

# Introdução ao Aprendizado Profundo

1001513 – Aprendizado de Máquina 2

Turma A – 2023/2

Prof. Murilo Naldi



[naldi@ufscar.br](mailto:naldi@ufscar.br)



# Agradecimentos

- Pessoas que colaboraram com a produção deste material: Ricardo Cerri, Diego Silva

# Introdução

3

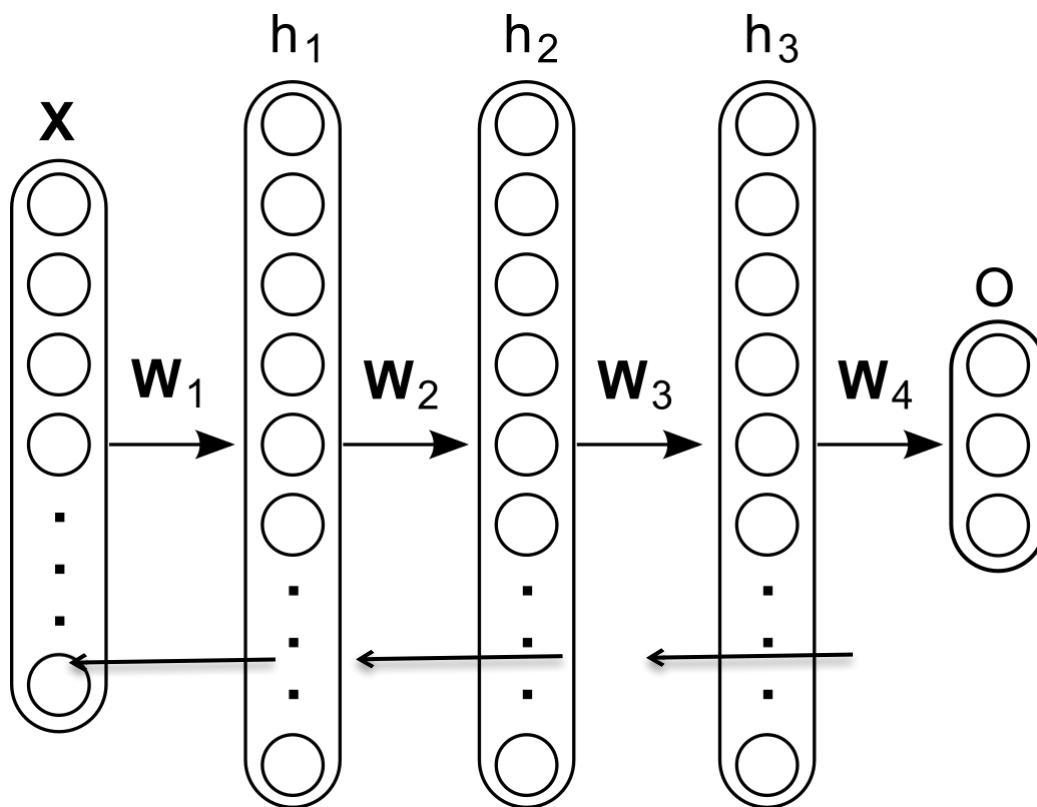
## □ Teorema da Aproximação Universal

- O teorema da aproximação universal diz que uma rede neural com uma única camada escondida e um número finito de neurônios pode aproximar qualquer função contínua.
- No entanto, o teorema não fala nada sobre tempo de treinamento, facilidade de implementação ou generalização.

# Introdução

4

- Em uma MLP eu posso ter  $L$  camadas ocultas



# Introdução

5

## □ Aprendizado Profundo

- Aprender modelos com múltiplas camadas ocultas
  - Cada camada corresponde a uma representação
  - Cada unidade representa uma característica da entrada
  - Unidades podem ser ativadas simultaneamente

# Justificativa Teórica

6

- Arquiteturas profundas podem representar funções de maneira mais compacta
  - Há funções que podem ser representadas com uma única camada com um número exponencial de neurônios
    - Porém um número polinomial de neurônios se pudermos aumentar o número de camadas
  - *Exploring Strategies for Training Deep Neural Networks*
    - Larrochelle et. al. 2009

# Exemplos de sucesso

7


- Carros auto-dirigíveis
  - Mistura de diferentes tecnologias
  - Diversos sensores
  - Estruturas complexas
  - Objetivo:
    - navegação sem mapas



# Exemplos de sucesso

8

- Processamento de linguagem natural (PLN)
  - Responder a perguntas, modelagem de linguagem, classificação de texto, análise de mensagens ou sentimentos, etc
  - Representações distribuídas, redes neurais convolucionais, recorrentes e recursivas,  
*embedding*,  
*autoencoder*, etc

The logo for OpenAI ChatGPT, featuring the OpenAI logo (a stylized knot) and the text "OpenAI" and "ChatGPT" in white on a black background.

OpenAI  
ChatGPT



# Exemplos de sucesso

9

## □ Outras áreas:

- Agregação e sumarização de notícias com detecção de fraudes (*fakes*)
- Reconhecimento visual e de padrões
- Recomendação de conteúdo (e propaganda)
- Composição de padrões
  - Músicas, imagens, etc

# Desafios das redes profundas

10

- *“The standard learning strategy—consisting of randomly initializing the weights of the network and applying gradient descent using backpropagation—is known empirically to find poor solutions for networks with 3 or more hidden layers ... For that reason, artificial neural networks have been limited to one or two hidden layers”*

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1).

# Por que é difícil treinar?

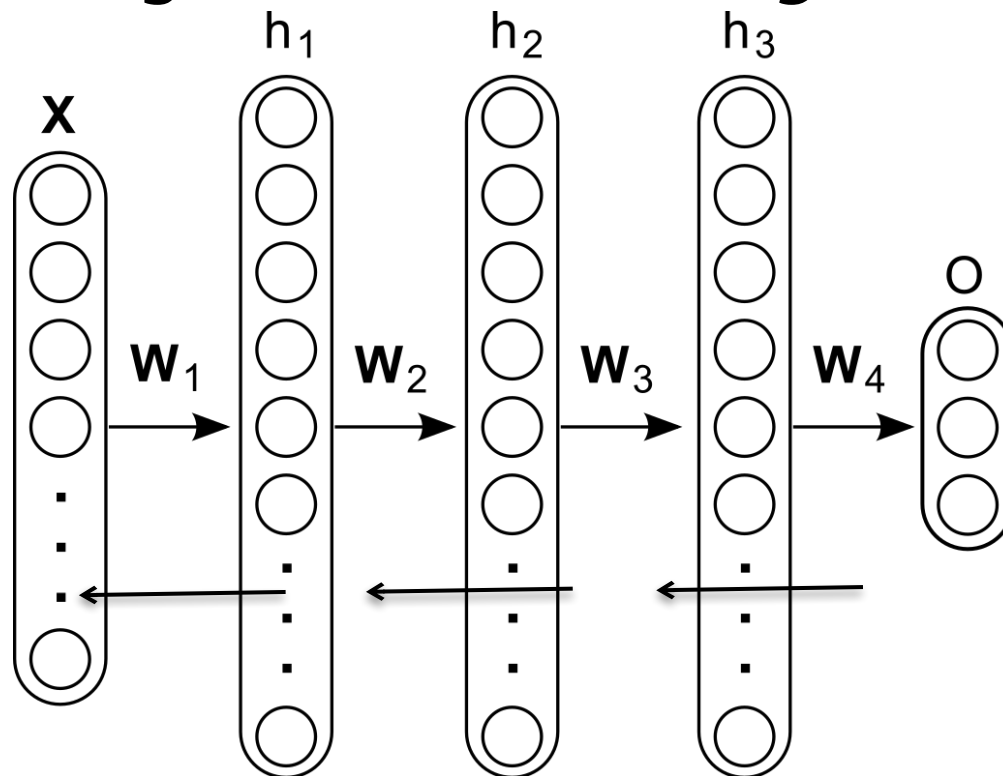
11

- Durante o treinamento, os pesos das redes neurais recebem uma atualização proporcional à derivada parcial da função de erro em relação ao peso atual
- A medida que o comprimento da sequência aumenta, espera-se que a magnitude do gradiente diminua (ou cresça incontrolavelmente), retardando o processo de treinamento.
- Problemas como desaparecimento (ou explosão) do gradiente...

# Por que é difícil treinar?

12

- Gradientes que desaparecem ou explodem impossibilitam o ajuste correto do modelo
  - Podendo gerar *underfitting*



# Por que é difícil treinar?

13

## □ Dilema da *viés vs variância*

- Viés (*bias*): erro causado pela diferença entre a previsão esperada (ou média) do modelo e o valor verdadeiro
  - Ou seja, quão longe as previsões desses modelos estão do valor de treino
- Variância: erro causado pela variância da predição de um modelo
  - Ou seja, quanto o modelo varia em relação a sua predição

