Aprendizagem Profunda
(Autoencoders)

Marcondes Ricarte e Aluizio Fausto Ribeiro Araújo

Universidade Federal de Pernambuco

Centro de Informática

Conteúdo

- *Auto-encoder* Profundo
  - *Stacked Auto-encoder*
  - Outros Tipos de *Auto-encoders*
- Ferramentas

Auto-encoder Profundo
(*Deep Autoencoder*)

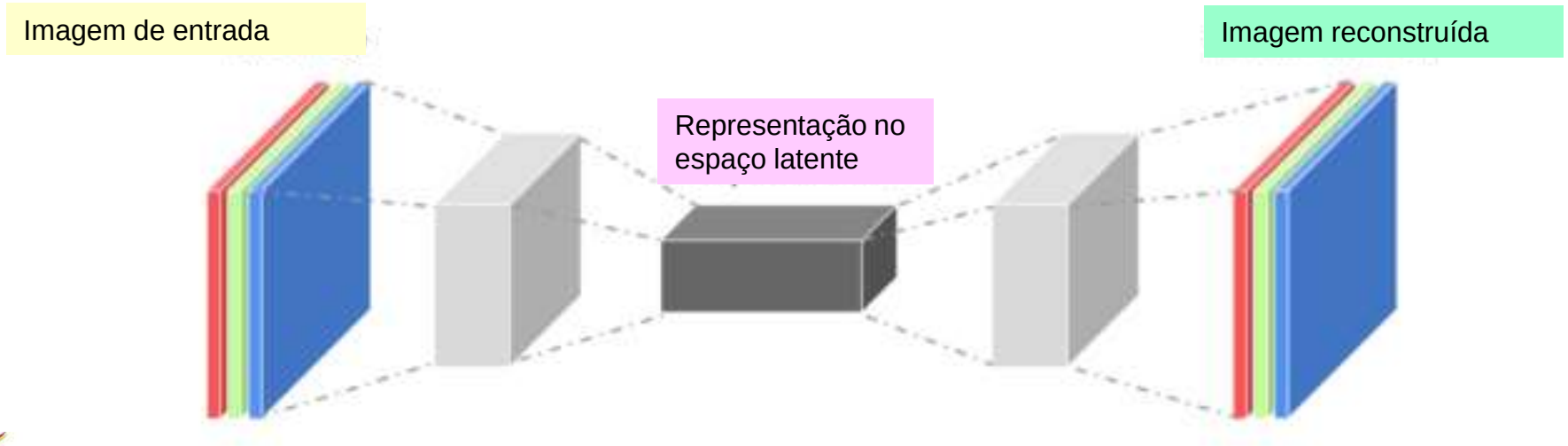
- Perspectiva do modelo:
  - "We expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object". LeCun, Bengio, Hinton, Nature, 2015;
  - "As I've said in previous statements: most of human and animal learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don't know how to make the cake". Yann LeCun, March 14, 2016 (Facebook).

Auto-encoder Profundo
(*Deep Autoencoder*)

- Definição:
  - *Autoencoders* capturam distribuição de padrões de entrada aprendendo a reconstruir as configurações de entrada. Um autoencoder é uma rede neural não supervisionada para extração de características que aprende um conjunto de parâmetros para reconstruir sua saída o mais próxima possível de sua entrada.
  - *Autoencoders* são modelos não-lineares que podem ter capacidade de generalização mais poderosa que o PCA,
    - Um *autoencoder* linear deve aprender a mesma representação que o PCA;
  - *Autoencoders* profundos têm várias camadas escondidas.

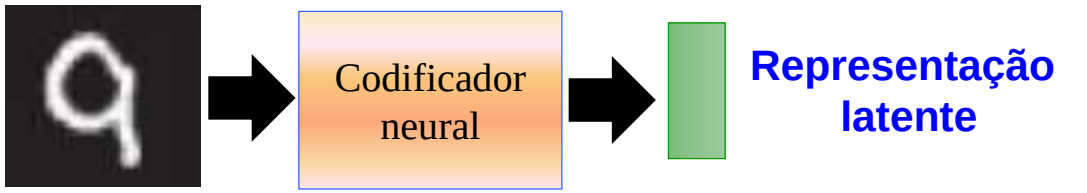
Auto-encoder Profundo
(*Deep Autoencoder*)

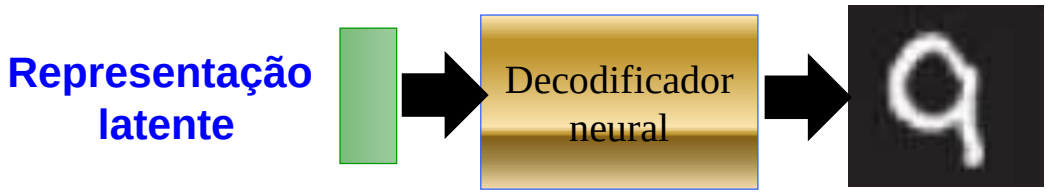
- Autoencoders foram originalmente pensados para reprodução dos padrões de entrada na saída,
  - Inicialmente propostos para lidar com imagens;
  - Objetiva-se reproduzir uma entrada de uma codificação aprendida;
- Estrutura:



Auto-encoder Profundo
(*Deep Autoencoder*)

- Principais operações:
  - Codificador: Compacta a entrada em um espaço latente;
  - Decodificador: Reconstrói a entrada do espaço latente;
- Exemplo:





Auto-encoder Profundo
(*Deep Autoencoder*)

- Propriedades dos *Autoencoders*:
  - Compactam dados semelhantes aos que foram treinados;
  - Saídas reconstruídas são degradadas com respeitos às entradas originais;
  - Pode apresentar sobretreinamento para redes maiores que o necessário.

