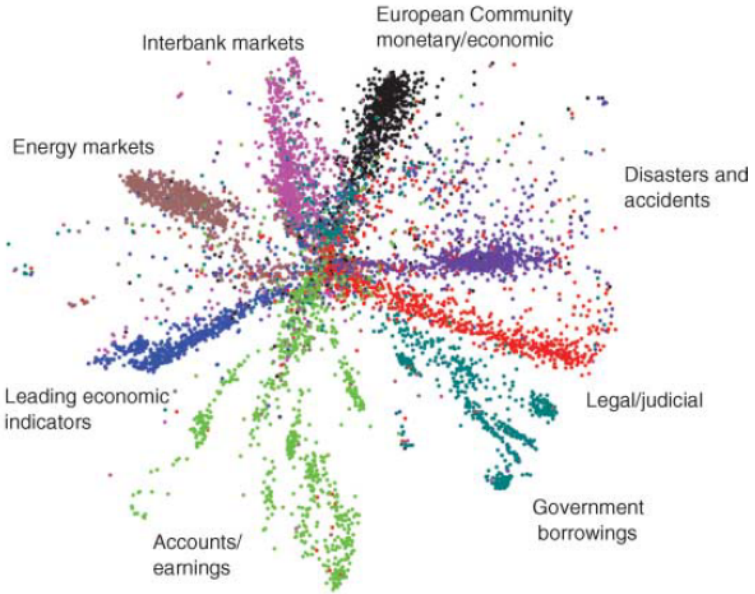
(d) $k = 10$

Aplicações (I) - redução de dimensionalidade

PCA
(k=2)



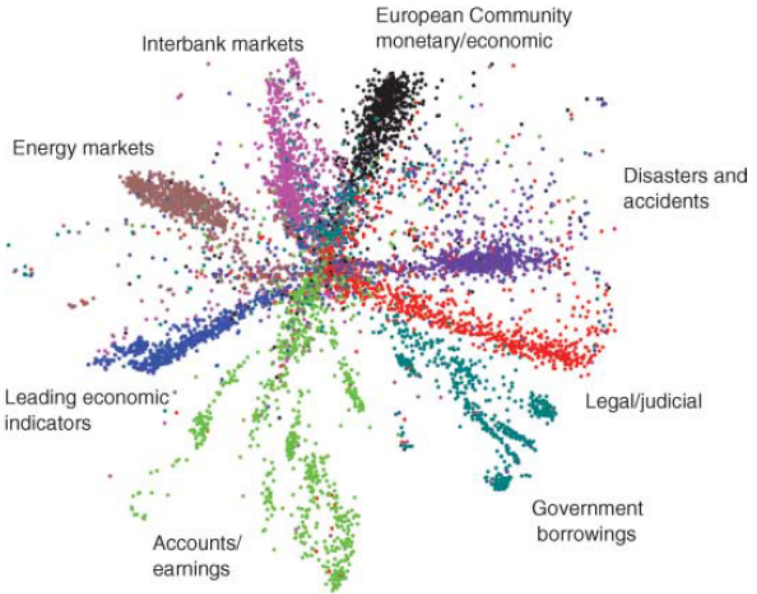
Autocodificadora
(2000-500-250-125-2)

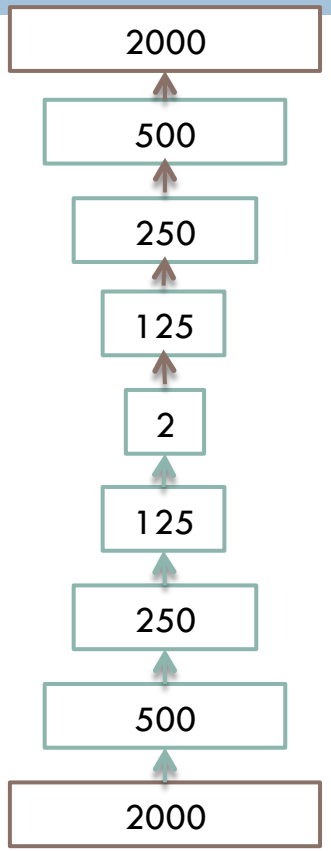


Reducing the Dimensionality of Data with Neural Networks (Science, 2006)

Aplicações (I) - redução de dimensionalidade

Autocodificadora
(2000-500-250-125-2)

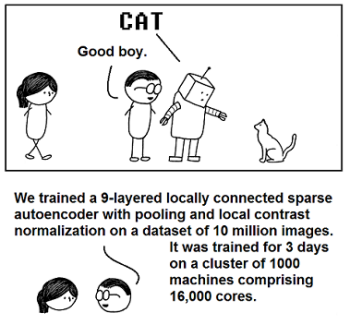


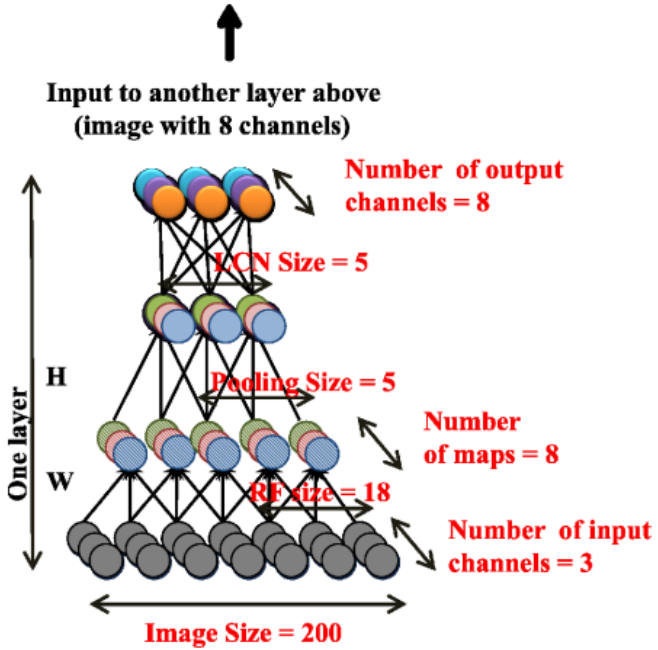


Reducing the Dimensionality of Data with Neural Networks (Science, 2006)

Aplicações (II) – aprendizado de conceitos

- 10M vídeos do YouTube
  - 1 frame por vídeo (200x200)
- A rede aprendeu os conceitos de face humana e de face de gatos.





Le et al, Building high-level features using large-scale unsupervised learning, 2012

Redes Convolucionais

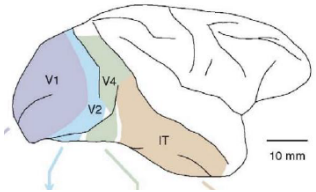
Experimento de Hubel e Wiesel

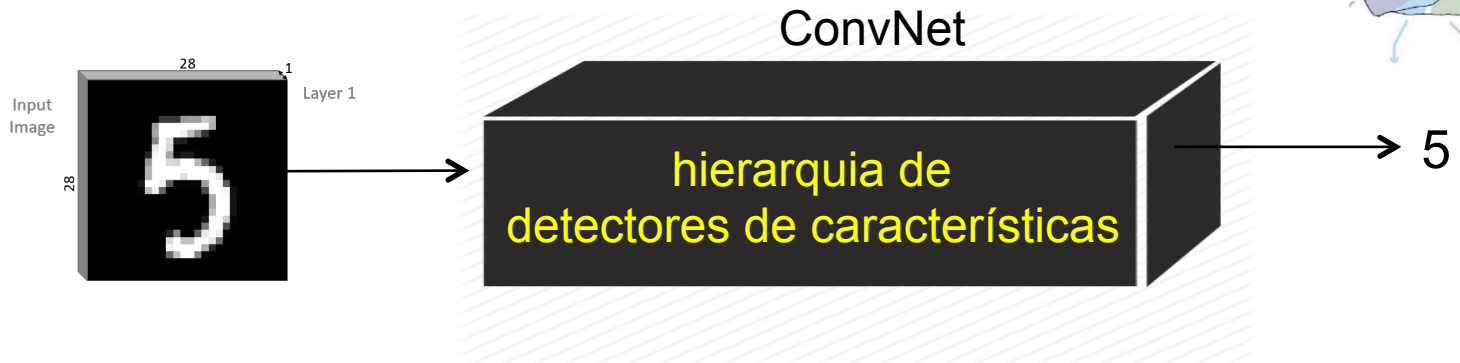
Uma descoberta fortuita!

Hubel, D. and Wiesel, T. (1968). Receptive fields and functional architecture of monkey striate cortex. Journal of Physiology.

Redes Convolucionais

- Se inspiram no funcionamento do córtex visual.
- Sua arquitetura é adaptada para explorar a correlação espacial existente em imagens naturais.
- São redes de propagação adiante.