# Por que é difícil treinar?

14

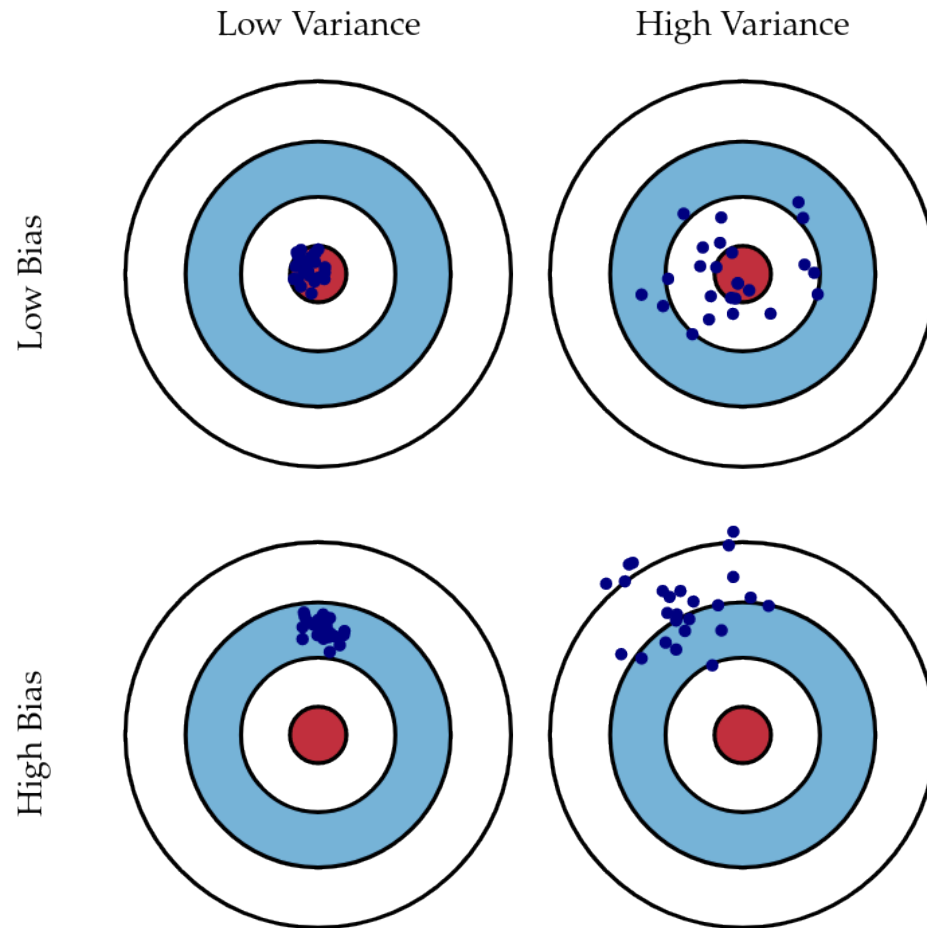


Fig. 1 Graphical illustration of bias and variance.

# Por que é difícil treinar?

15

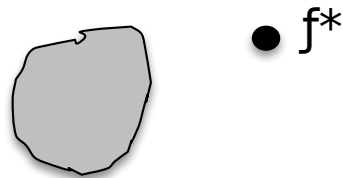
- Dilema da *viés vs variância*
  - Viés baixo e variância baixa: ideal
  - Viés baixo e variância alta: superajuste (*overfitting*) dos dados de treino, baixa generalização
  - Viés alto e variância baixa: subajuste (*underfitting*) aos dados de treino, se perdendo na predição
  - Viés alto e variância alta: caos

# Por que é difícil treinar?

16

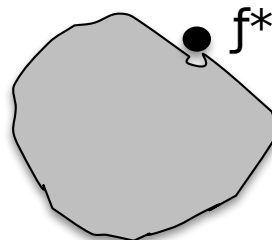
## □ Muitos parâmetros

- Explora um espaço muito maior de funções
- Difícil ajustá-los individualmente de forma correta

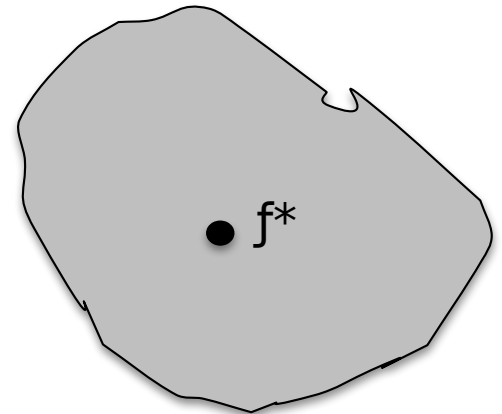


Possível  $f$

Baixa variância  
*Bias* alto



Possível  $f$



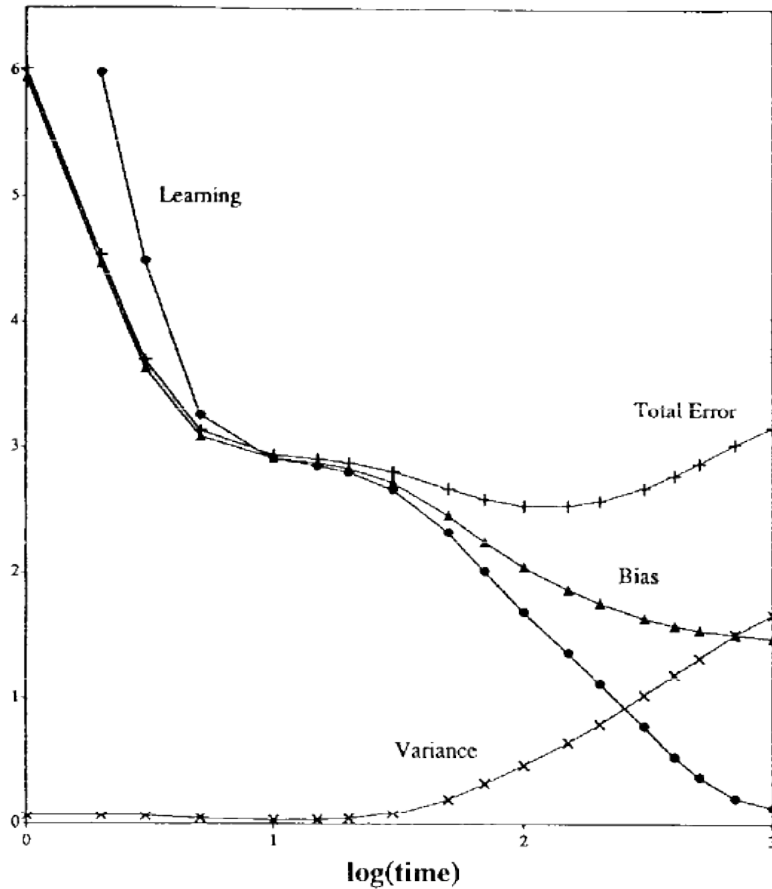
Possível  $f$

Alta variância  
*Bias* baixo

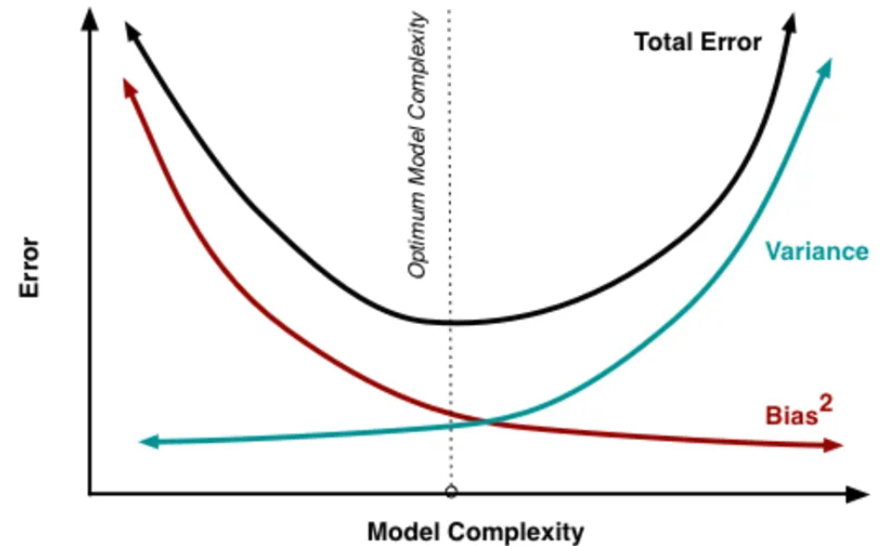


# Por que é difícil treinar?

17



S. Geman, E. Bienenstock and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," in Neural Computation, vol. 4, no. 1, pp. 1-58, Jan. 1992, doi: 10.1162/neco.1992.4.1.1.



Bias-variance dilemma. Zaid Alissa Almaliki. Towards Data Science. <https://towardsdatascience.com/bias-variance-dilemma-74e5f1f52b12>

# Por que é difícil treinar?

18

- Usar métodos de regularização melhores
  - Treinamento não supervisionado
  - Treinamento com *dropout* estocástico
  - Utilização de outros estimadores

# Pré-treino

19

- Inicializar as camadas escondidas utilizando aprendizado não supervisionado
  - Força a rede a aprender a estrutura da distribuição dos dados de entrada
  - Encoraja as camadas escondidas a codificar essa estrutura

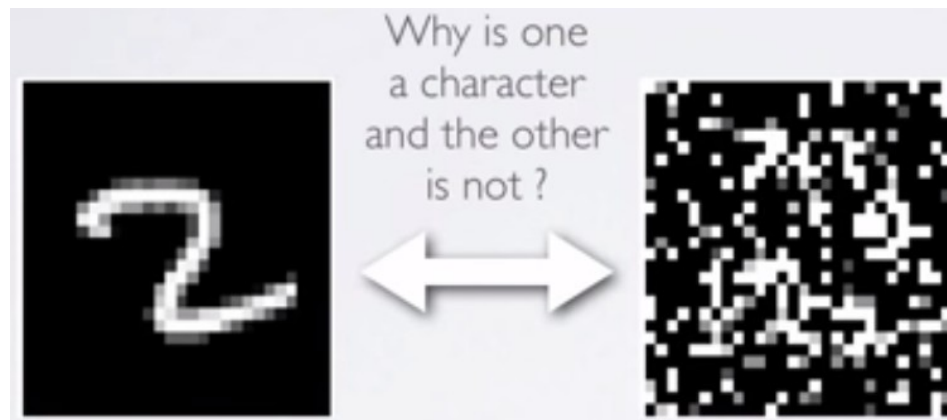


Figura de Larrochelle

# Pré-treino

20

- Durante a fase de aprendizado, a rede não supervisionada tenta imitar os dados fornecidos
  - Usa o erro em sua saída imitada para se corrigir
  - Às vezes, o erro é expresso como uma baixa probabilidade de ocorrência da saída incorreta ou pode ser expresso como um estado instável