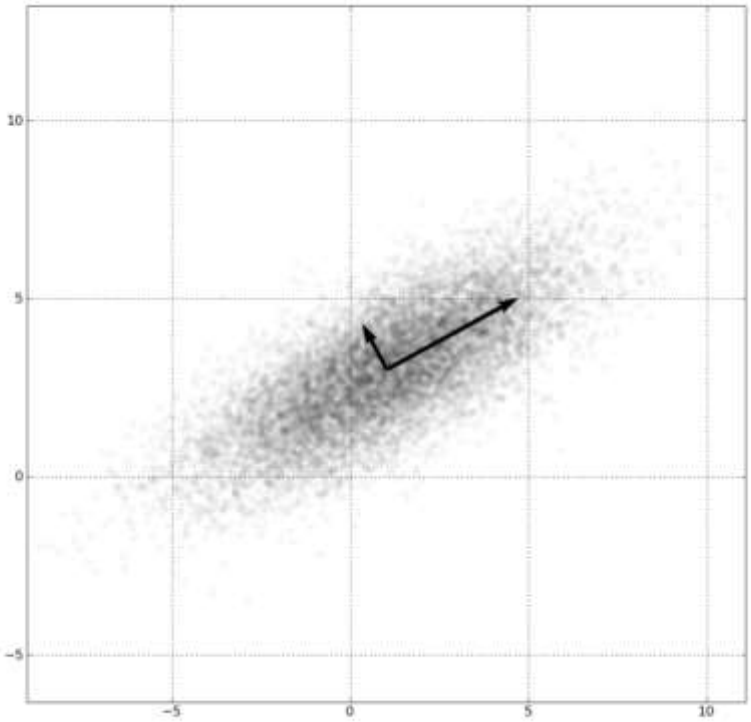
Auto-encoder Profundo
(*Deep Autoencoder*) - PCA

- Análise de Componentes Principais (*Principal Component Analysis* - PCA) é um procedimento matemático que transforma linearmente um conjunto de variáveis originais em outro com novas variáveis, chamadas de componentes.
  - A primeira das novas variáveis é a componente principal porque guarda a maior quantidade de informação sobre os dados;
  - O segundo componente representa a maior parte da variação que não foi representada pelo primeiro, assim, sucessivamente para os componentes de mais alta ordem;
  - PCA é usado para reduzir complexidade de dados multidimensionais;
  - Cálculo do PCA: Envolve determinação dos autovalores e autovetores na matriz de correlação (ou covariância) original;

Auto-encoder Profundo
(*Deep Autoencoder*) - PCA

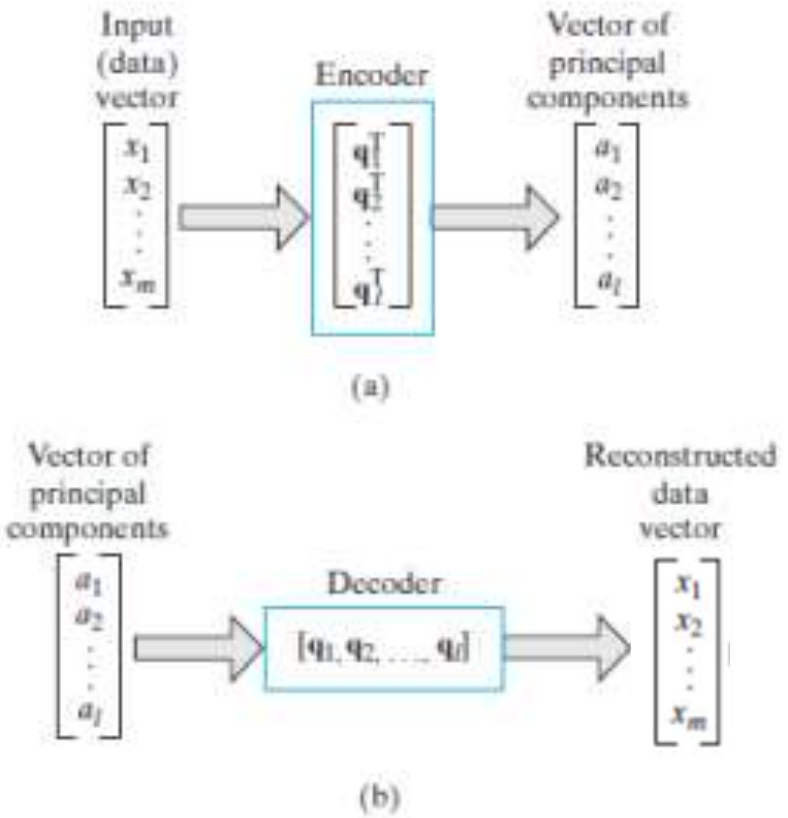
- PCA :
  - Reduz a dimensionalidade do espaço de um problema;
  - Cria novo conjunto de características que capture máximo de informação do conjunto original de entrada;
  - Gera novas características por combinação linear das originais;
  - Busca preservar a variância original do conjunto de entrada.