ImageNet: taxas de erro (2010-2014)

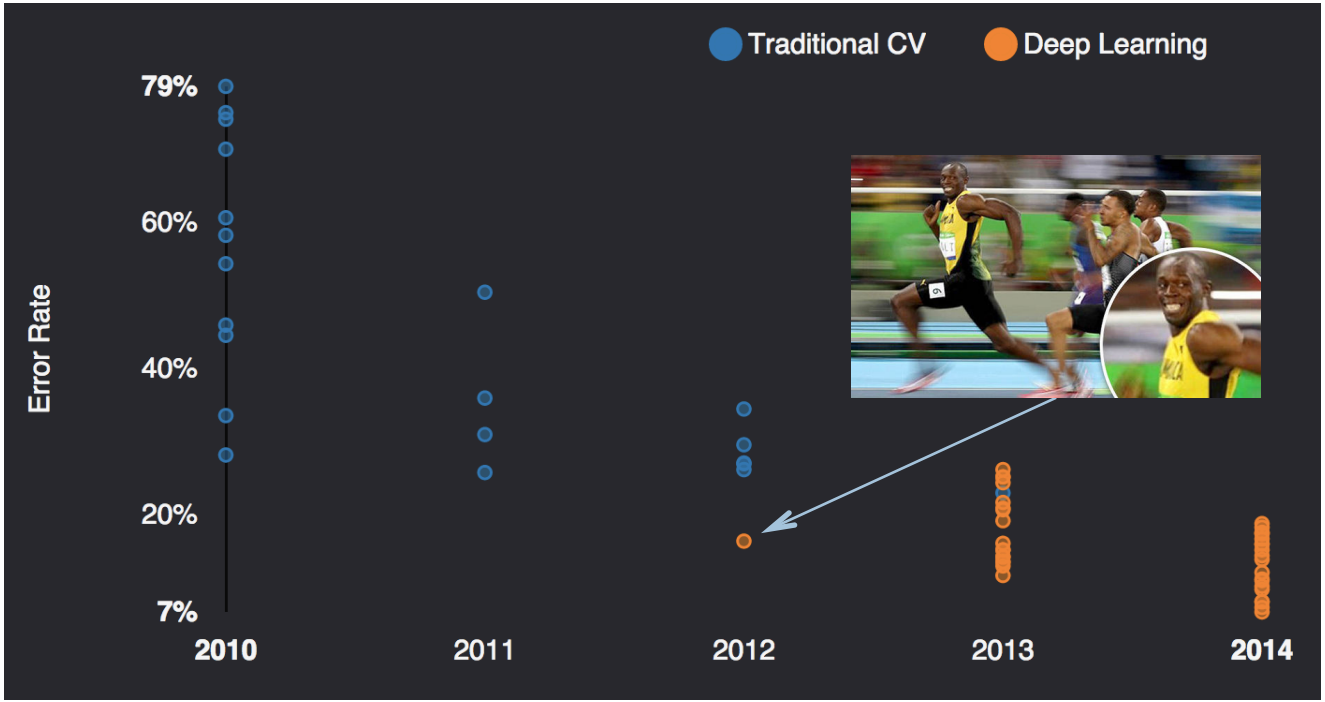


Gráfico reproduzido do material de Mathew Zeiler (Clarifai)

Conceitos e Operações

- campos receptivos locais (local receptive fields),
- compartilhamento de pesos (shared weights),
- convolução (convolution),
- subamostragem (subsampling, pooling).

Redes completamente conectadas

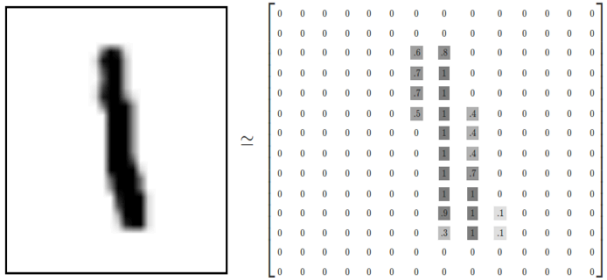
- Suponha
  - uma rede completamente conectada com uma camada oculta para o MNIST (imgs 28x28);
  - essa única camada oculta tenha o dobro de unidades da camada de entrada.
- Nesse caso, teríamos da ordem de 10^6 parâmetros para ajustar durante o treinamento!

Redes completamente conectadas

- resulta em uma quantidade grande de parâmetros → risco de sobreajuste (*overfitting*).
- inadequadas a imagens de alta resolução → potencial de sobreajuste (*overfitting*).
- tempo para computar as pré-ativações.

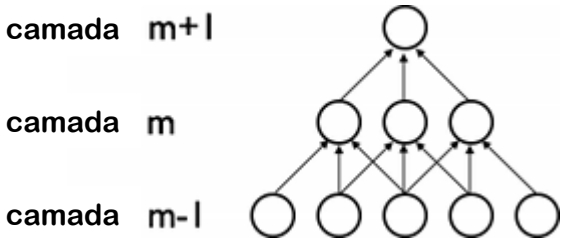
Redes completamente conectadas

- não considera a estrutura espacial existente em alguns domínios
  - e.g., imagens → pixels próximos e os localizados em regiões distantes tratados indistintamente.
- a própria rede teria que detectar as dependências existentes na estrutura espacial da distribuição subjacente às imagens de entrada.



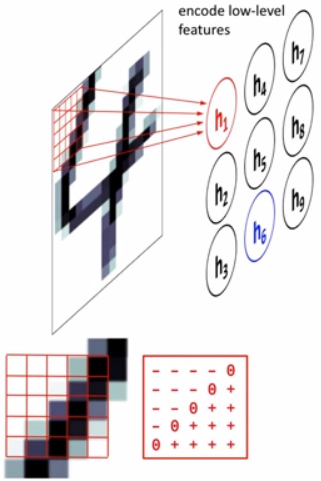
Campos receptivos locais

- Uma CNN utiliza conectividade local.
- e.g., cada unidade de $L^{(1)}$ está conectada a um subconjunto de unidades da camada $L^{(0)}$.
  - subconjunto → campo receptivo local dessa unidade.



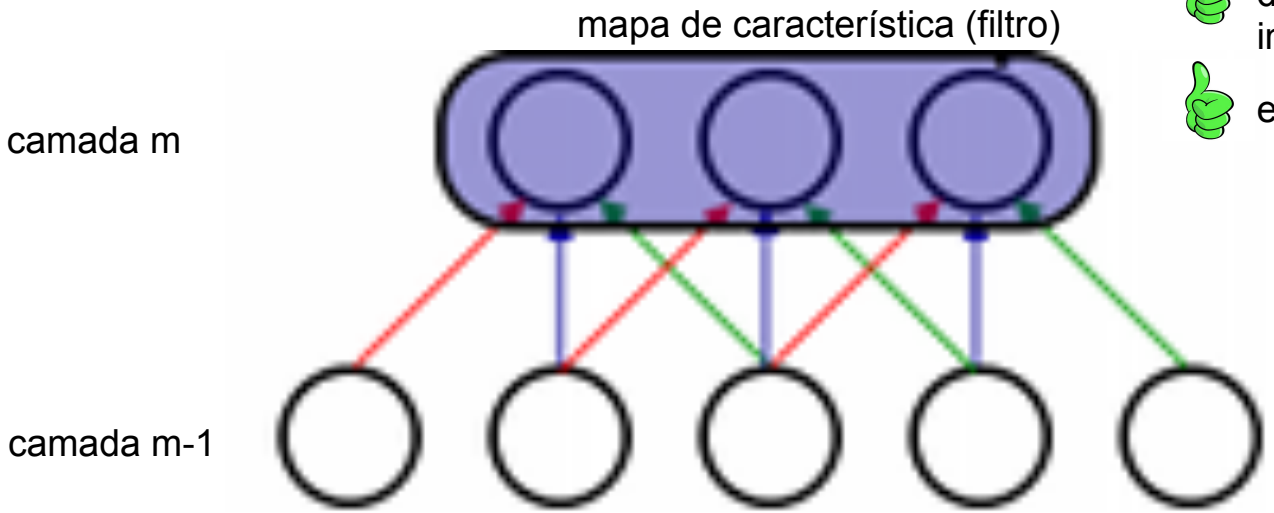
Campos receptivos locais

- Por meio de seu campo receptivo local, cada unidade de $L^{(1)}$ pode detectar características visuais elementares
  - (e.g., arestas orientadas, extremidades, cantos).
- Que podem então ser combinadas por camadas subsequentes para detectar características visuais mais complexas.
  - (e.g., olhos, bicos, rodas, etc).



Créditos: Victor Lavrenko

Compartilhamento de pesos

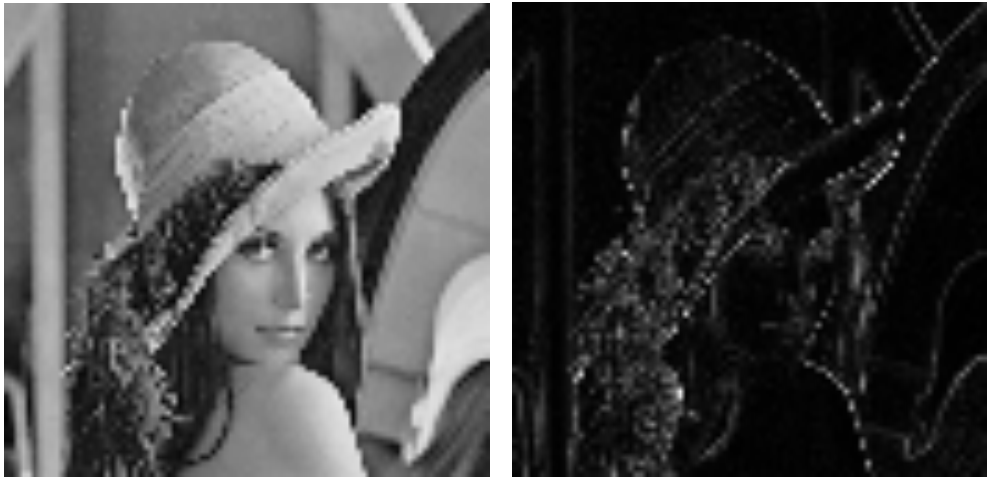


- detecção de característica independente de sua posição
- eficiência do aprendizado

Figura adaptada de http://deeplearning.net/tutorial/lenet.html

Convolução

- Cada unidade em um filtro realiza uma convolução sobre seu respectivo campo receptivo.



