Autoencoders

Pedro Jorge Oliveira Câmara

Julho 2024

Resumo

Autoencoder é uma rede neural que tem como objetivo receber uma entrada x e retornar como saída uma reconstrução $x' \approx x$. Idealmente, a entrada é transformada em uma representação z de menor dimensão e, a partir dela, reconstruída, de maneira a tentar recuperar o dado original com a menor perda de informação possível. Neste trabalho, são exploradas a definição, a relação com a decomposição em valores singulares e a construção dos autoencoders. Posteriormente, são feitos alguns experimentos e comentários sobre sua aplicação no contexto de remoção de ruídos em sinais.

1 Introdução

Seja x um vetor qualquer. Um autoencoder possui a especificação:

Entrada: x

Saída: x', tal que $x' \approx x$

Seu objetivo é, dada uma entrada x, retornar uma reconstrução x'. A estrutura geral de um autoencoder é definida por duas partes, um encoder e e um decoder d:

Encoder $e(x)$**:** transforma x em um vetor z, idealmente de dimensão menor

Decoder $d(z)$**:** transforma z em uma reconstrução x', de maneira que $x' \approx x$

Na estrutura de um autoencoder, queremos que z tenha dimensão menor do que x para que ele possa, de alguma maneira, encontrar um z que melhor represente x. Esse vetor é chamado *representação*, *vetor latente*, *codificação* ou *embedding*, e é um elemento do *espaço latente*, o espaço de dimensão menor que melhor representa os dados de entrada. Dessa maneira, temos uma entrada x e uma saída $x' = d(e(x))$.

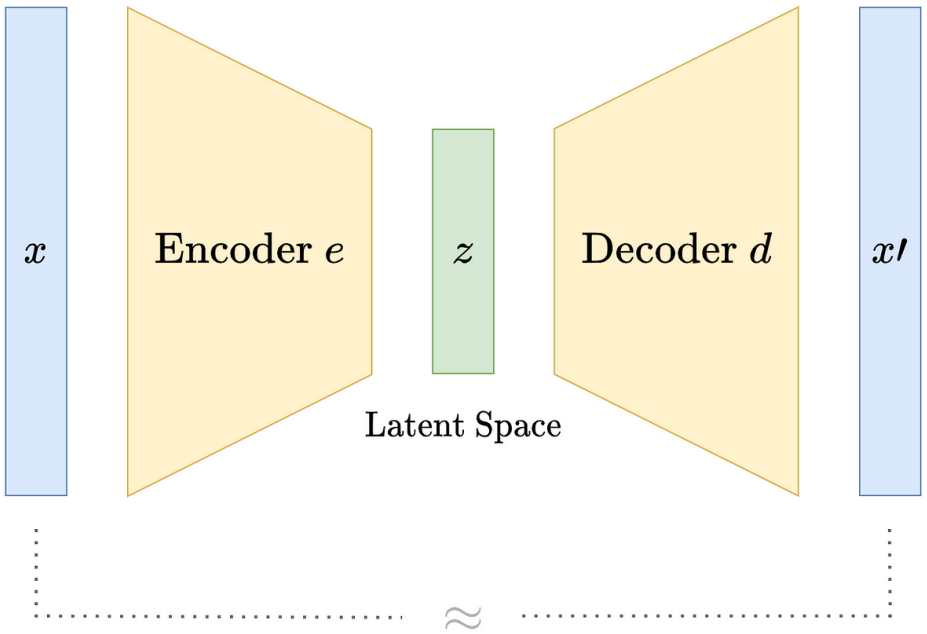


Figura 1: Estrutura de um autoencoder

Como, por definição, $z = e(x)$ está em um espaço de dimensão menor do que x, a reconstrução $x' = d(z)$ inevitavelmente terá alguma perda de informação. Para mensurar essa perda, definimos alguma função $L(x - d(e(x)))$, que irá quantificar o quão diferente a reconstrução $x' = d(e(x))$ é da entrada x.