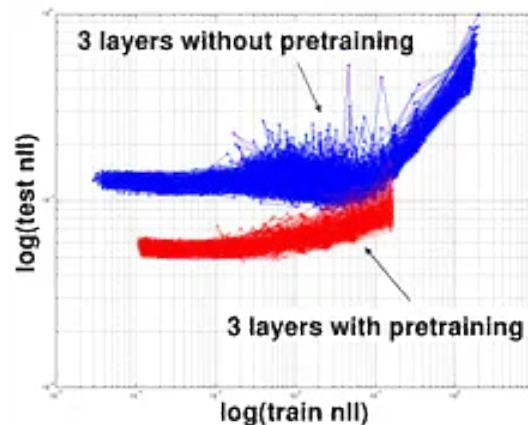
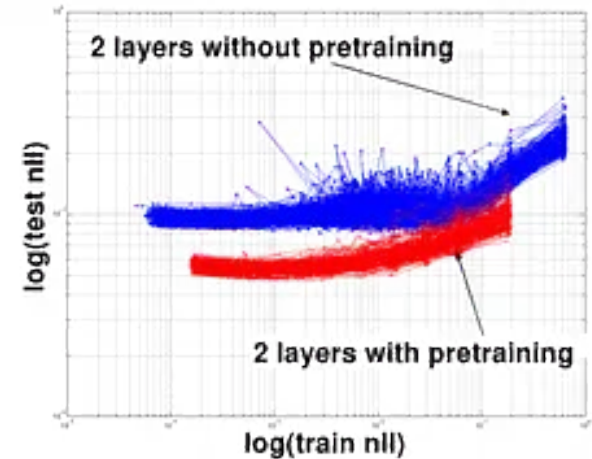
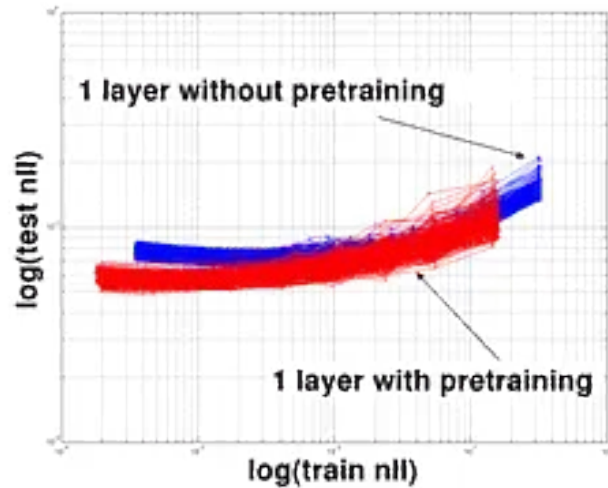
# Pré-treino

21

- Pode ser benéfico para:
  - Aprendizado de características inerentes da estrutura dos dados, sem os rótulos
  - Um modelo pré-treinado sobre dados não rotulados pode ser um bom início (*transfer learning*)
  - Pode servir como um método de regulação, já que encoraja modelo mais genérico
  - Boa inicialização ajuda evitar mínimos locais

# Pré-treino

22



Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010, March). Why does unsupervised pre-training help deep learning?. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 201-208). JMLR Workshop and Conference Proceedings.

# Pré-treino

23

- Procedimento ganancioso camada a camada
  - Treina uma camada por vez, da primeira até a última, utilizando aprendizado não supervisionado
  - Cada camada ajusta os parâmetros das camadas anteriores
    - Camadas anteriores vistas como extratores de características

# Pré-treino

24

- **Primeira camada:** encontra características que são mais comuns nos dados de treino do que em dados aleatórios
- **Segunda camada:** encontra combinações de características dos neurônios escondidos
- **Terceira camada:** combinações de combinações ...
- **Quarta camada:** ....
  - O pré-treino inicializa os parâmetros possibilitando regularização

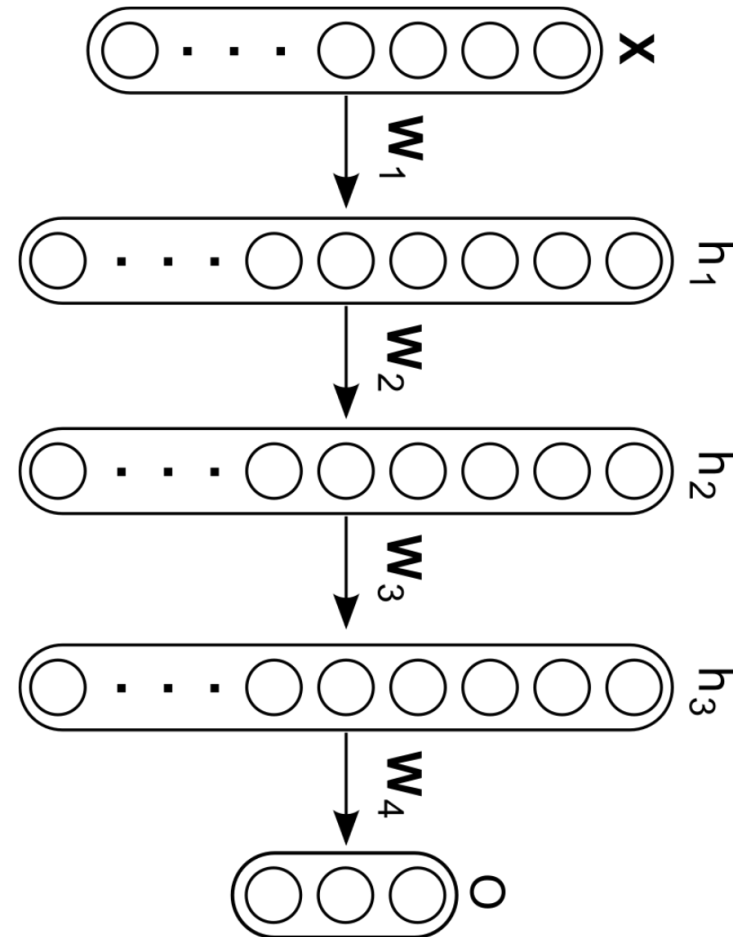


# Ajuste Fino

25

- Depois de treinadas todas as camadas
  - Adiciona camada de saída
  - Treina toda a rede com aprendizado supervisionado
- Todos os parâmetros são ajustados

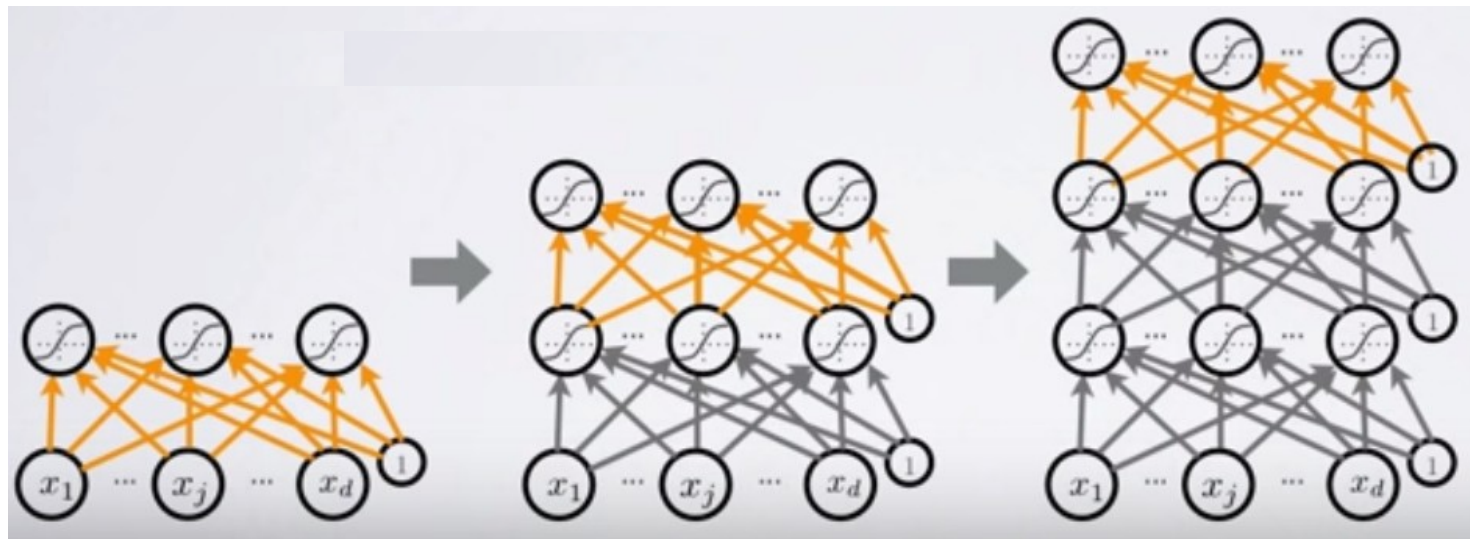
O ajuste é feito para tornar a rede mais discriminativa