Camada de convolução

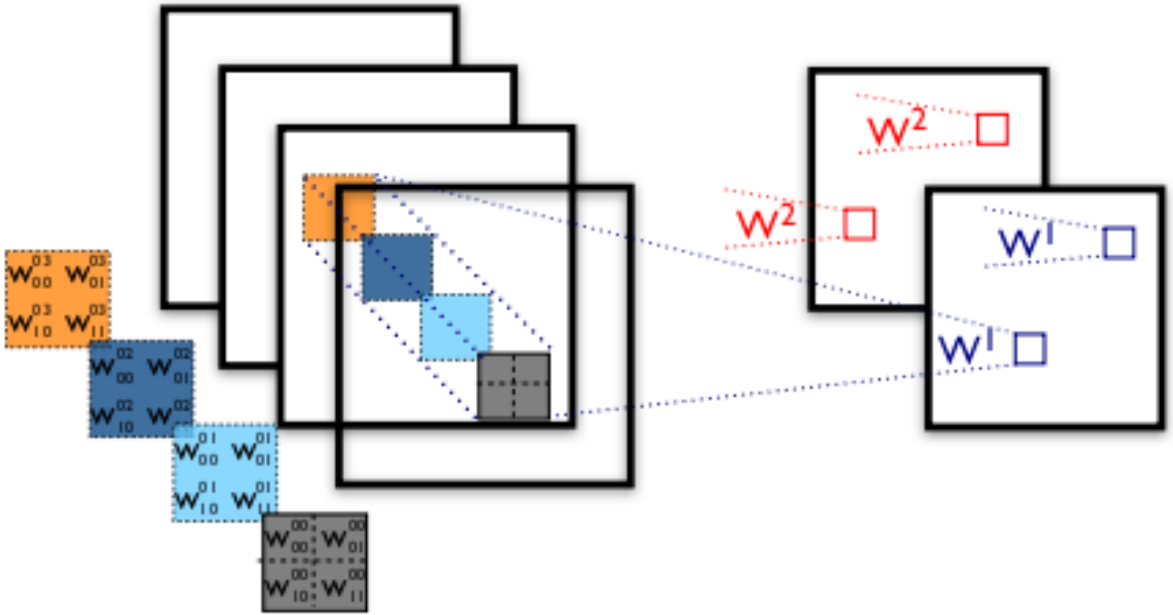


Figura adaptada de http://deeplearning.net/tutorial/lenet.html

Subamostragem (*downsampling*)

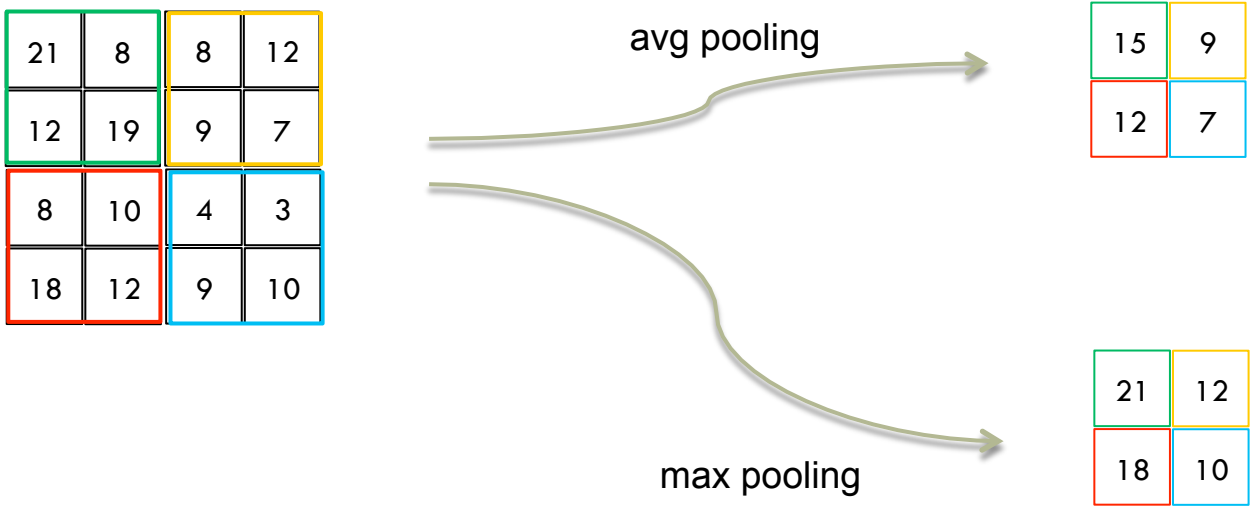
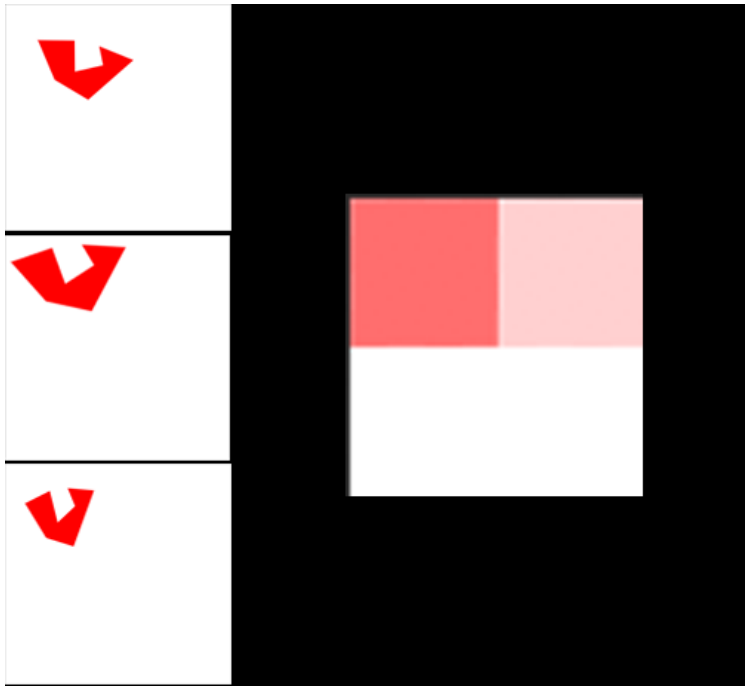


Imagem reproduzida do material disponibilizado na disciplina CS231n (http://cs231n.stanford.edu/)

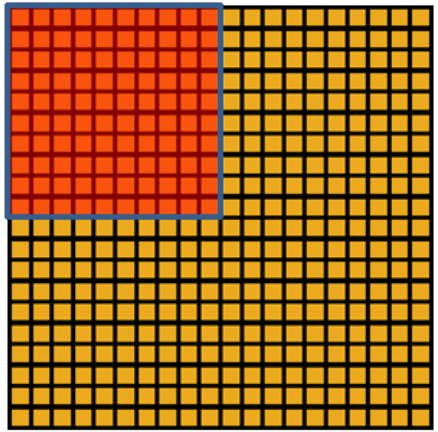
Subamostragem (*downsampling*)

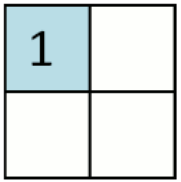


invariância com relação
a pequenas translações

Imagem reproduzida de https://www.quora.com/How-exactly-does-max-pooling-create-translation-invariance

Camada de subamostragem





camada m-1 camada m

Imagem reproduzida de http://ufldl.stanford.edu/wiki/index.php/Pooling

Normalização de contraste

(Local Constrast Normalization, LCN)

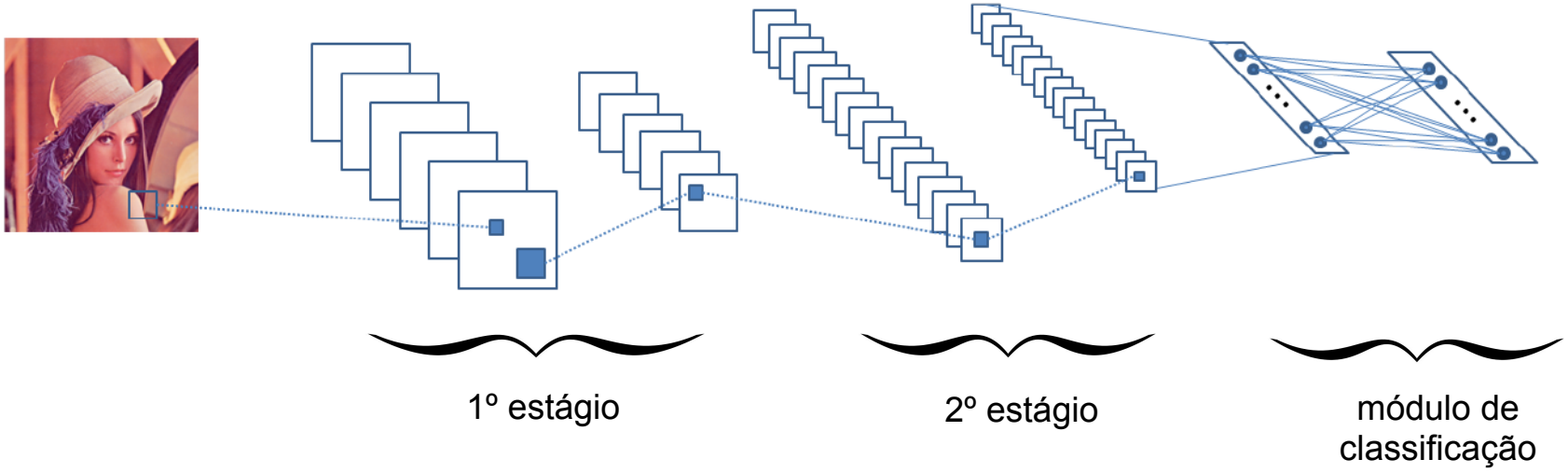
- Uma camada LCN normaliza o contraste de uma imagem de forma não linear.
  - aplica a normalização sobre regiões locais da imagem, considerando cada pixel por vez.
- A normalização pode corresponder a subtrair a média da vizinhança de um pixel particular e dividir pela variância dos valores de pixel dessa vizinhança.