Com isso, a construção de um autoencoder consiste definir uma função de custo L e encontrar as transformações encoder e e decoder d que a minimizem:

$$\arg\min_{d,\ e} L(x - d(e(x)))$$

2 Autoencoder Linear

O autoencoder mais simples possível é aquele que possui o encoder e e o decoder d como transformações lineares tal que elas compartilhem uma mesma matriz W:

$$e(x) = Wx = z$$
$$d(z) = W^T z = x'$$
$$d(e(x)) = W^T W x = x'$$

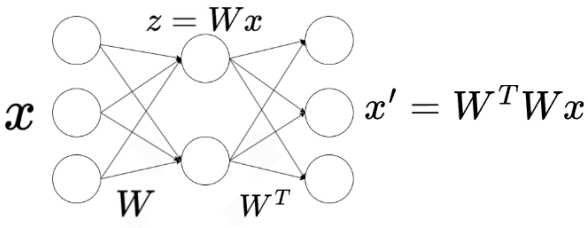


Figura 2: Estrutura de um autoencoder linear

Com a função de custo dada por:

$$L(x - d(e(x))) = ||x - W^T W x||_2^2$$

Seja X uma matriz que represente algum dado com n amostras e x_i os seus vetores. A função de custo será a norma de Frobenius:

$$L(x - d(e(x))) = \sum_{i=1}^{n} ||x_i - W^T W x_i||_2^2 = ||X - W^T W X||_F^2$$

O objetivo do autoencoder é encontrar as transformações encoder $e(x) = Wx = z$ e decoder $d(z) = W^T z = x'$ que minimizem esse erro de reconstrução:

$$\arg\min_W \sum_{i=1}^{n} ||x_i - W^T W x_i||_2^2$$

É possível demonstrar* que W irá convergir para a matriz U^T da decomposição em valores singulares de X:

$$U^T = \arg\min_W \sum_{i=1}^{n} ||X - W^T W X||_F^2$$

*Teorema extraído dos artigos:

- J. Karhunen e J. Joutsensalo, "Generalizations of principal component analysis, optimization problems, and neural networks"
- P. Baldi e K. Hornik, "Neural networks and principal components analysis: Learning from examples without local minima"
- Elad Plaut, "From Principal Subspaces to Principal Components with Linear Autoencoders"
- H. Bourlard e Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition"

2.1 Exemplo

Suponha $X_{3\times3}$ uma matriz de posto 3. Queremos construir um autoencoder linear que transforme cada $x_i \in \mathbb{R}^3$ em um vetor $Wx_i = z_i \in \mathbb{R}^2$ e que faça uma reconstrução $x_i' = W^T z_i$. Teremos que $W_{2\times3}$ e $W_{3\times2}^T$ de posto 2:

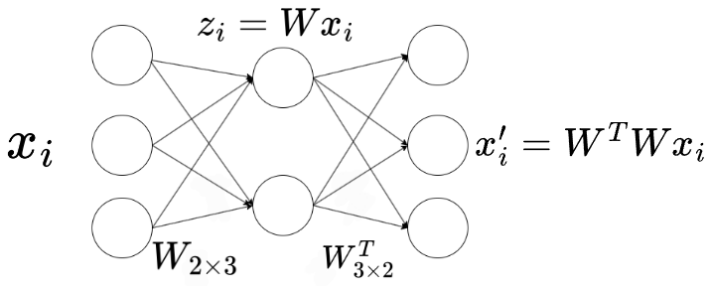


Figura 3: Estrutura de um autoencoder linear de $\mathbb{R}^3$ para $\mathbb{R}^2$

Um autoencoder linear que receba como entrada uma matriz X e tenha um espaço latente de dimensão k, na verdade converge para encontrar o mesmo subespaço gerado pela reconstrução $U_k \Sigma_k V_k^T$ de posto k do SVD de X, e está somente projetando os vetores de X nesse subespaço:

$$X' = W^T W X = U_k U_k^T X$$

Dessa forma, durante o treinamento, os vetores $W^T_{3\times 2}$ convergem para gerarem o mesmo plano que U_2, fazendo com que a reconstrução de um vetor x_i de X seja apenas uma projeção de $\mathbb{R}^3$ para $\mathbb{R}^2$.

3 Não linearidade

O autoencoder linear é, portanto, análogo à decomposição em valores singulares. A sua capacidade está, no entanto, na possibilidade de aplicação de transformações não lineares na entrada x, chamadas de *funções de ativação*.

Cada amostra x_i, no processo de encoder, passará não somente por uma transformação linear $W_1 x_i$, mas sofrerá também a aplicação de uma função não linear f de ativação, de maneira a tornar o vetor latente $z_i = f(W_1 x_i)$. Da mesma forma, a reconstrução a partir de cada z_i, no decoder, será não somente $W_2 z_i$, mas também a aplicação de uma função não linear $g(W_2 z_i) = x'_i$.