# Pré-treino

26

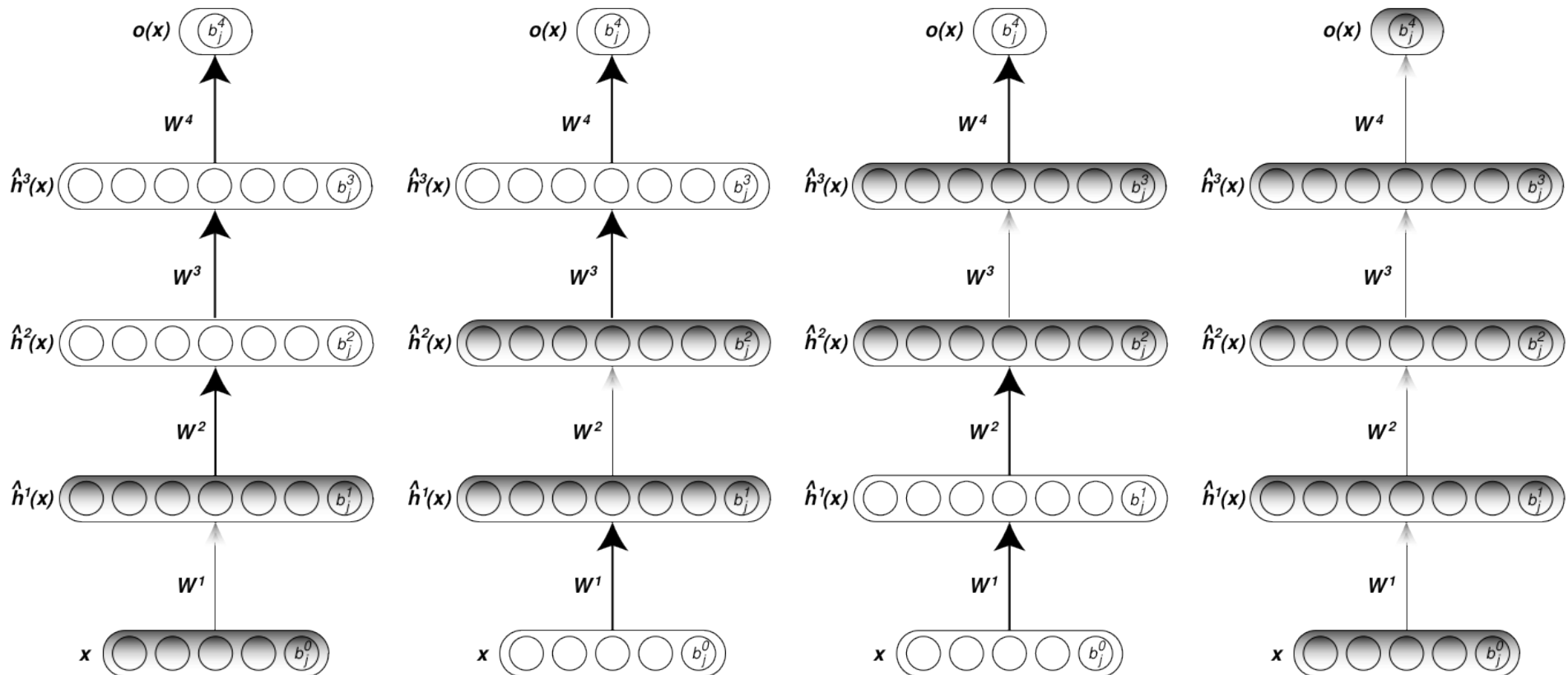
- Procedimento ganancioso camada a camada



Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1).

# Pré-treino

27



(a) First hidden layer pre-training (b) Second hidden layer pre-training (c) Third hidden layer pre-training (d) Fine-tuning of whole network

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1).

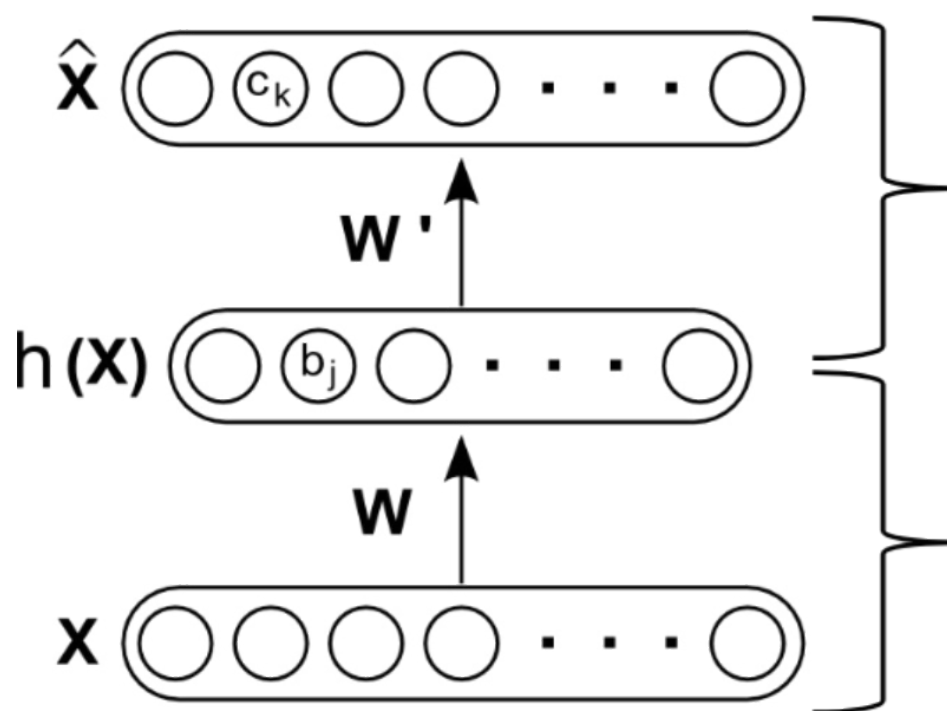
# Exemplo - Autoencoder

28

- Rede neural treinada para reproduzir sua entrada na camada de saída (não supervisionada)
- A representação é aprendida na camada do meio (camada escondida)
- Duas partes:
  - Codificação
  - Decodificação

# Autoencoder

29



Decodificação

$$\begin{aligned}\hat{\mathbf{x}} &= o(\hat{a}(\mathbf{x})) \\ &= \text{sigm}(\mathbf{c} + \mathbf{W}'\mathbf{h}(\mathbf{x}))\end{aligned}$$

Codificação

$$\begin{aligned}\mathbf{h}(\mathbf{x}) &= g(a(\mathbf{x})) \\ &= \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})\end{aligned}$$

# Autoencoder – Loss Function

30

- Motivação: a representação escondida mantém toda a informação da entrada
  - Usando uma camada escondida menor que a entrada, o *autoencoder* vai comprimir a informação
  - Ignora a informação que não é útil
- A função de perda (*loss function*) compara a entrada com a saída
  - Treinamos o *autoencoder* para minimizar-lá por meio do gradiente descendente

# Autoencoder – Loss Function

31

□ Para entradas binárias

$$l(f(\mathbf{x})) = -\sum_k \left( x_k \log(\hat{x}_k) + (1 - x_k) \log(1 - \hat{x}_k) \right)$$

▣ Se  $x_k = 1$ , tentamos “puxar”  $\hat{x}_k$  para 1

▣ Se  $x_k = 0$ , tentamos “puxar”  $\hat{x}_k$  para 0

# Autoencoder – Loss Function

32

- Para entradas com valores reais

$$l(f(\mathbf{x})) = \frac{1}{2} \sum_k (\hat{x}_k - x_k)^2$$

- ▣ Soma das diferenças ao quadrado (distância Euclidiana quadrática)
- ▣ Função de ativação linear na camada de saída



# Autoencoder – Loss Function

33

- O gradiente da função de erro tem a seguinte forma:

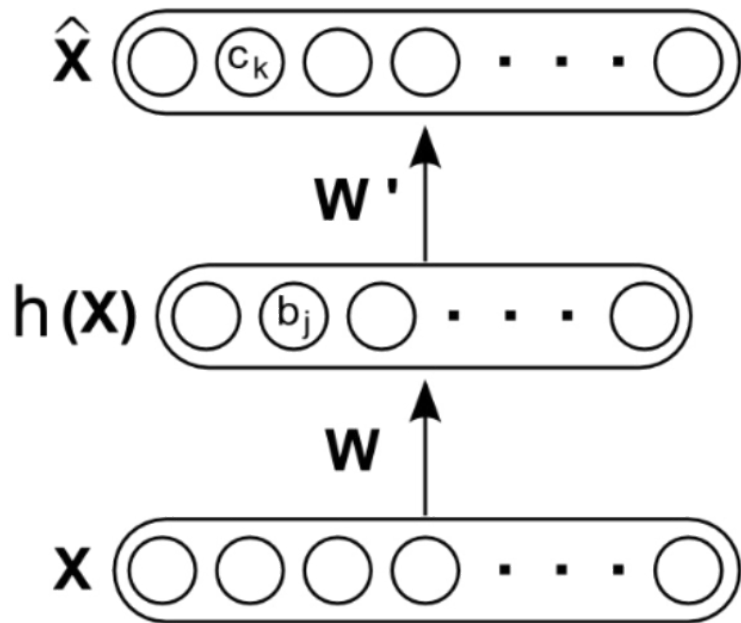
$$\nabla_{\hat{\mathbf{a}}(\mathbf{x}^{(t)})} l\left(f\left(\mathbf{x}^{(t)}\right)\right) = \hat{\mathbf{x}}^{(t)} - \mathbf{x}^{(t)}$$

- ▣ Parâmetros obtidos usando o algoritmo Backpropagation, como em uma rede neural convencional