invariância de brilho

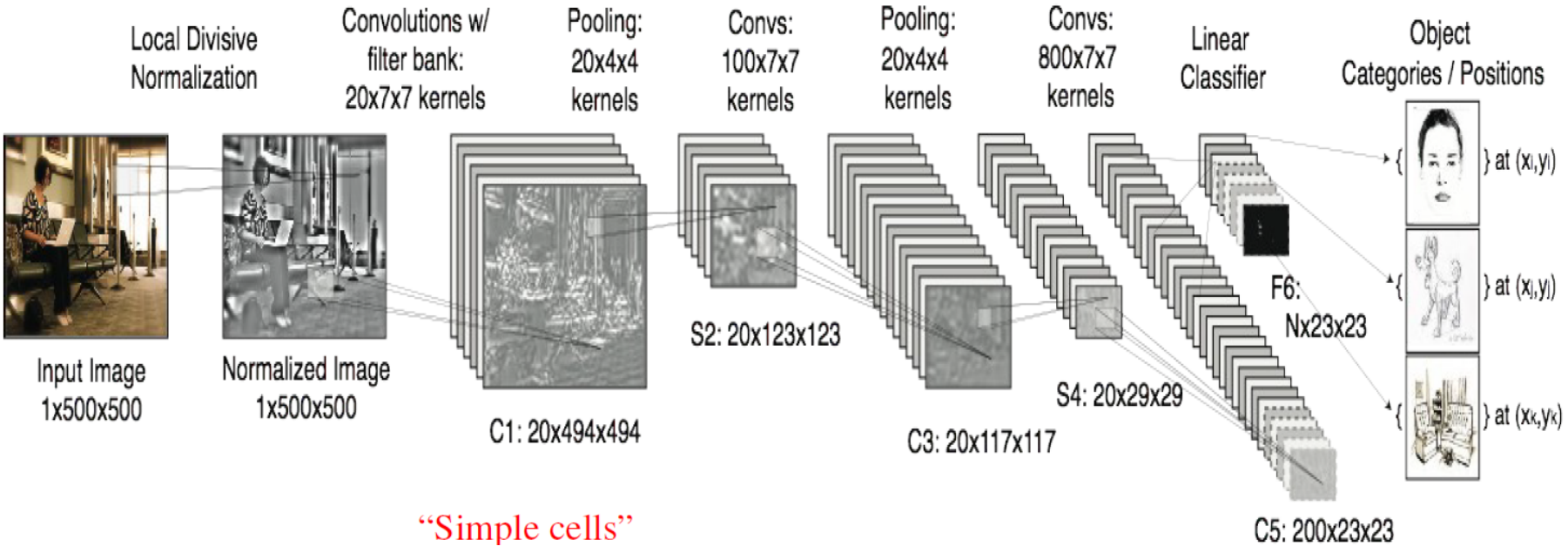
Arquitetura

- Em uma convnet, encontramos um ou mais estágios, cada qual composto por camadas:
  - de convolução,
  - de subamostragem,
  - de normalização de contraste,
  - completamente conectadas (módulo de classificação).

Arquitetura



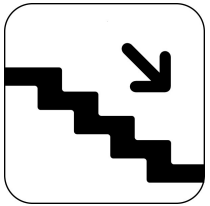
Aplicações (I) – LeNet, 1980's



Yan Le Cun et al, Convolutional Networks and Applications in Vision (2010)

Aplicações (II) – AlexNet, 2012

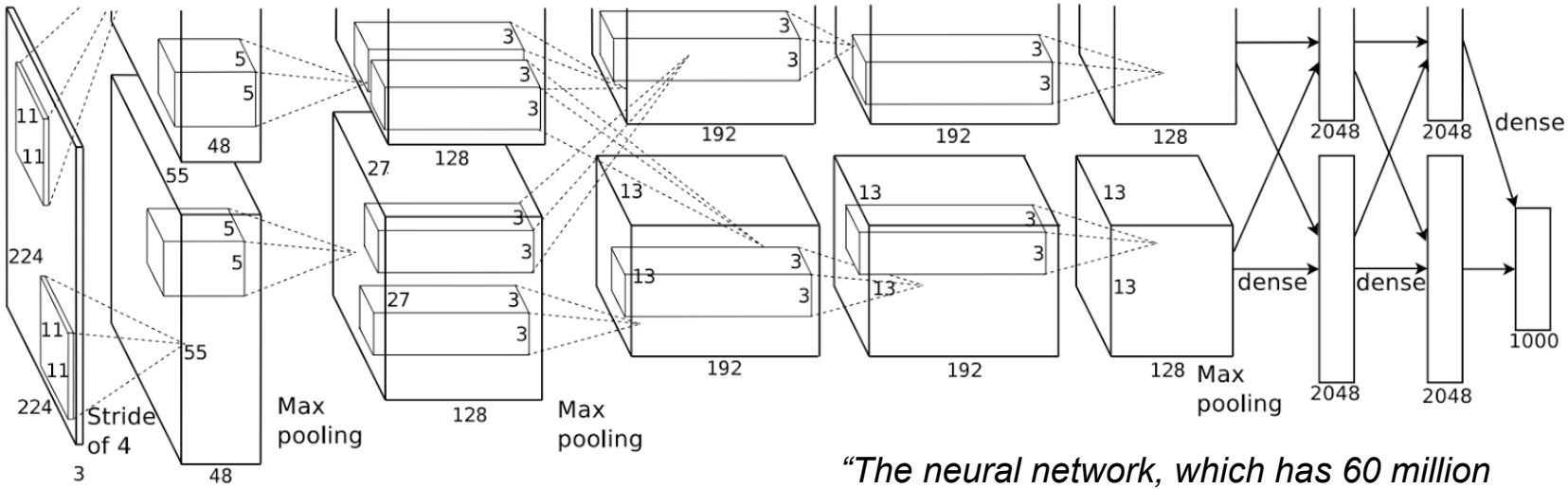
- Competição ILSVRC (*ImageNet Challenge*)
  - ~1,2M de imagens de alta resolução para treinamento;
  - ~1000 imagens por classe; 1000 classes!
- Ganhou a edição 2012
  - Até então, soluções incluiam características produzidas manualmente, estudadas e melhoradas por décadas.



9%

ImageNet Classification with D
https://papers.nips.cc/paper/4824-im
by A Krizhevsky - 2012 - Cited by 6933 -

Aplicações (II) – AlexNet, 2012

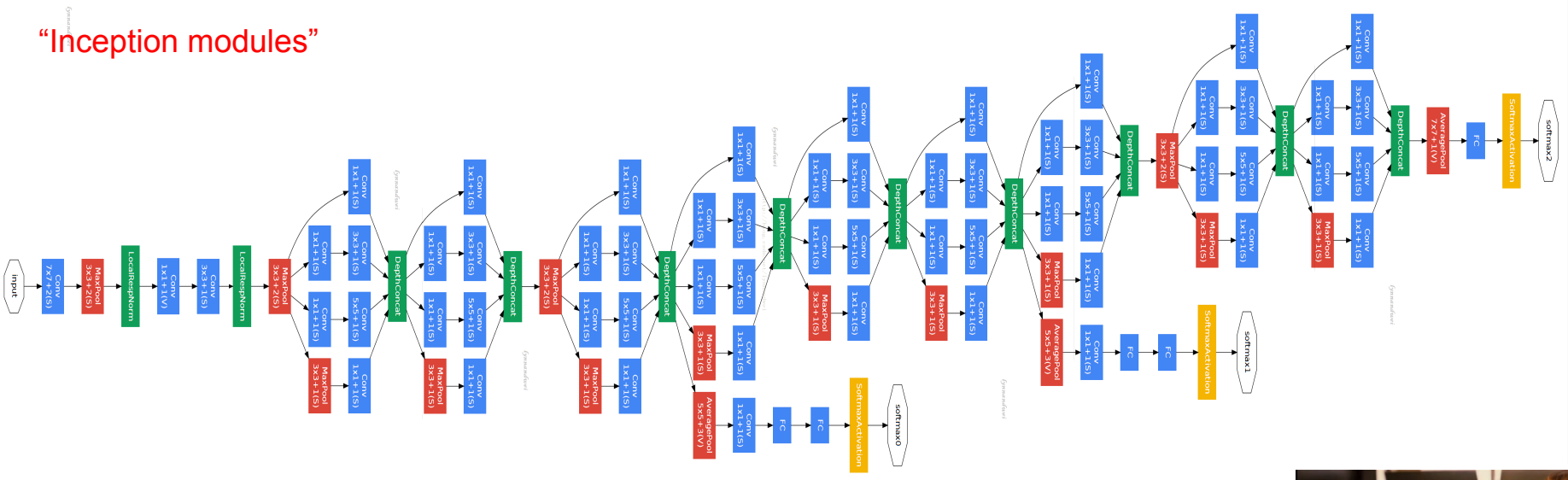


"The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax."