Mudança de eixos do PCA



Auto-encoder Profundo
(*Deep Autoencoder*) - PCA

- No processo de redução de dimensionalidade, a projeção linear de R^m para R^L é chamado de codificação (*encoding*).
- O processo inverso, a projeção de R^L para R^m é chamado de decodificação (*decoding*).



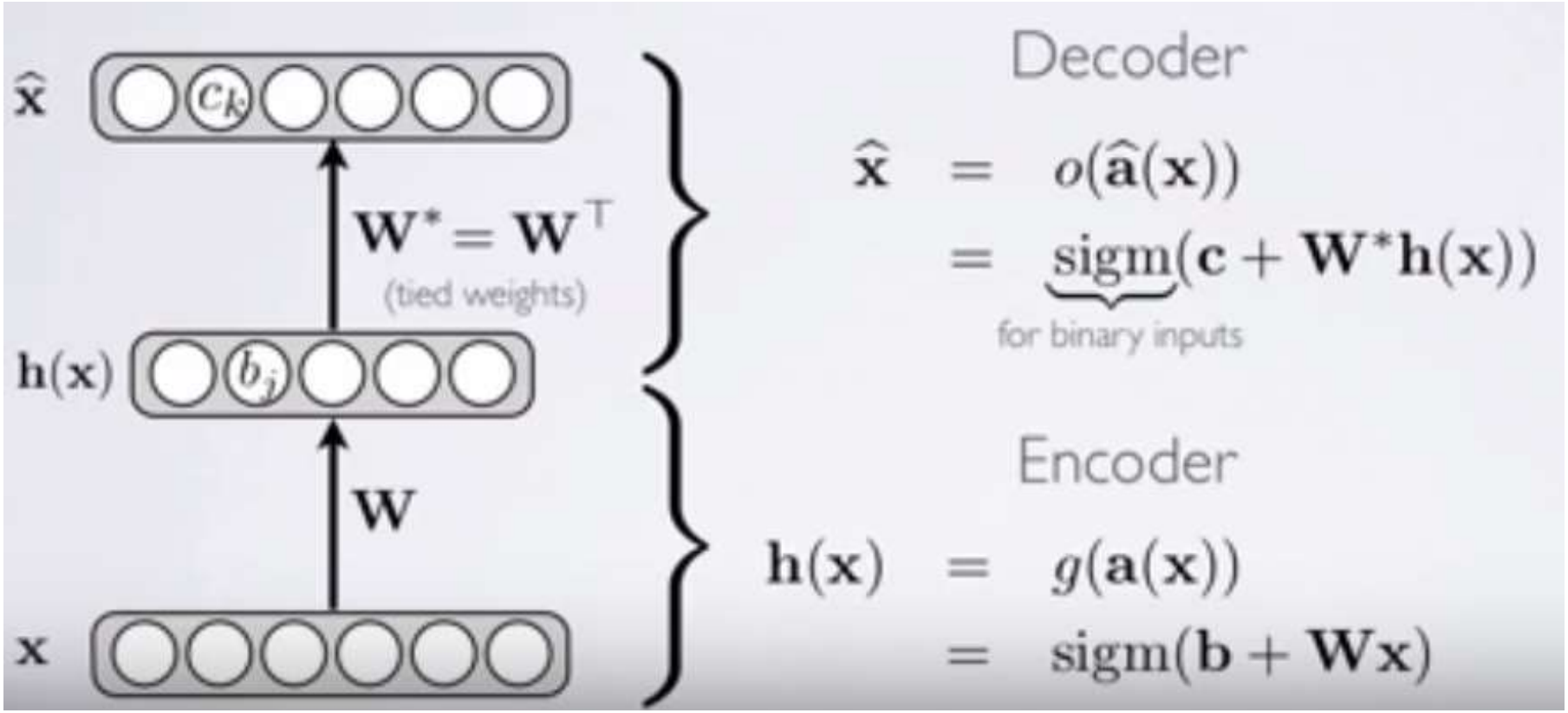
Auto-encoder Profundo
(*Deep Autoencoder*) - PCA

- Limitações:
  - Hipótese assumida: Direções das projeções com maior variância são as que melhor representam as características originais (válido apenas para colunas linearmente correlacionadas);
  - PCA é um método linear de redução de dimensionalidade: Só considera transformações ortogonais, logo, o método não trata de mapeamentos não-lineares;
  - Alterativa ao PCA: Extensões do método com transformações não lineares;
    - Autoencoders: Mapeamentos não-lineares ou empilhamento (representação hierárquica).