# Autoencoder

34

- Camada oculta menor do que a camada de entrada



Aprende bem a distribuição  
dos dados de treino



Não generaliza bem para  
dados diferentes



# Camada Sobrecompleta

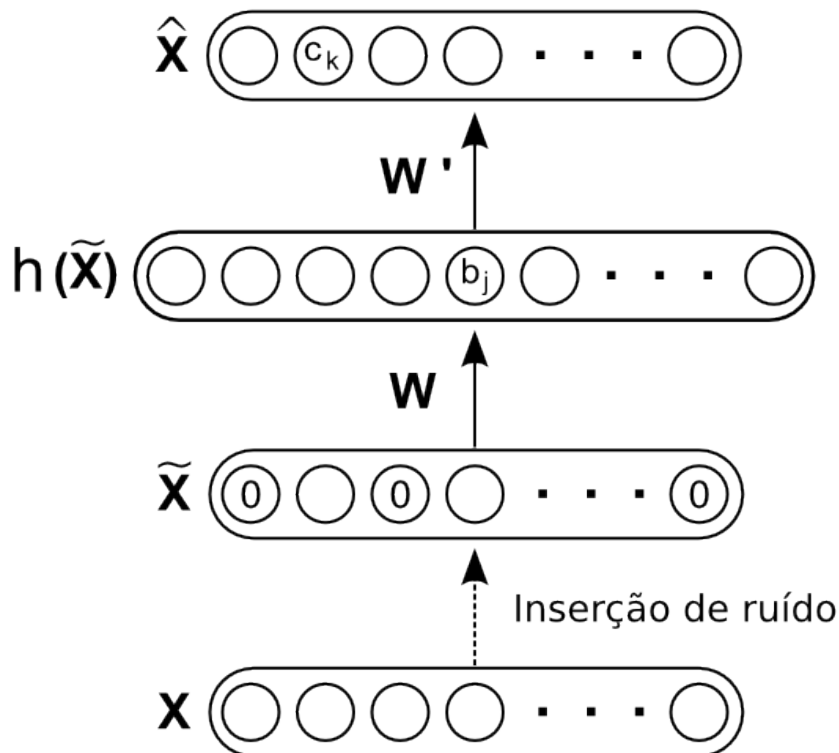
35

- Camada oculta maior do que a camada de entrada
  - Não há compressão na camada oculta
  - Cada unidade oculta pode apenas copiar um diferente componente de entrada
  - Não há garantia de que as unidades ocultas irão extrair uma estrutura que tem significado

# Denoising

36

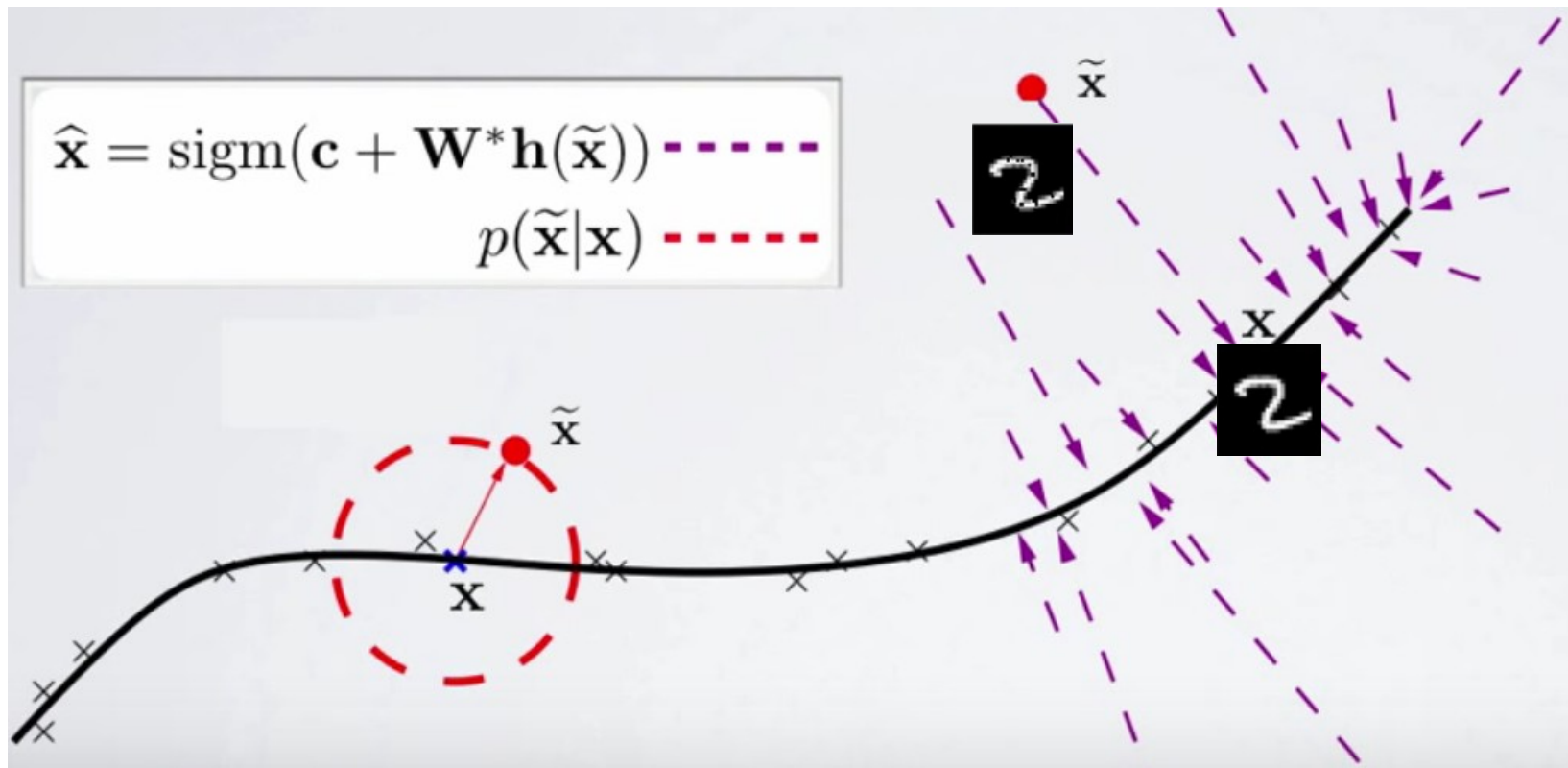
- Usar autoencoder para retirar ruído
  - Entrada corrompida, mas erro sobre original



# Denoising

37

## □ Ilustração



Créditos: Hugo Larrochelle

# Pré-treino Autoencoder

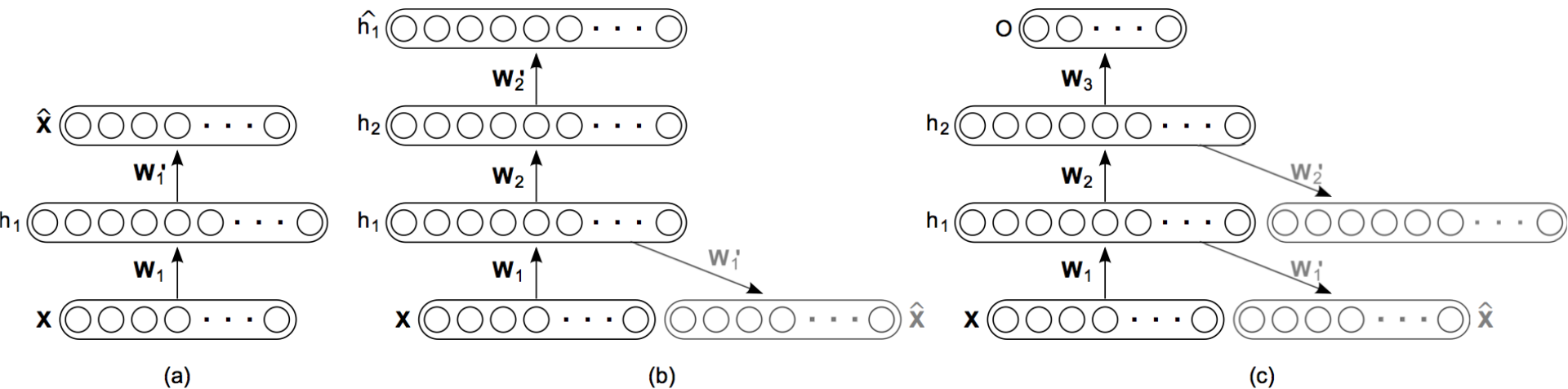
38

- Autoencoder também podem ter mais de uma camada intermediária
  - Por se tratar de um processo não supervisionado, podemos aplicar pré-treino
    - Camada a camada
    - Comparando os dados do conjunto

# Pré-treino

39

- Procedimento ganancioso camada a camada
  - Autoencoder



# Exemplo

40

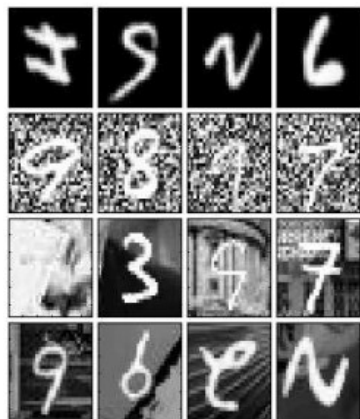
## An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation

Hugo Larochelle  
Dumitru Erhan  
Aaron Courville  
James Bergstra  
Yoshua Bengio

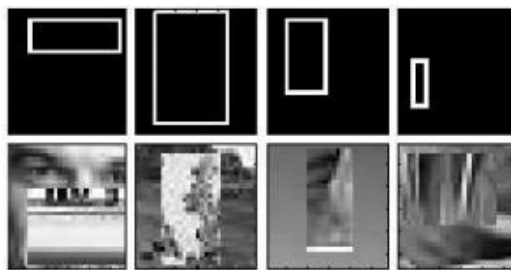
Dept. IRO, Université de Montréal C.P. 6128, Montreal, Qc, H3C 3J7, Canada

LAROCHEH@IRO.UMONTREAL.CA  
ERHANDUM@IRO.UMONTREAL.CA  
COURVILA@IRO.UMONTREAL.CA  
BERGSTRJ@IRO.UMONTREAL.CA  
BENGIOY@IRO.UMONTREAL.CA