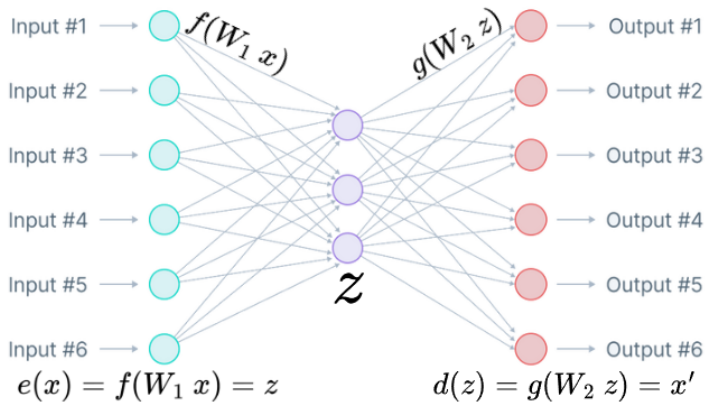


Figura 4: Autoencoder com funções de ativação

Nos autoencoders lineares, a adição de novas camadas não faria diferença: possuir transformações $e(x) = W_3W_2W_1x$ como um encoder e, por associatividade, é o mesmo que possuir uma única transformação linear $e(x) = Ax$, para $A = W_3W_2W_1$. A adição da não linearidade passa a tornar possível o aumento no número de camadas e a aplicação de funções de ativação a cada uma delas.

Podemos, então, definir, por exemplo, o encoder como uma sequência de três funções de ativação, $e(x) = f_3(W_3f_2(W_2f_1(W_1x)))$, assim como para o decoder, e eles não precisam necessariamente ser simétricos.

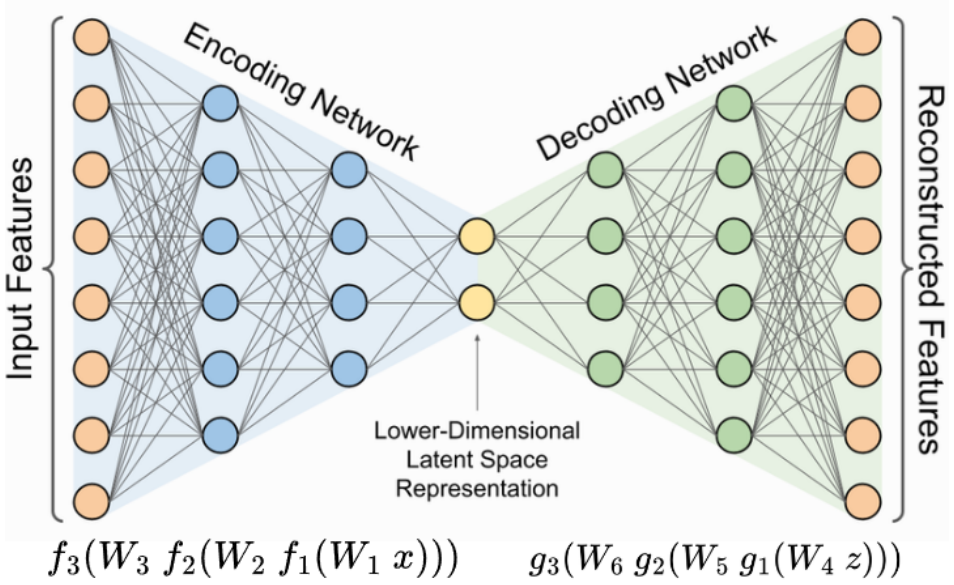


Figura 5: Autoencoder com uma sequência de funções de ativação

Um problema inerente dessa possibilidade é a hiperparametrização: a quantidade de camadas, quantidade de neurônios por camadas e as funções de ativação não são aprendidas pelo autoencoder; elas devem ser passadas como parâmetro para sua construção. Isso faz com que encontrar valores ideais desses hiperparâmetros, que tornem o desempenho da rede neural o melhor possível, seja um desafio por si só.

4 Experimentos

Uma das possíveis aplicações dos autoencoders é a remoção de ruídos. Podemos construir um autoencoder e treiná-lo com dados X de alguma natureza, de maneira que ele aprenda a representá-los em um espaço latente Z e, a partir dele, reconstrua dados $X' \approx X$. Após a etapa de treinamento, é possível que ele, a partir de uma nova amostra x, poluída com algum ruído, como entrada, reconstrua uma aproximação do dado original sem ruidosidade.

São feitas nesta seção algumas experimentações de construção de autoencoders com diferentes funções de ativação, sobre sua funcionabilidade no contexto de remoção de ruídos em sinais digitais e comparações com o autoencoder linear e a decomposição em valores singulares.