Autoencoder Empilhado
(*Stacked Autoencoder*)

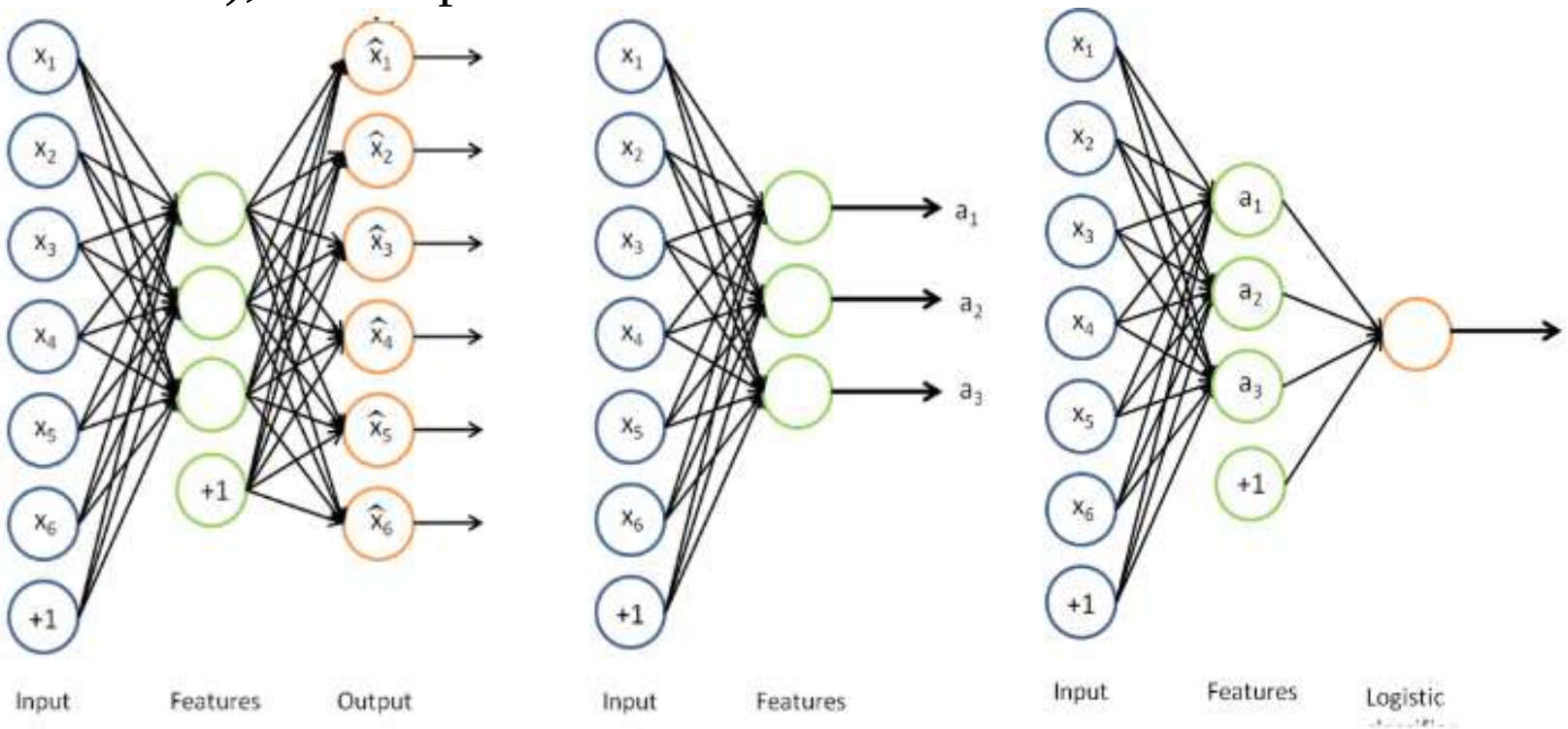
- Um tipo de modelo que tenta descobrir características genéricas dos dados:
  - Aprende a identificar funções através do aprendizado de subcaracterísticas;
  - Compressão: pode usar novas características como um novo conjunto de treinamento.
- Ao colocar uma camada escondida menor que a entrada, a rede é forçada a criar uma representação compacta do espaço de entrada.
- Função de minimização de erro: $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$

Autoencoder Empilhado
(*Stacked Autoencoder*)



Autoencoder Empilhado
(*Stacked Autoencoder*)