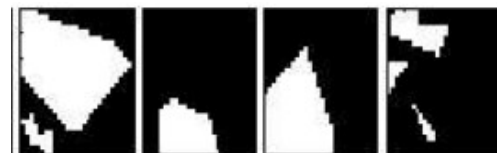
## Dados com variações



Variações no MNIST



Variações de retângulos



Convexa ou não



# Exemplo

41

## An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation

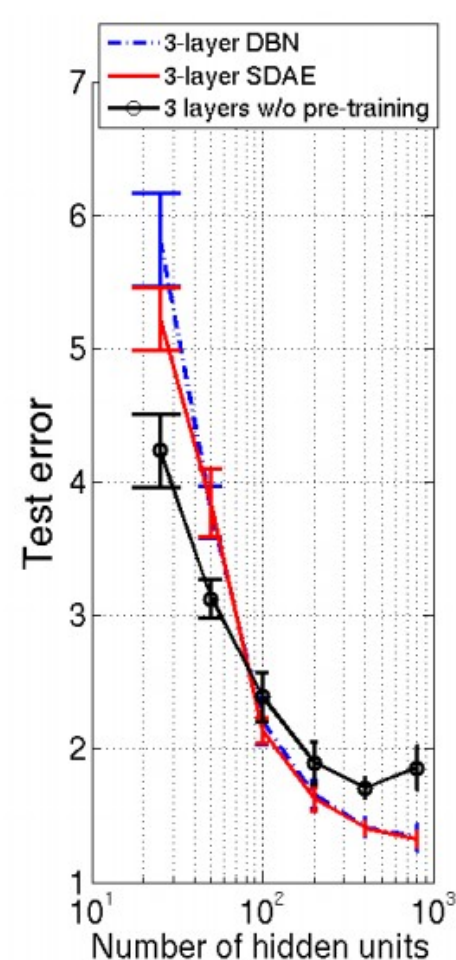
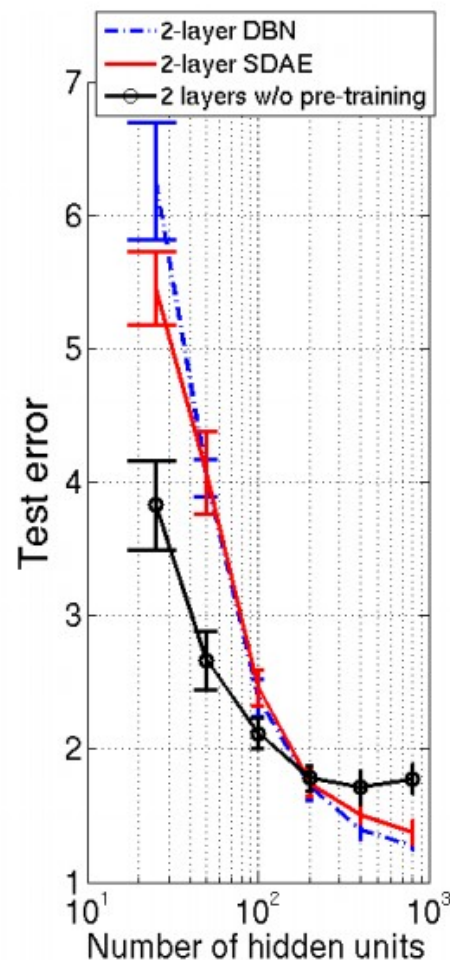
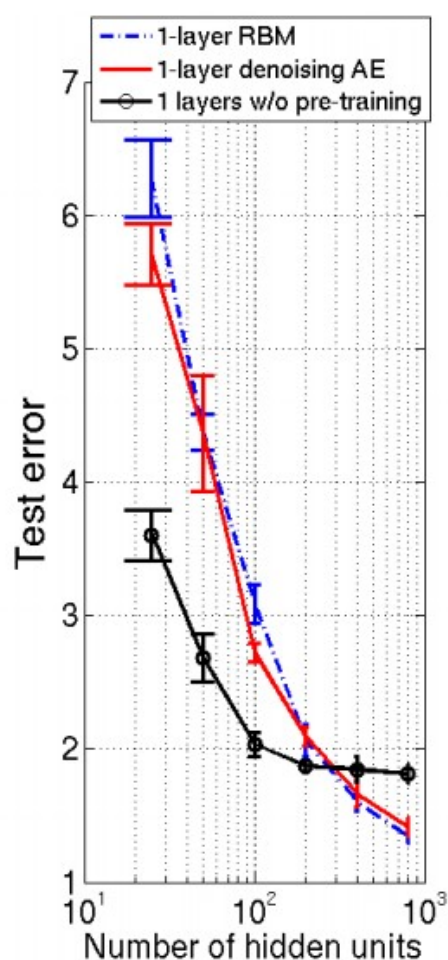
Network		MNIST-small classif. test error	MNIST-rotation classif. test error
Type	Depth		
Deep net	1	<b>4.14</b> % $\pm$ 0.17	15.22 % $\pm$ 0.31
	2	<b>4.03</b> % $\pm$ 0.17	<b>10.63</b> % $\pm$ 0.27
	3	<b>4.24</b> % $\pm$ 0.18	11.98 % $\pm$ 0.28
	4	4.47 % $\pm$ 0.18	11.73 % $\pm$ 0.29
Deep net + autoencoder	1	3.87 % $\pm$ 0.17	11.43 % $\pm$ 0.28
	2	<b>3.38</b> % $\pm$ 0.16	9.88 % $\pm$ 0.26
	3	<b>3.37</b> % $\pm$ 0.16	<b>9.22</b> % $\pm$ 0.25
	4	<b>3.39</b> % $\pm$ 0.16	<b>9.20</b> % $\pm$ 0.25
Deep net + RBM	1	3.17 % $\pm$ 0.15	10.47 % $\pm$ 0.27
	2	<b>2.74</b> % $\pm$ 0.14	9.54 % $\pm$ 0.26
	3	<b>2.71</b> % $\pm$ 0.14	<b>8.80</b> % $\pm$ 0.25
	4	<b>2.72</b> % $\pm$ 0.14	<b>8.83</b> % $\pm$ 0.24

# Exemplo

43

## Why Does Unsupervised Pre-training Help Deep Learning?

Dumitru Erhan\*  
Yoshua Bengio  
Aaron Courville  
Pierre-Antoine Manzagol  
Pascal Vincent



# Deep Autoencoder

44

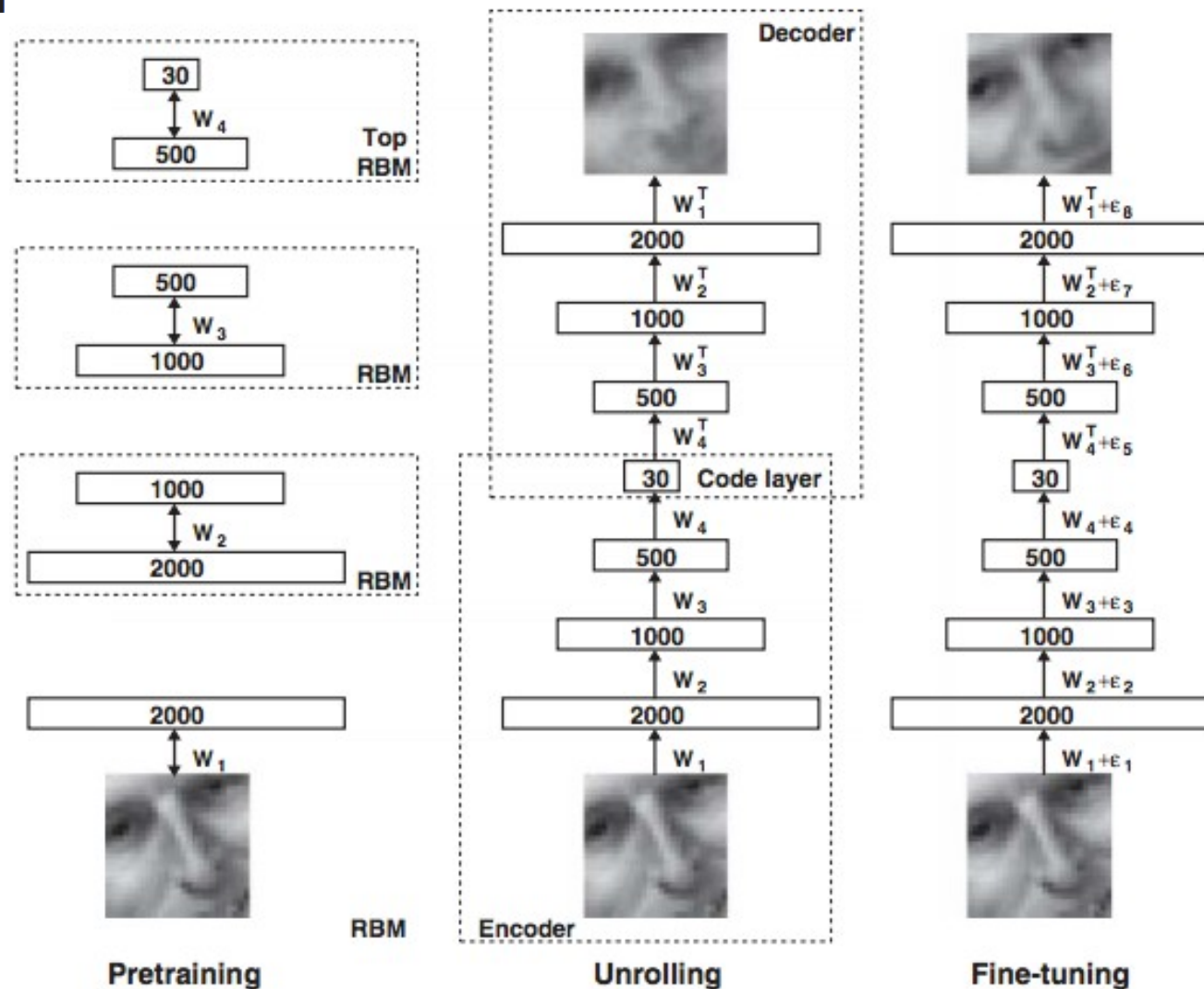
- Pré-treino com RBMs (Restricted Boltzmann Machines) pode ser utilizado para inicializar um Deep Autoencoder
  - Primeiro treina-se uma Stacked RBM
  - Depois do pré-treino, as RBMs são “desenroladas” para criar um Deep Autoencoder
  - Depois é aplicado ajuste fino com Back-propagation