ImageNet Classification with Deep Convolutional Neural Networks (2012)

Aplicações (III) – GoogLeNet, 2015

"Inception modules"

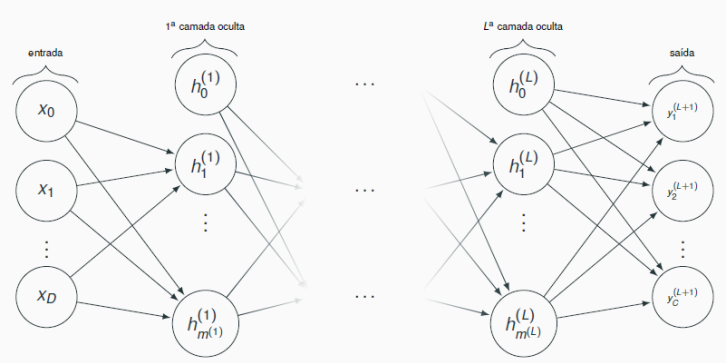


Going Deeper with Convolutions (2015)

Redes Recorrentes

Redes de propagação adiante

- Hipóteses sobre os dados
  - são IID
  - têm tamanho fixo

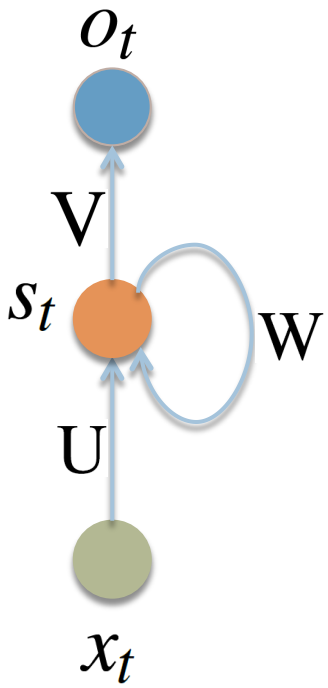


De que forma considerar dados em que há dependências de curto/longo prazo?

Redes recorrentes

- Aplicáveis a dados sequenciais
  - Texto
  - Vídeo
  - Áudio
  - ...

Unidade de recorrência



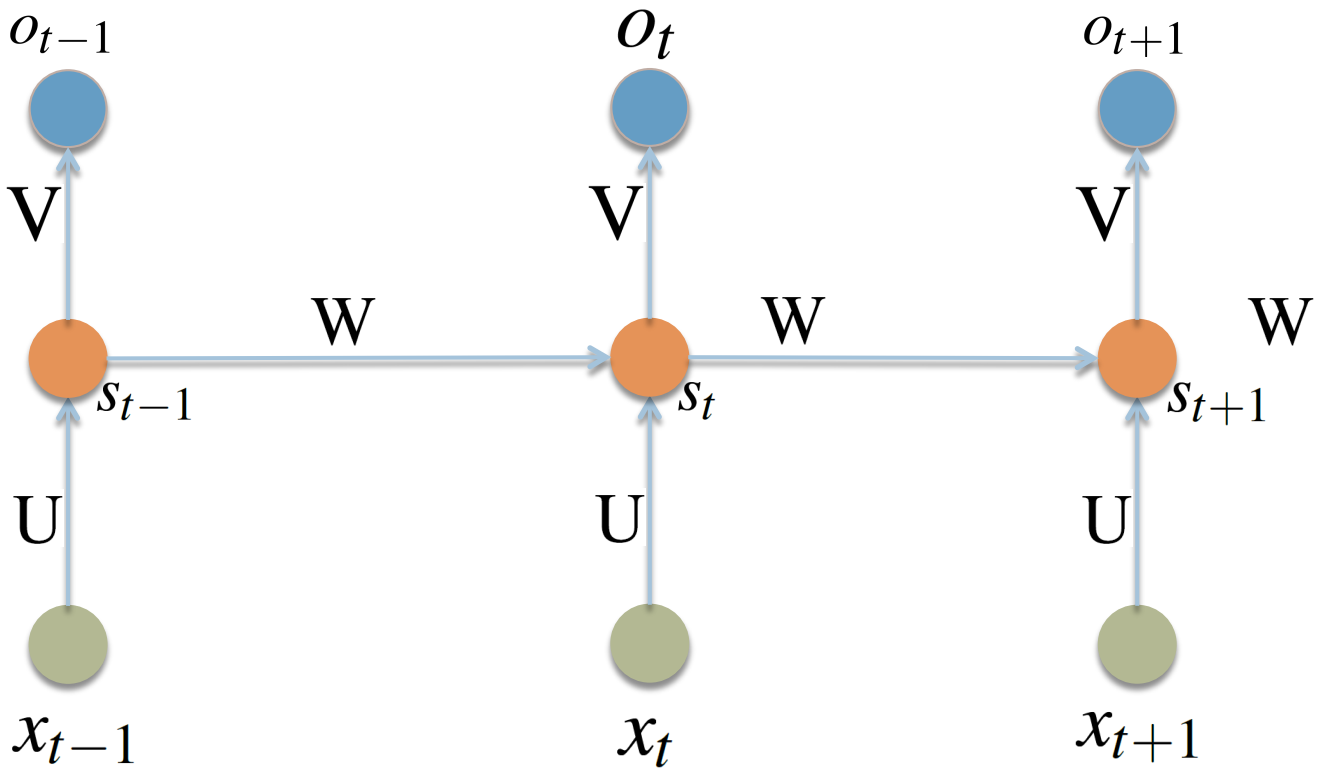
$$s_t = \sigma(U x_t + W s_{t-1} + b_s)$$

$$o_t = \text{softmax}(V s_t + b_o)$$

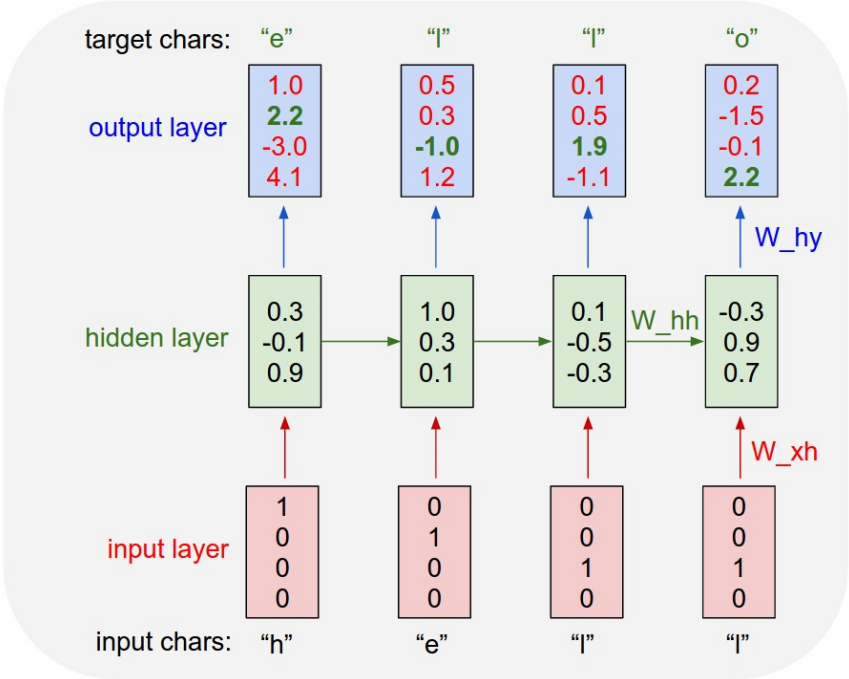
Desdobramento no tempo *(unfolding in time)*

- Situação em que uma RNN recebe uma sequência de sinais da entrada, um em cada passo de tempo.
- Uma RNN que recebe uma sequência de n vetores de entrada pode ser vista como uma rede alimentada adiante de n camadas.

Desdobramento no tempo *(unfolding in time)*



Exemplo



Créditos: Andrej Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks

Retropropagação através do tempo

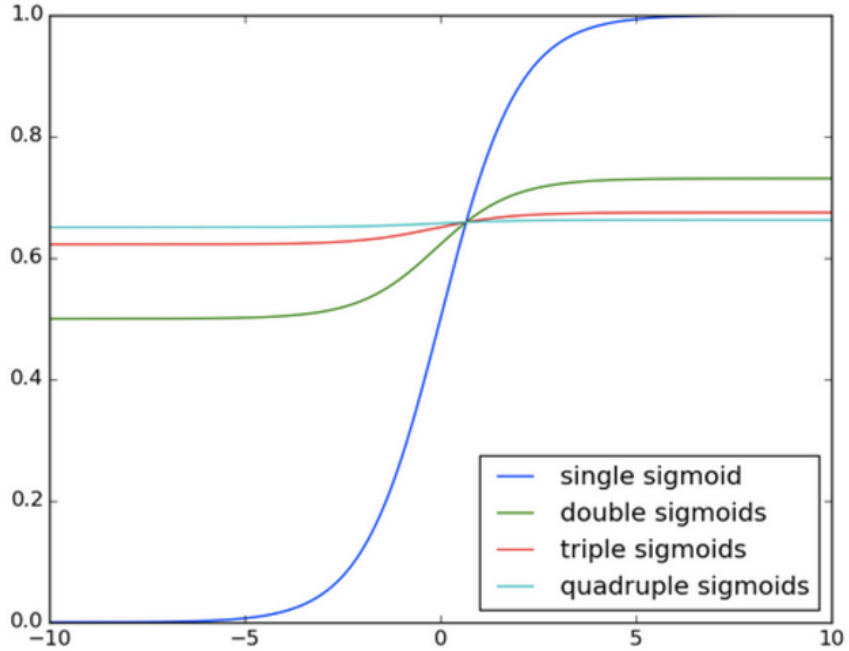
Backpropagation Through Time (BPTT)

- Restrição no cálculo dos gradientes: igualdade das matrizes de pesos em cada camada oculta.
- Exemplo:
  - w_1 e w_2 pesos de conexões correspondentes em duas camadas (i.e. instantes de tempo) ocultas distintas.
  - o BPTT calcula os gradientes relativos a w_1 e w_2 e usa a média dessas quantidades para atualizar w_1 e w_2.