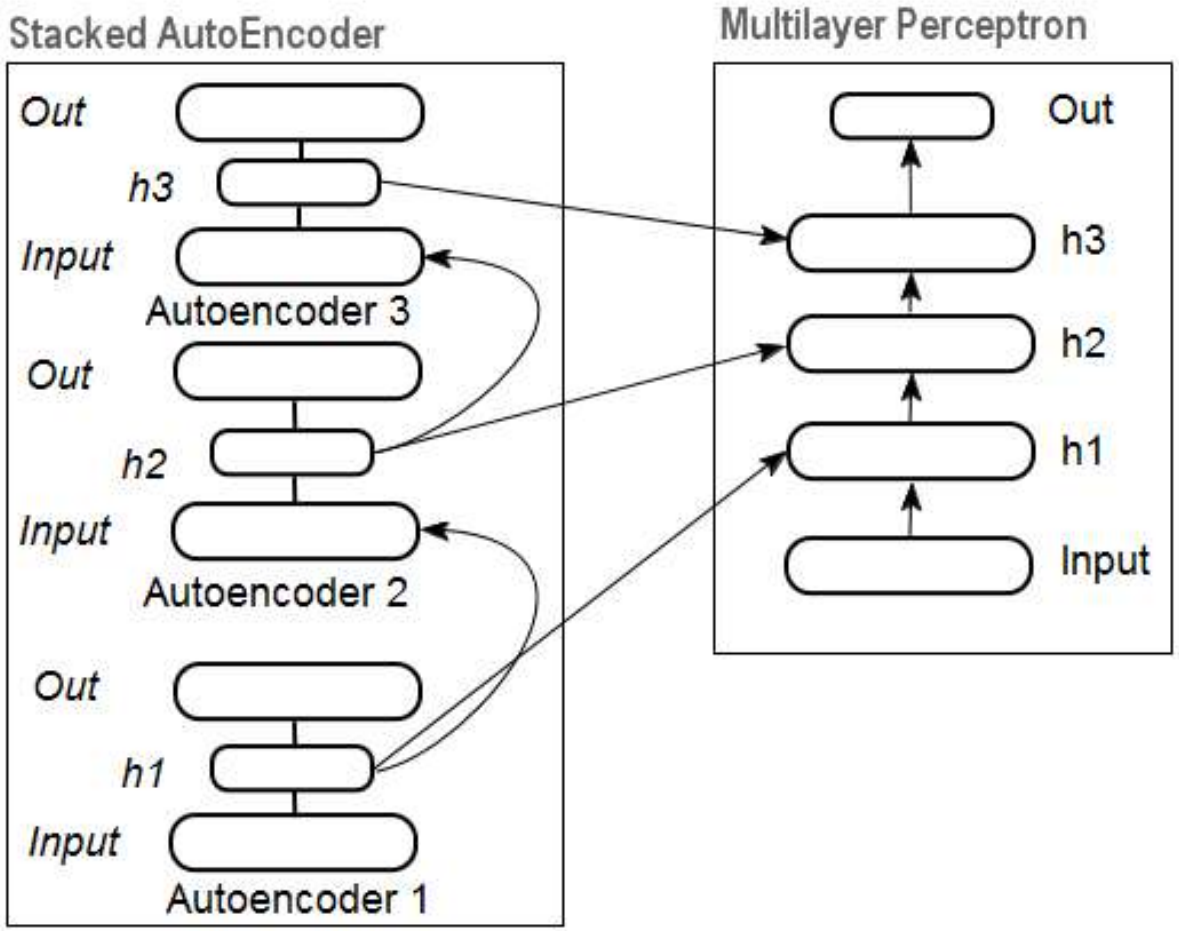
- Exemplo: autoencoder com 6 entradas e 3 nodos escondidos;
  - Exemplos de treinamento possuem 6 bits (cinco 0s e um 1);
  - Representa os exemplos em 3 bits, a_1, a_2 e a_3, (representação binaria), sem supervisão.



Autoencoder Empilhado
(*Stacked Autoencoder*)

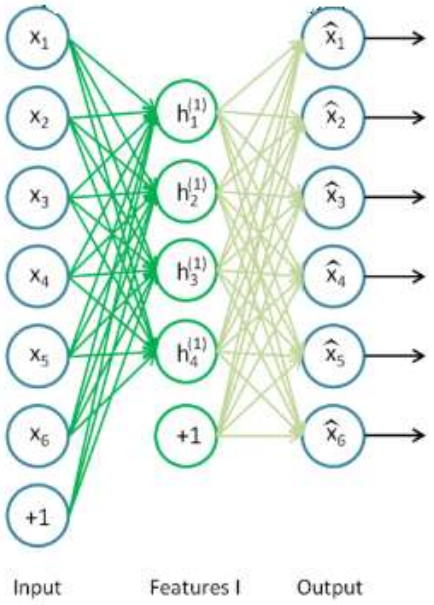
- O número de nodos na camada escondida deve ser **menor** que a dimensão de entrada para evitar que a rede aprenda a solução trivial, ou seja, simplesmente copiar a entrada;
- Com menos nodos para codificar a entrada, a rede é forçada a aprender uma representação compacta do espaço de entrada.