# Deep Autoencoder

45

## Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton\* and R. R. Salakhutdinov



# Deep Autoencoder

46

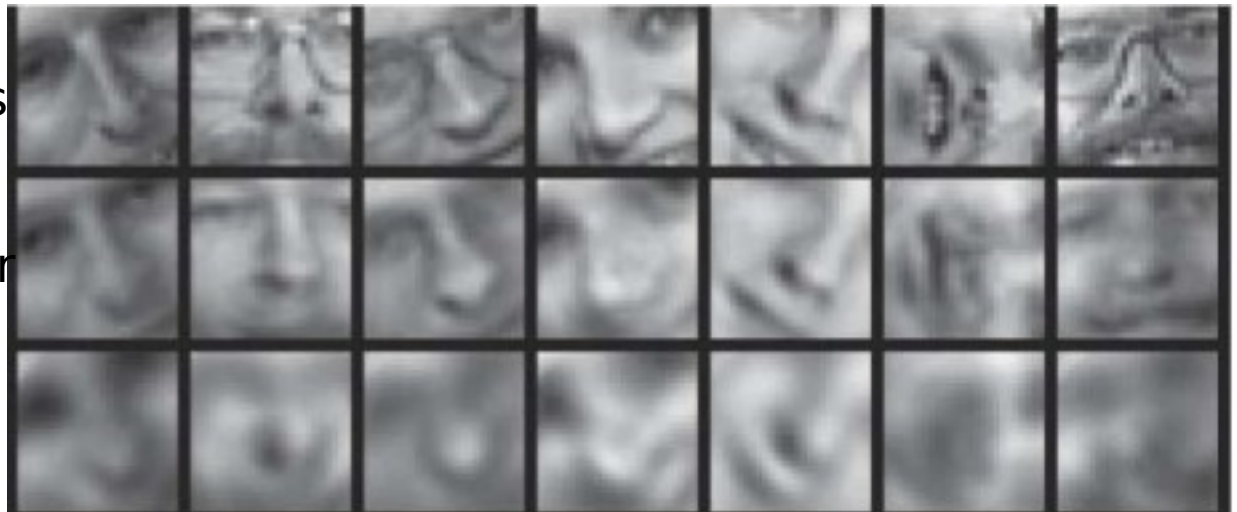
## Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton\* and R. R. Salakhutdinov

Dados originais

Deep autoencoder

PCA



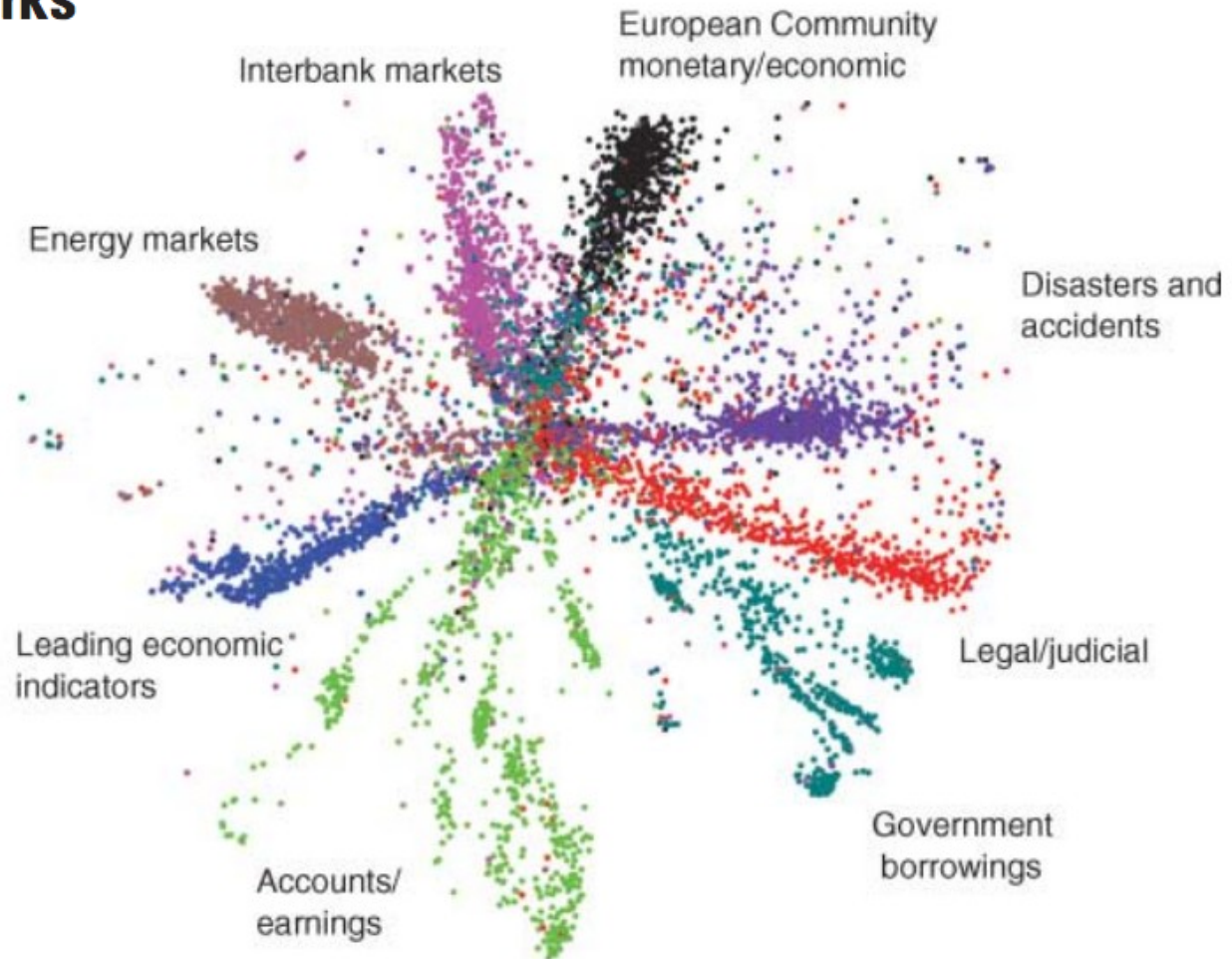
# Deep Autoencoder

47

## Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton\* and R. R. Salakhutdinov

Visualização





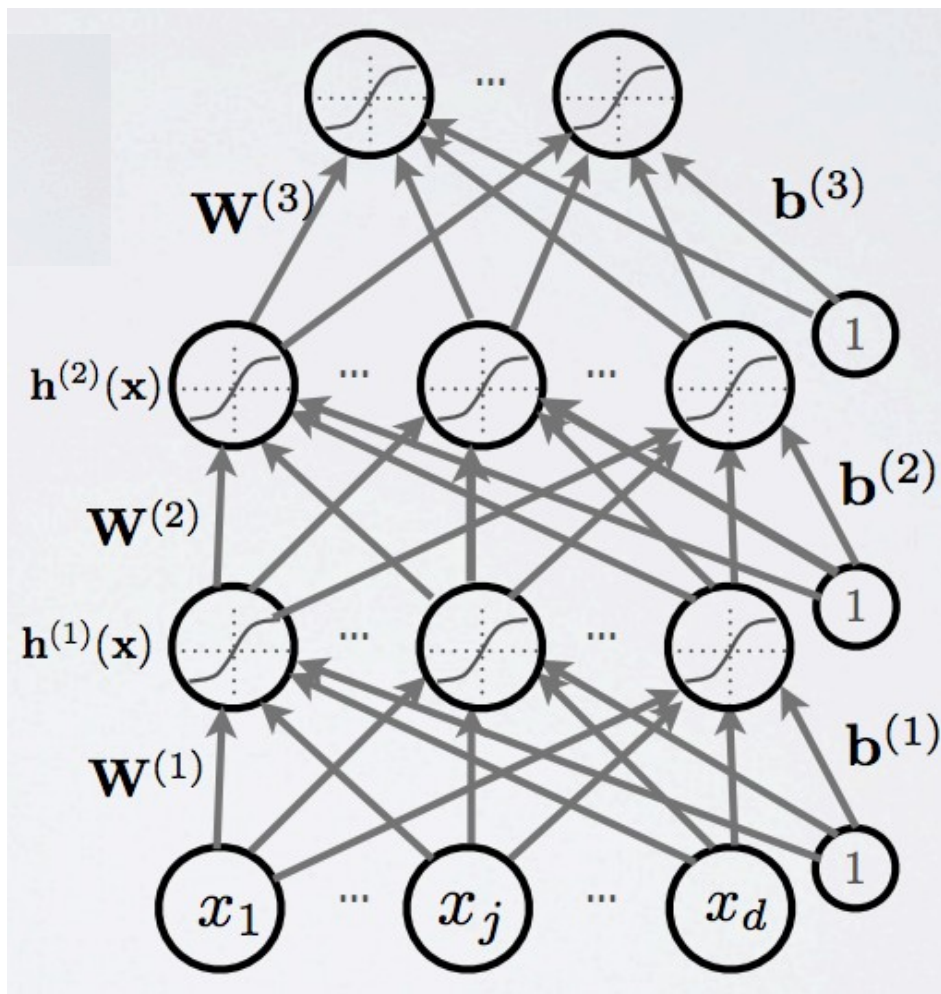
# Dropout

48

- *Dropout* é uma técnica de regularização usada para evitar *overfitting*.
- Ideia: é zerar neurônios escondidos aleatoriamente durante atualização ou iteração de treinamento
  - Resultado:
    - Neurônios escondidos não podem se coadaptar a outros neurônios escondidos
    - Cada neurônio escondido é forçado a extrair características mais gerais