$$\frac{\partial J}{\partial w_2} \qquad \frac{\partial J}{\partial w_1}$$

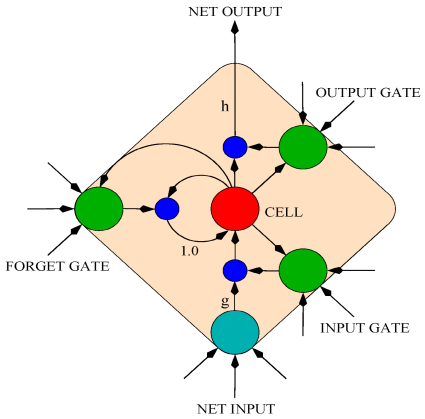
Dissipação dos gradientes

(vanishing gradients)

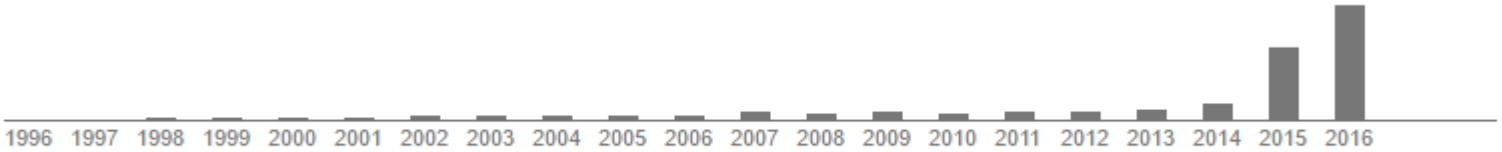


Bengio, et al., Learning Long-Term dependencies with Gradient Descent is Difficult, IEEE Transaction on Neural Networks, Vol. 5, No.2, 1994.
Pascanu, et al., On the difficulty of training Recurrent Neural Networks.

LSTMs e GRUs



Juergen Schmidhuber



https://scholar.google.de

Aplicações (I) – geração de texto

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

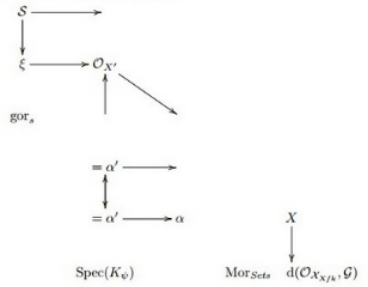
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_i}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\mathbb{R}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Sutskever, Martens, and Hinton: "Generating Text with Recurrent Neural Networks.

Aplicações (II) – legendas automáticas

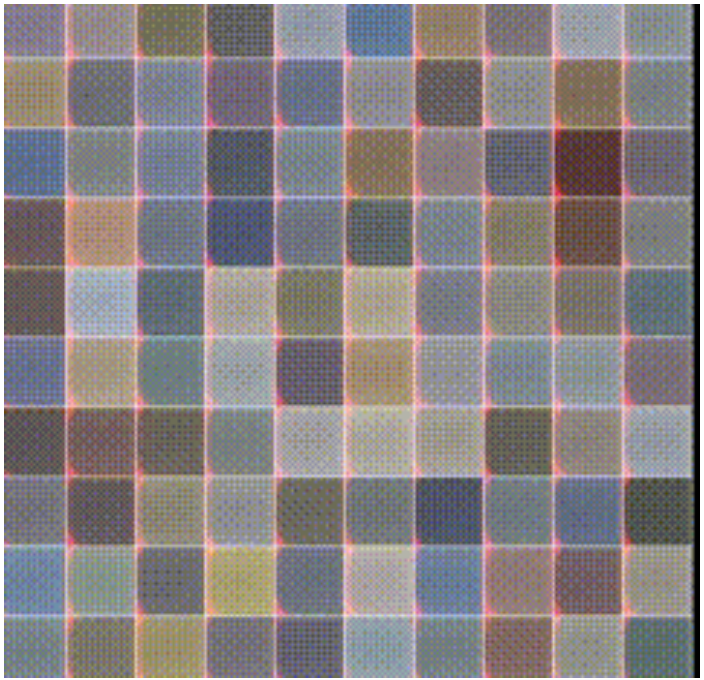
A close up of a hot dog on a bun.

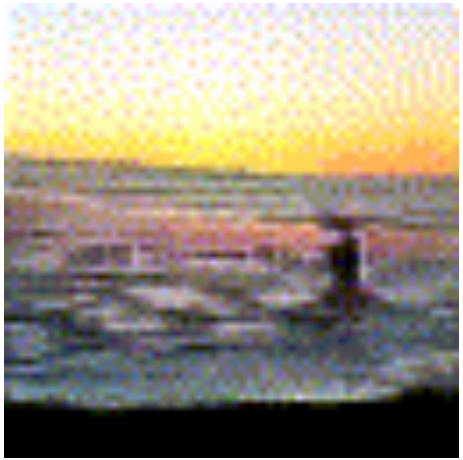
A bath room with a toilet and a bath tub.

A vase filled with flower sitting on a table.

Long-term Recurrent Convolutional Networks for Visual Recognition and Description, 2016.

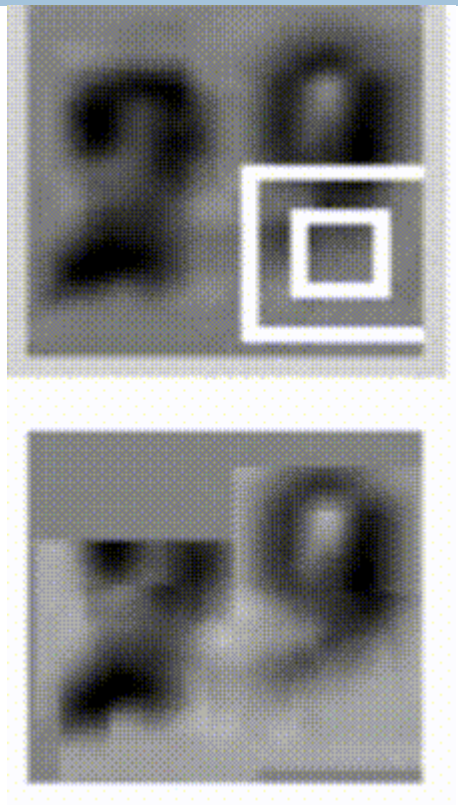
Aplicações (III) – geração de imagens





DRAW - A Recurrent Neural Network For Image Generation (2015)

Aplicações (IV) – modelos de atenção



Multiple Object Recognition with Visual Attention (2015)

Técnicas para Treinamento de Redes Profundas

Técnicas

- Pré-treinamento não supervisionado
- Uso de ReLU
- Desligamento (*dropout*)
- Normalização em lote

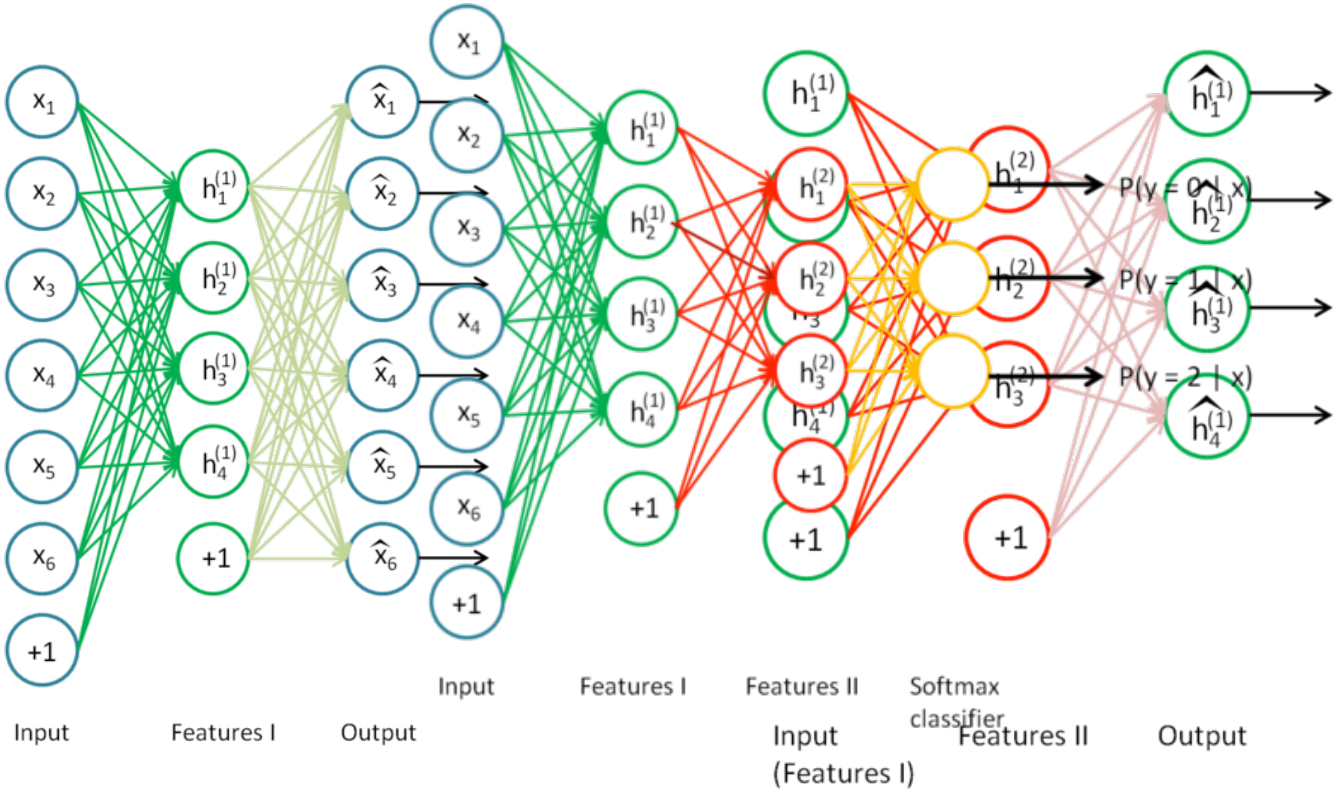
Pré-treinamento não supervisionado (*unsupervised pre-training*)

- Objetivo: iniciar os pesos da rede.
- Passos:
  - Pré-treinar uma camada por vez (*layer-wise*);
    - autocodificadoras!
  - Empilhar cada camada sobre as demais;
  - Refinar (e.g. com SGD + backprop)

Reducing the Dimensionality of Data with Neural Networks (Science, 2006)

Pré-treinamento não supervisionado

(*unsupervised pre-training*)



Pré-treinamento não supervisionado (*unsupervised pre-training*)

- Essa técnica foi um dos fatores responsáveis pelo ressurgimento do interesse por redes neurais em 2006.
- Tem sido substituída por técnicas propostas mais recentemente (e.g., dropout, ReLUs, etc).