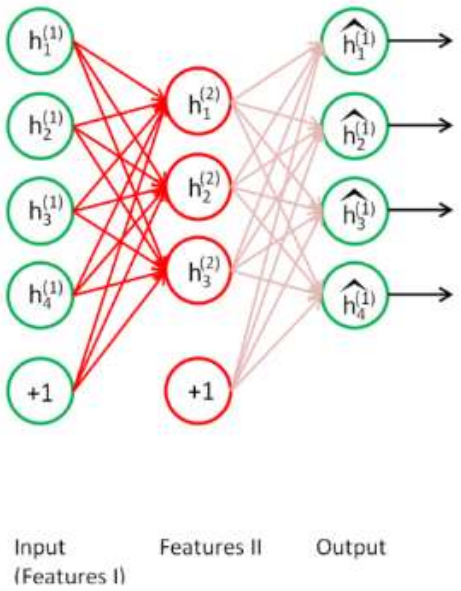
Autoencoder Empilhado
(*Stacked Autoencoder*)



Autoencoder Empilhado
(*Stacked Autoencoder*)

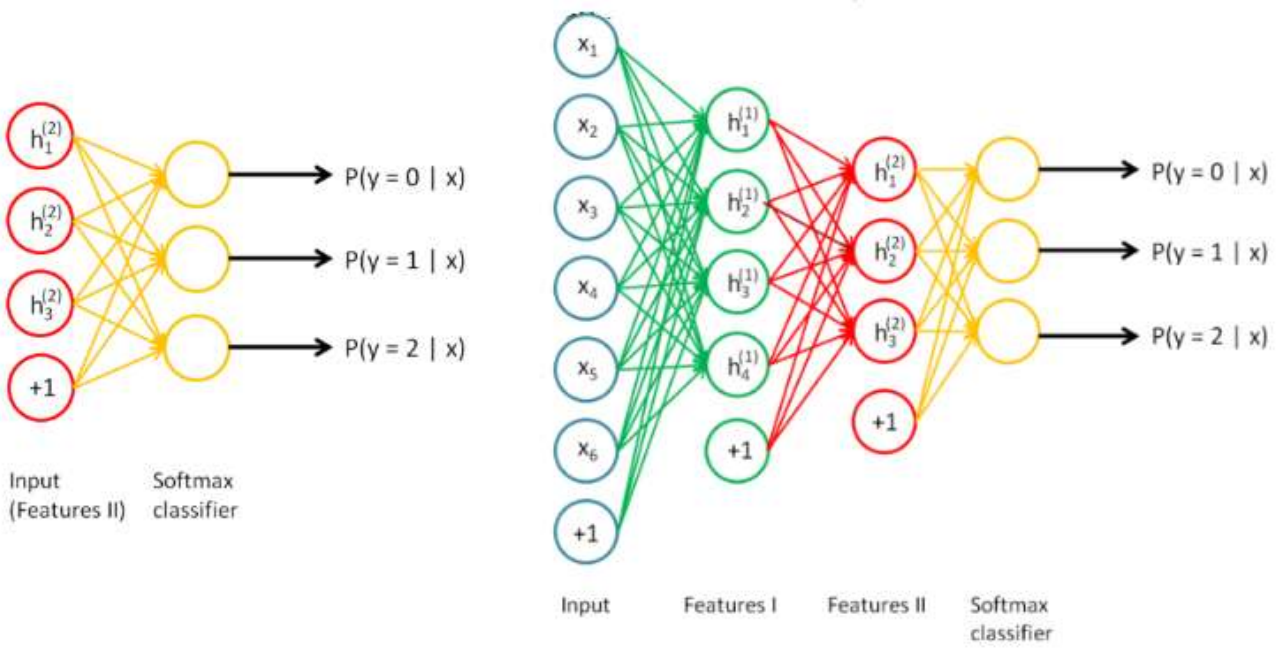
- (Bengio, 2007) Empilhar muitos auto-encoders esparsos e treinálos de forma gulosa;
- O código gerado por um é repassado como entrada para o seguinte;





Autoencoder Empilhado
(*Stacked Autoencoder*)

- Realizar treinamento supervisionado na última camada utilizando características finais;
- Treinamento supervisionado da rede como um todo para realizar ajustes finos dos pesos;



Autoencoder Empilhado

(*Stacked Autoencoder*) - Treinamento *Greedy Layer-Wise*

1. Treinar primeira camada utilizando dados sem rótulos;

2. Fixar os parâmetros livres da primeira camada e começar a treinar a segunda usando a saída da primeira camada como entrada não-supervisionada da segunda camada;

3. Repetir 1 e 2 de acordo com o número desejado de camadas;

4. Usar a saída da camada final como entrada para uma camada/modelo supervisionado e treinar de modo supervisionado com pesos anteriores fixos;

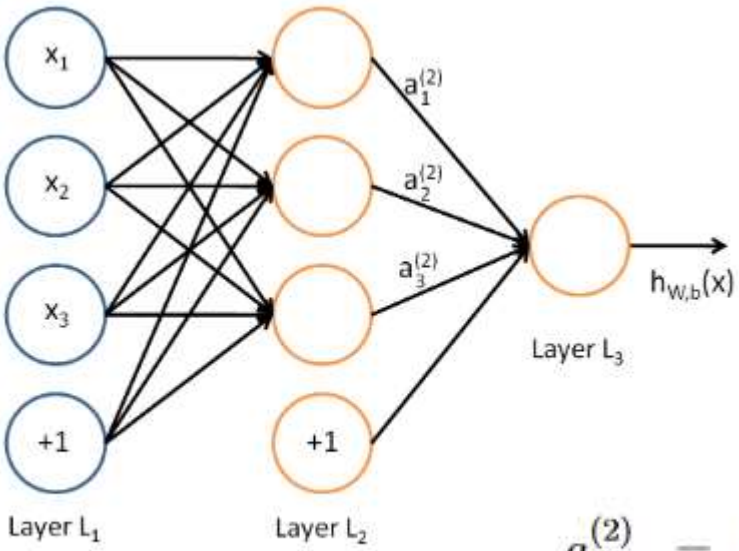
5. Liberar todos os pesos e realizar ajuste fino da rede como um todo utilizando uma abordagem supervisionada;