# Dropout

49

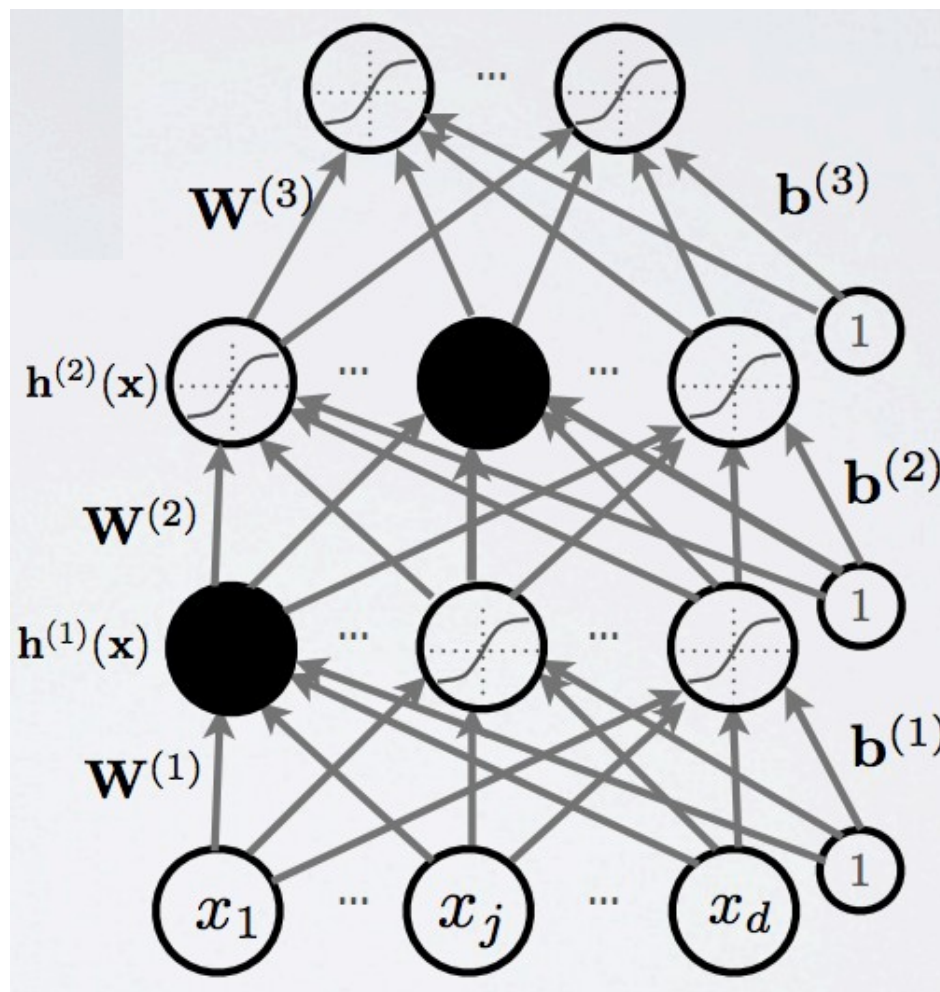


Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929-1958.



# Dropout

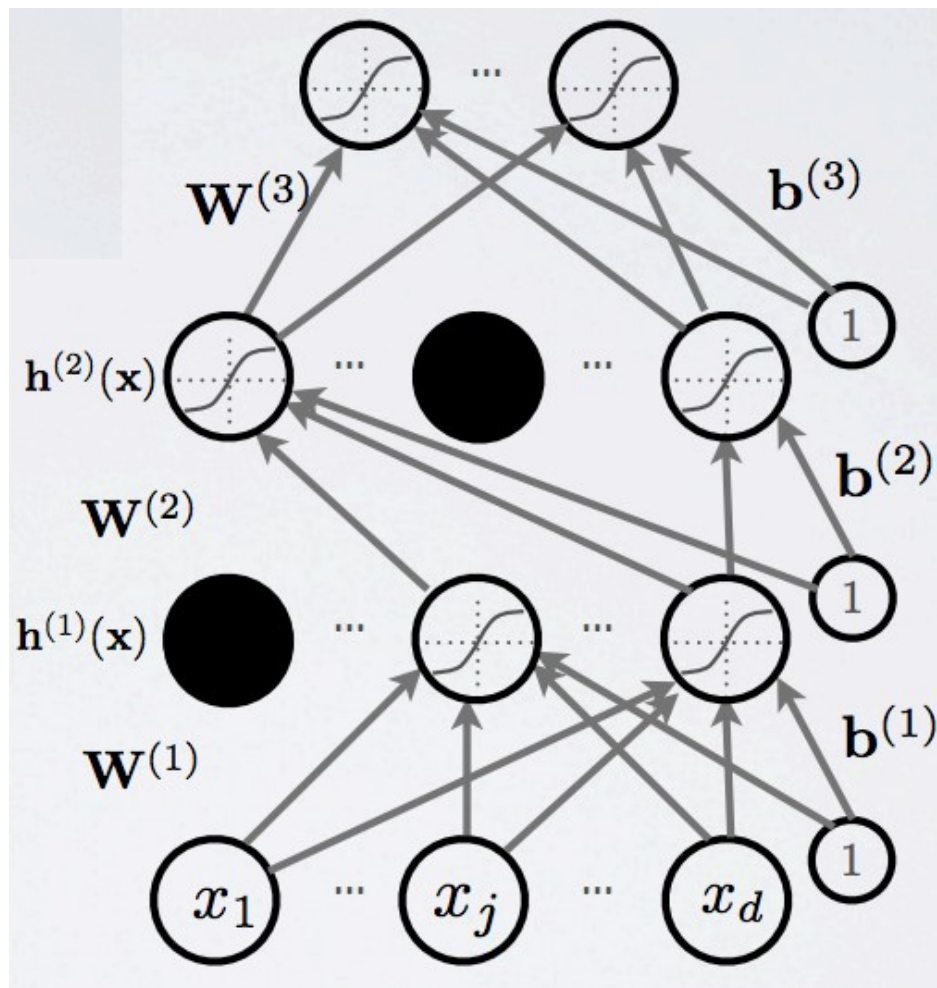
50



Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929-1958.

# Dropout

51



Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929-1958.

# Treino

52

- Rede com *dropout* é conhecida como “*thinned*” ou diluída/reduzida
- Treinamento é feito emitindo os sinais para frente desta rede e atualizando seus pesos com retropropagação
  - Os gradientes são calculados sobre a média dos erros em *mini-batches*
  - Pesos conectados a neurônios que sofreram *dropout* não contribuem

# Resultados

53

Method	Unit Type	Architecture	Error %
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, ( $5 \times 240$ ) units	0.94
DBN + finetuning (Hinton and Salakhutdinov, 2006)	Logistic	500-500-2000	1.18
DBM + finetuning (Salakhutdinov and Hinton, 2009)	Logistic	500-500-2000	0.96
DBN + dropout finetuning	Logistic	500-500-2000	0.92
DBM + dropout finetuning	Logistic	500-500-2000	<b>0.79</b>

Table 2: Comparison of different models on MNIST.

# Resultados

54

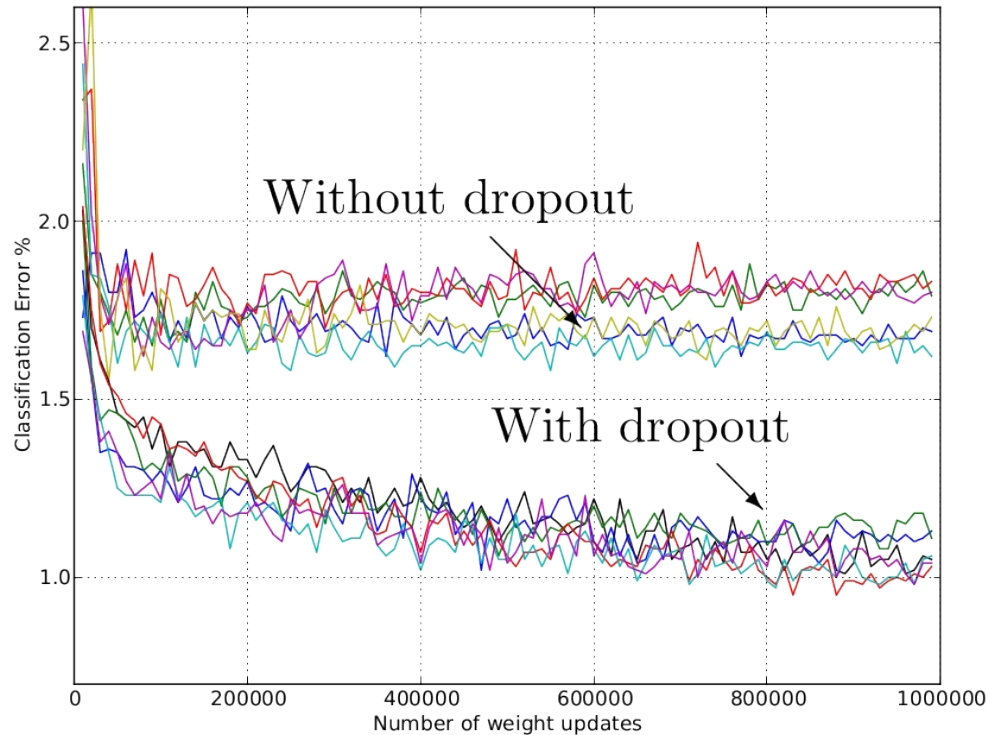


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (January 2014), 1929–1958.

# Deep Learning

58

## □ Mais informações

- Aulas de Hugo Larrochelle
- <https://youtube.com/playlist?list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH&si=yvLplkg6P2W0ORP->

# Referências

59

- Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1).
- S. Geman, E. Bienenstock and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," in *Neural Computation*, vol. 4, no. 1, pp. 1-58, Jan. 1992, doi: 10.1162/neco.1992.4.1.1.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (January 2014), 1929–1958.
- Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010, March). Why does unsupervised pre-training help deep learning?. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 201-208). *JMLR Workshop and Conference Proceedings*.