Reducing the Dimensionality of Data with Neural Networks (Science, 2006)

Unidades Lineares Retificadas

(rectified linear units, ReLU)

- Durante anos, foi um consenso usar sigmoides!
- Propriedade indesejada: sua saturação prejudica o cálculo dos gradientes durante o SGD!
- Atualmente, o consenso é utilizar ReLUs
  - são mais eficientes para o treinamento
  - apresentam menos problemas de saturação

Rectified linear units improve restricted boltzmann machines, 2010

Unidades Lineares Retificadas

(rectified linear units, ReLU)

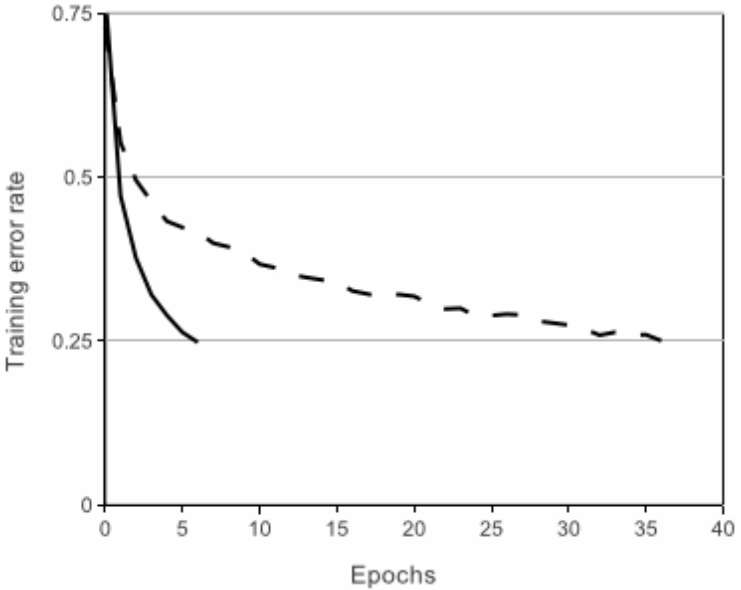
- 2011: o grupo de Yoshua Bengio demonstrou que ReLUs permitem o treinamento supervisionado sem a usar pré-treinamento não supervisionado.
- 2015: uma rede convolucional treinada no ImageNet com ReLU pela primeira vez atingiu precisão super-humana.

Deep sparse rectifier neural networks, 2011
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015

Unidades Lineares Retificadas

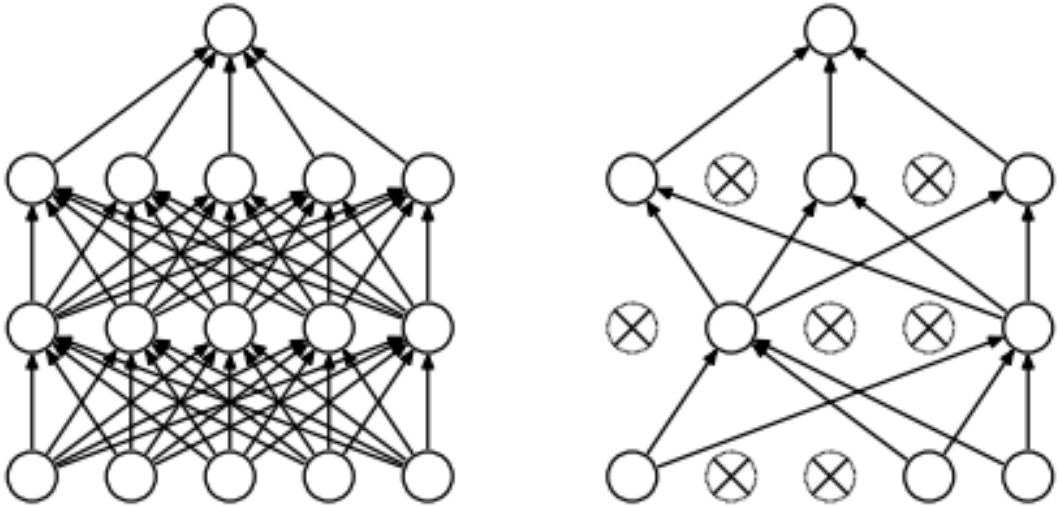
(rectified linear units, ReLU)

- Convnet de 4 camadas com ReLUs.
  - CIFAR-10 (60K, 32x32 imgs)
  - Alcança o mesmo patamar de erro 6 vezes mais rápido que tanh.



ImageNet Classification with Deep Convolutional Neural Networks, 2012.

Desligamento (*dropout*)



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Dropout - A Simple Way to Prevent Neural Networks from Overfitting (JMLR, 2014)

Desligamento (*dropout*)

- Essa técnica pode ser interpretado como uma forma de acréscimo de dados (*data augmentation*).
  - zerar a ativação de algumas unidades é equivalente a fornecer um exemplo moldado para produzir ativações iguais a zero para aquelas unidades.
  - cada exemplo moldado é muito provavelmente diferente do exemplo original correspondente.
  - cada máscara diferente corresponde a um exemplo moldado de forma diferente.

Normalização em Lote (Batch Normalization)

- Consiste em normalizar os dados fornecidos a cada camada oculta.
- Experimentos:
  - produz um efeito regularizador no treinamento, em alguns casos eliminando a necessidade de aplicar o desligamento.
  - aceleração do tempo de treinamento: diminuição de 14 vezes na quantidade de épocas necessárias.