Autoencoder Empilhado
(*Stacked Autoencoder*) - Treinamento *Greedy Layer-Wise*

- Evita muitos dos problemas associados ao treinamento de uma rede profunda de modo totalmente supervisionado,
  - Cada camada foca apenas no processo de aprendizagem;
  - Pode tirar vantagem de dados não rotulados;
  - Quando a rede completa é treinada de modo supervisionado, os pesos já estão minimamente ajustados (evita mínimos locais);

Autoencoder Empilhado

(*Stacked Autoencoder*) - Treinamento *Greedy Layer-Wise*



$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

Autoencoder Empilhado
(*Stacked Autoencoder*) - Treinamento *Greedy Layer-Wise*

- Função de custo de erro supervisionado:

$$J(W, b; x, y) = \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 .$$

- Função de custo de erro não-supervisionado:

$$p(x) = \text{sigm}(c + W\text{sigm}(b + W'x))$$

$$R = -\sum_i x_i \log p_i(x) + (1 - x_i) \log(1 - p_i(x))$$

Autoencoder Empilhado
(*Stacked Autoencoder*) - Treinamento *Greedy Layer-Wise*

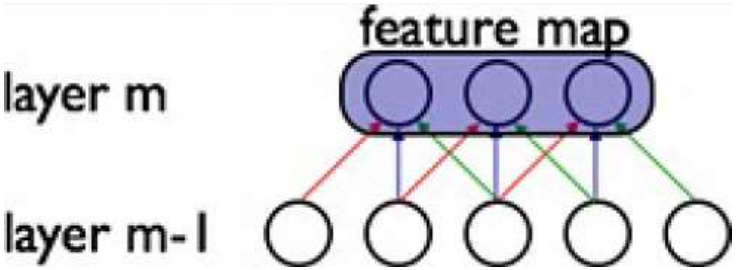
- Se o *Stacked autoencoder* completo for treinado com retropropagação há grande chance de ocorrer o problema dos gradientes diluídos;
- O treinamento pode se tornar demorado ou mesmo inviável;
- A solução está em treinar cada camada como um *autoencoder* isolado, o que reduz o caminho do gradiente para uma única camada;

Autoencoder Espaçado
(*Spaced Autoencoder*)