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICML, 2015

Considerações Finais

Fatores para o sucesso

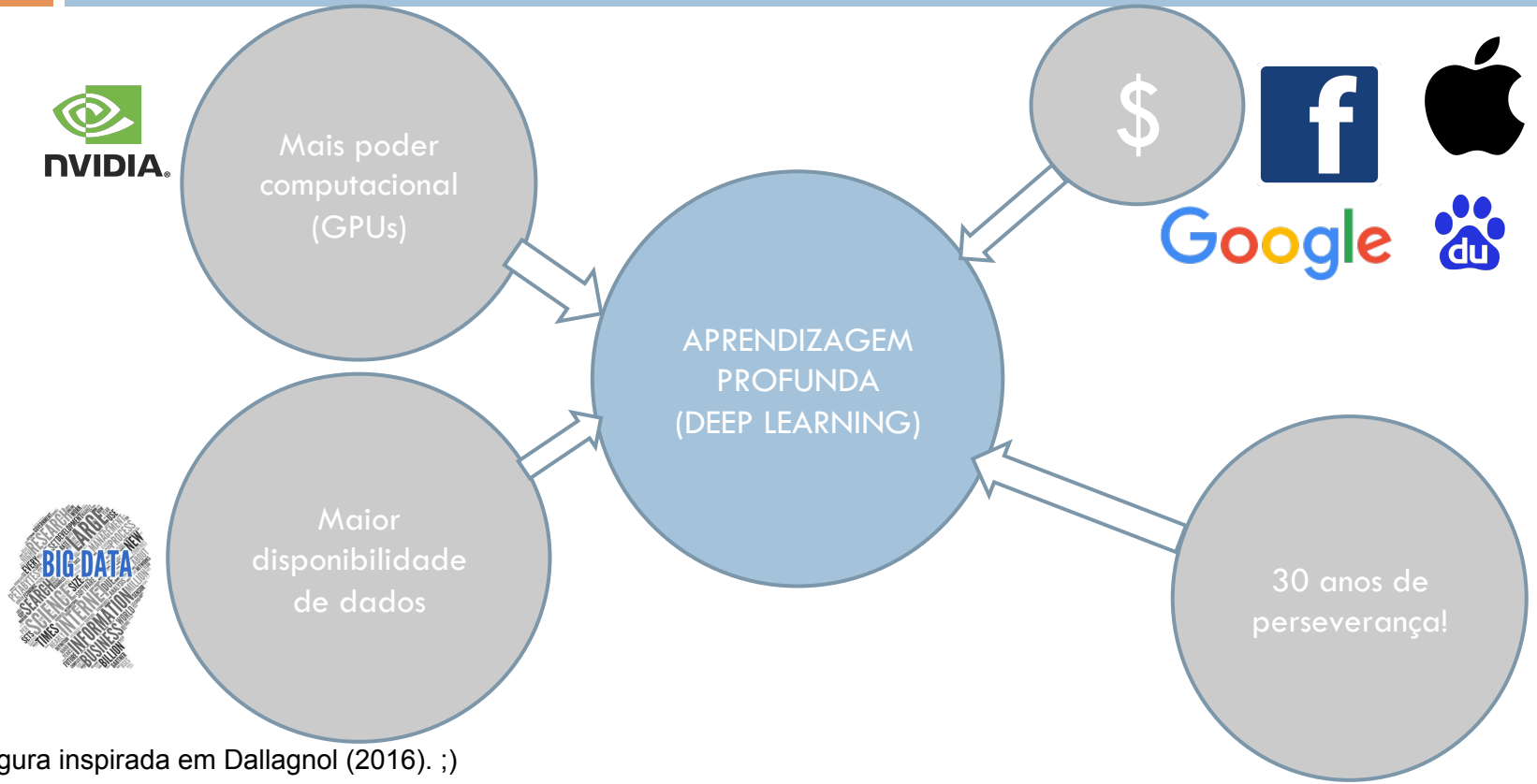
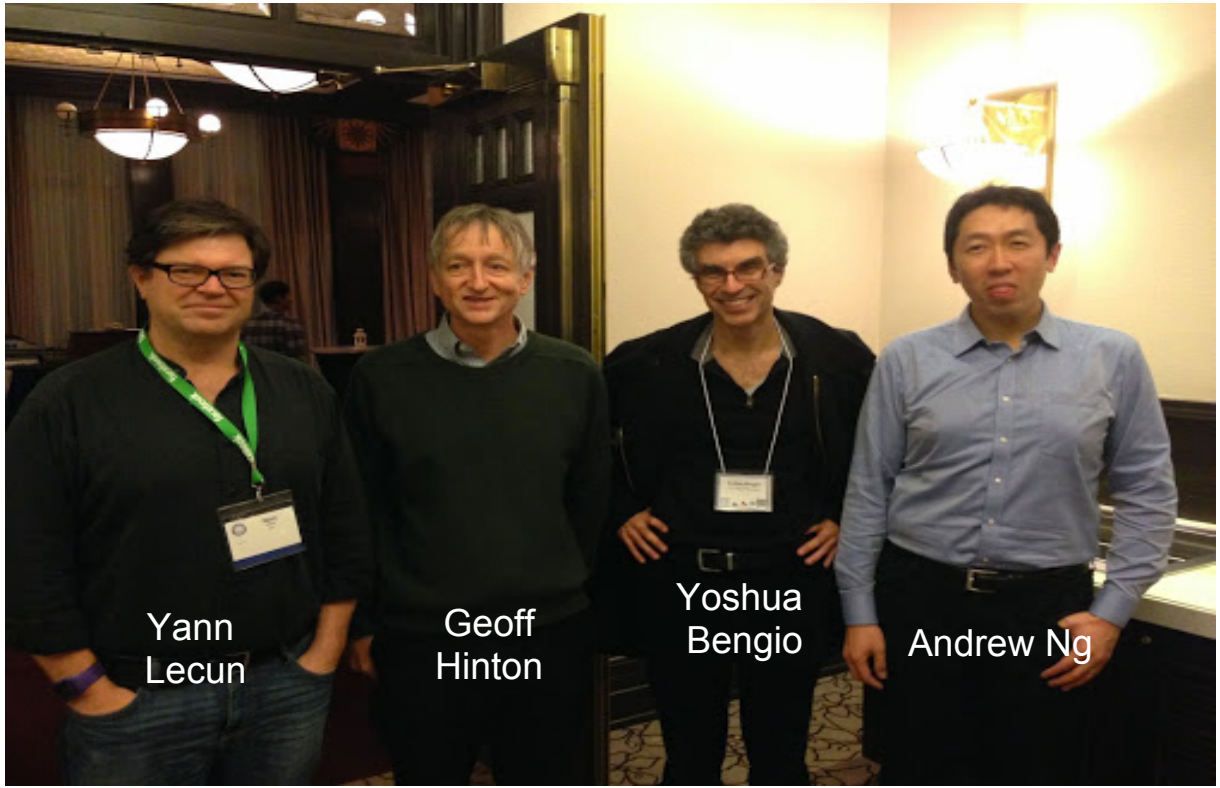


Figura inspirada em Dallagnol (2016). ;)

Esses são os caras!



Ferramentas

Treinamento é uma arte

- técnica de otimização
- quantidade de épocas de treinamento,
- quantidade de camadas, de unidades ocultas,
- tipo de cada camada,
- função de custo, regularização,
- taxa de aprendizagem, momentum, escalonamento,
- early stopping, weight decay, tamanho do minilote,
- ....

Não banque o herói!

Y. Bengio. Practical Recommendations for Gradient-Based Training of Deep Architectures, 2012

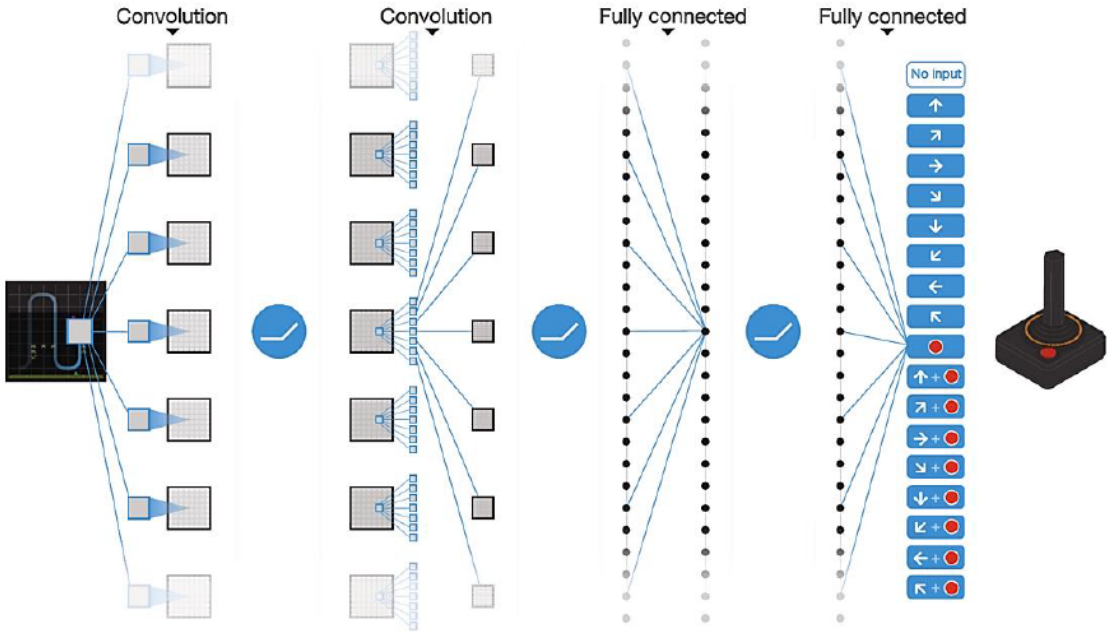
Ferramentas

- Caffe (http://caffe.berkeleyvision.org/)
- Torch (http://torch.ch/)
- TensorFlow (https://www.tensorflow.org/)
- MXNet (https://mxnet.readthedocs.io/}
- Theano (http://deeplearning.net/software/theano/)

Tendências

- Aprendizado multimodal – redes treinadas com texto mais imagem, ou áudio mais vídeo, etc..
- Modelos de atenção (*atention models*)
- Modularização – reuso e composição de modelos
- Deep Q-Learning

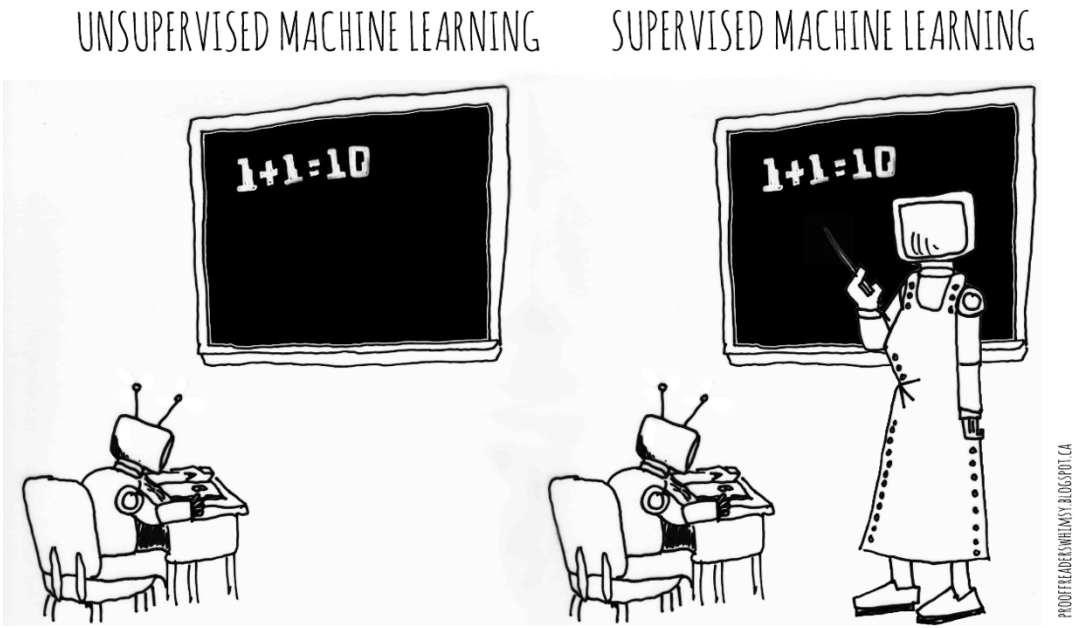
Tendências - Deep Q-Learning



AlphaGo

Aprendizado não supervisionado

Possivelmente, a grande fronteira a ser alcançada!



INTRODUÇÃO À APRENDIZAGEM PROFUNDA

OBRIGADO!

Eduardo Bezerra (ebezerra@cefet-rj.br)