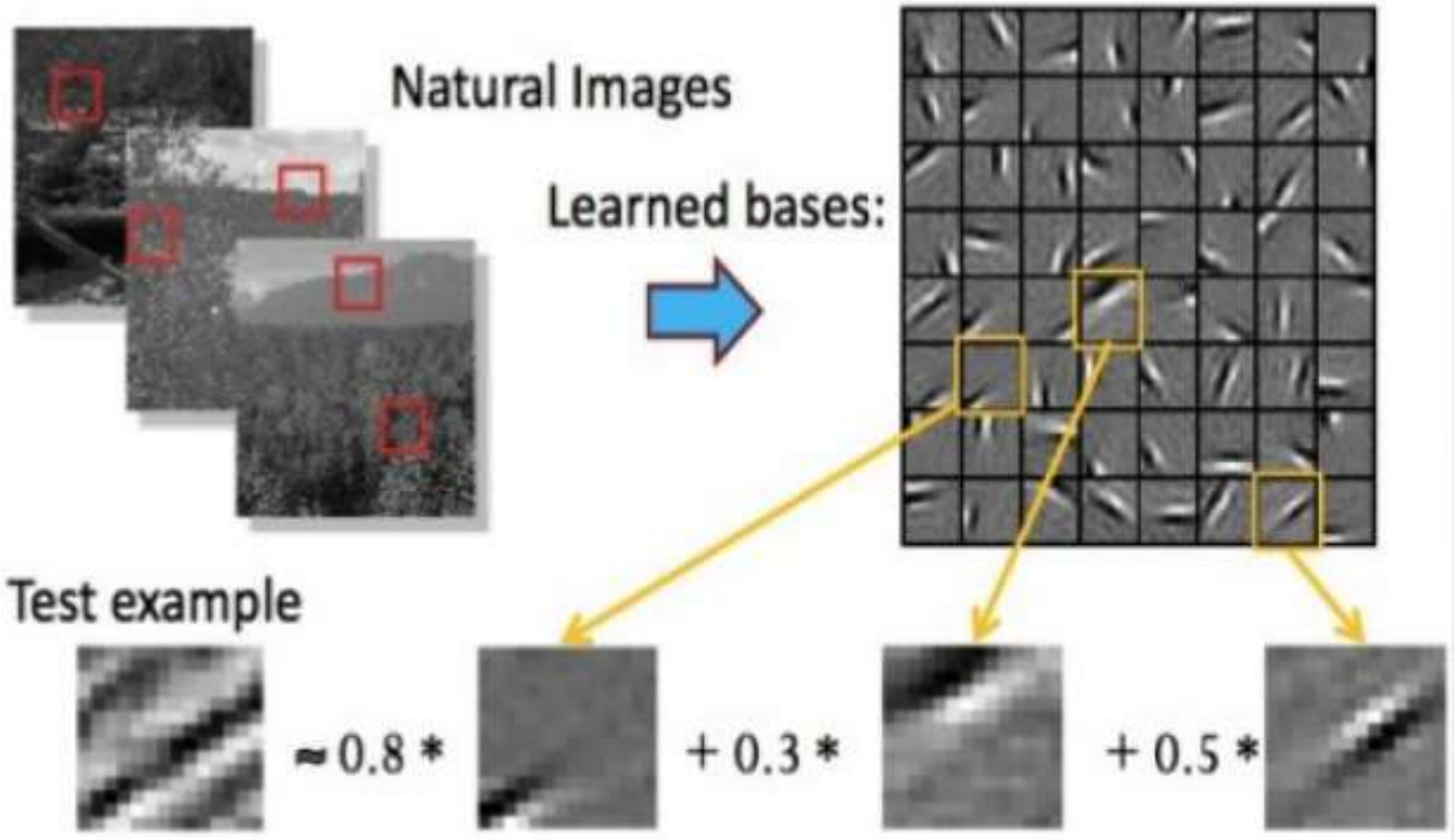
- Outra possibilidade de *Autoencoders* está em restringir as ativações das camadas escondidas;
  - Modifica-se a função de custo da rede para incluir um termo que penaliza muitas ativações nos nodos.
- Como resultado, uma codificação esparsa e aprendida (poucos nodos ativados por vez), mesmo que uma dada camada escondida seja maior que a dimensionalidade de entrada.
- $\Omega(\boldsymbol{h})$ – penalização por esparsidade.
- Função de minimização de erro: $L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h})$

Autoencoder Convolucional
(*Convolutional Autoencoder*)

- Emprega compartilhamento de pesos para reduzir o numero de parâmetros no *autoencoder* e ao mesmo tempo explorar conhecimento prévio dos problemas:
  - Por exemplo, pode-se trabalhar com blocos de tamanho 32x32 utilizando pequenas regiões 3x3 onde todas utilizam os mesmos pesos, efetivamente reduzindo a dimensão de entrada de 1024 para 9.



Autoencoder Convolucional
(*Convolutional Autoencoder*)



Autoencoder Denoising Empilhado
(*Stacked Denoising Autoencoder*)

- O *Denoising Autoencoder* (DAE) é uma versão estocástica do *Autoencoder* que recebe valores corrompidos em sua entrada e visa gerar representação latente dos dados não corrompidos:
  - Este modelo lida com o risco de função de identidade corrompendo aleatoriamente a entrada pela adição de ruído para que o *Autoencoder* reconstrua o sinal ou elimine seu ruído.
- Em Vincent et. al., 2010 é proposta a arquitetura do *Stacked Denoising Autoencoder* (SDA).
- *Denoising Autoencoders* podem ser empilhados através da alimentação da representação latente do DAE da camada anterior, servindo de entrada para a camada atual.
  - Função de minimização de erro: $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$

Ferramentas

Software	Creator	Platform	Written in	Interface
Apache MXNet	Apache Software Foundation	Linux, macOS, Windows,AWS, Android,iOS, JavaScript	Small C++core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl
Apache SINGA	Apache Incubator	Linux, macOS, Windows	C++	Python, C++, Java
Caffe	Berkeley Vision and Learning Center	Linux, macOS, Windows	C++	Python, MATLAB, C++
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python(Keras), Kotlin
Intel Data Analytics Acceleration Library	Intel	Linux, macOS, Windows on Intel CPU	C++, Python, Java	C++, Python, Java[10]
Keras	François Chollet	Linux, macOS, Windows	Python	Python, R
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan	Linux, macOS	Python, C, CUDA	Python
TensorFlow	Google Brainteam	Linux, macOS, Windows,Android	C++, Python, CUDA	Python (Keras), C/C++, Java, Go, R, Julia
Theano	Université de Montréal	Cross-platform	Python	Python (Keras)
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	Linux, macOS, Windows,Android, iOS	C, Lua	Lua, LuaJIT,C, utility library for C++/OpenCL

Referências

- Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H., (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19: 153.
- Deep Learning Tutorial. LISA Lab, University of Montreal.
- Du, K.-L. & Swamy M. N. S. (2019). *Neural Networks and Statistical Learning*. Springer, 2nd edition.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61: 85-117.
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A, (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 11: 3371-3408.
- https://cs.nju.edu.cn/wujx/paper/CNN.pdf
- https://arxiv.org/pdf/1812.05069.pdf