Para o caso dos autoencoders não lineares e linear, eles são construídos a partir de uma mesma arquitetura de quantidade camadas, quantidade de neurônios e dimensão k do espaço latente, descrita mais adiante. Para o SVD, é feita a decomposição $X = U\Sigma V^T$ na matriz dos dados originais, sem ruído, e, para cada nova amostra x ruidosa, a reconstrução x' é encontrada a partir da projeção $x' = U_k U_k^T x$. Idealmente, os resultados encontrados pelo autoencoder linear e o SVD devem ser similares.

4.1 Sinais digitais

Um sinal é um conjunto de dados observados de algum fenômeno, geralmente ao longo do tempo, que pode representar alguma informação.

Apesar de, geralmente, serem contínuos, precisamos capturar os valores de um determinado sinal ao longo de intervalos de tempo discretos, conforme descrito na figura.

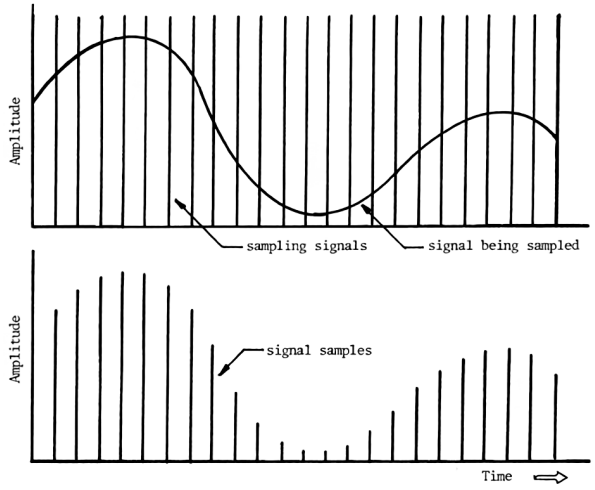


Figura 6: Exemplo de amostras de um sinal

4.2 Construção dos dados

Cada vetor x_i é um sinal, constituído de 100 amostragens, cada uma correspondendo ao valor $f_i(t)$ do sinal f_i no tempo t:

$$x_i = \begin{bmatrix} f_i(t_1) \\ f_i(t_2) \\ \vdots \\ f_i(t_{100}) \end{bmatrix}$$

Onde $t_j = \frac{j}{10}\ \forall j \in \{0, 1, 2, ..., 99\}$, , ou seja, $t \in \{0, 0.1, ..., 9.8, 9.9\}$.

Os dados formados por esses sinais constituem uma matriz X construída da forma

$$X_{100\times 20.000} = \begin{bmatrix} f_1(t_1) & f_2(t_1) & \dots & f_{20.000}(t_1) \\ f_1(t_2) & f_2(t_2) & \dots & f_{20.000}(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(t_{100}) & f_2(t_{100}) & \dots & f_{20.000}(t_{100}) \end{bmatrix}$$

Cada função $f_i(t)$, que representa um sinal, é uma amostra do conjunto:

$$f_i(t) \in \left\{ \begin{array}{l} sin(rt),\ cos(rt), \\ sin(rt) + cos(rt), \\ sin(rt) - cos(rt), \\ sin(rt) \cdot cos(rt) \end{array} \right\}$$

Onde r é uma amostra da distribuição normal com média 0 e desvio padrão 2.25:

$$r \sim \mathcal{N}(0,\ 2.25)$$

Quando $r = 1$, podemos visualizar alguns dos sinais potencialmente presentes na matriz X:

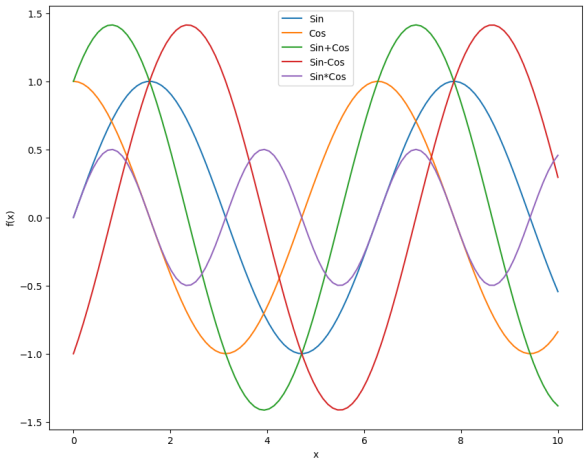


Figura 7: Possíveis sinais da matriz X

4.3 Construção dos autoencoders

Os autoencoders construídos diferenciam-se apenas pelas funções de ativação utilizadas em cada um. Toda a estrutura restante é mantida igual para todos:

Entrada: $x \in \mathbb{R}^{100}$, um sinal

Espaço latente: $z \in \mathbb{R}$ ou $z \in \mathbb{R}^5$, a depender do experimento

Saída: $x' \in \mathbb{R}^{100}$, sinal reconstruído

Camadas de encoder:

· $100 \to 50 \to 25 \to 10 \to 5 \to 1$, quando $z \in \mathbb{R}$

· $100 \to 50 \to 25 \to 10 \to 5$, quando $z \in \mathbb{R}^5$

Camadas de decoder:

· $1 \to 5 \to 10 \to 25 \to 50 \to 100$, quando $z \in \mathbb{R}$

· $5 \to 10 \to 25 \to 50 \to 100$, quando $z \in \mathbb{R}^5$

Funções de ativação - Entre todas as camadas há a mesma função de ativação:

Função de ativação	**Fórmula**
Identity(x) (Autoencoder Linear)	x
ReLU(x)	$\max(0, x)$
Sigmoid(x)	$\frac{1}{1+e^{-x}}$
Tanh(x)	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$
LeakyReLU(x)	$\max(0, x) + \alpha \cdot \min(0, x)$
PReLU(x)	$\max(0, x) + \alpha \cdot \min(0, x)$
GELU(x)	$x \cdot \Phi(x)$

Tabela 1: Funções de ativação utilizadas nos autoencoders

OBS: As funções LeakyReLU(x) e PReLU(x) compartilham a mesma fórmula, mas em uma o parâmetro α é passado e mantido constante, enquanto que na outra ele é atualizado internamente conforme o treinamento, respectivamente.

OBS2: A função $\Phi(x)$ na GELU(x) é a Função de Distribuição Acumulada da Distribuição Gaussiana.

OBS3: A função de ativação Identity(x) não aplica nenhuma transformação, apenas retorna a entrada. Ela é usada para construir o autoencoder linear.

4.4 Experimentação: $sin(x)$ ruidoso

São geradas 10 amostras da função $sin(x)$ em 100 pontos no intervalo $[0, 10)$, espaçados igualmente por um *step* de 0.1. A cada ponto, é somada ao valor $sin(x)$ uma perturbação aleatória $l \cdot p$, onde $p \sim \mathcal{N}(0, 1)$ e l é o quão forte ela é. Para baixos níveis de ruído, $l = 0.1$; para altos níveis, $l = 1$.

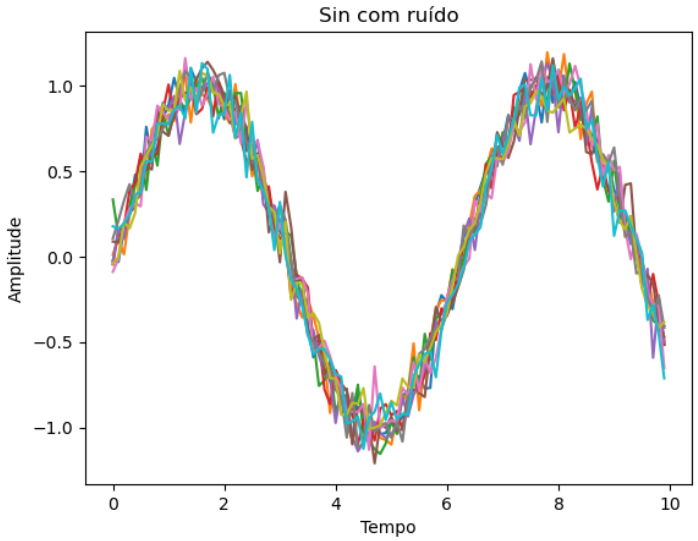
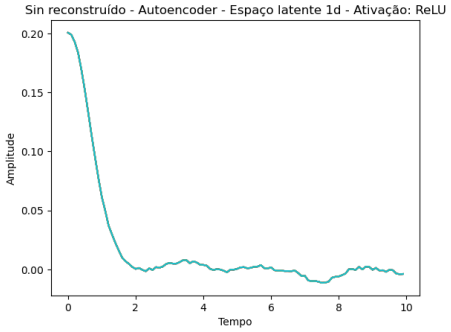


Figura 8: $sin(x) + 0.1p$

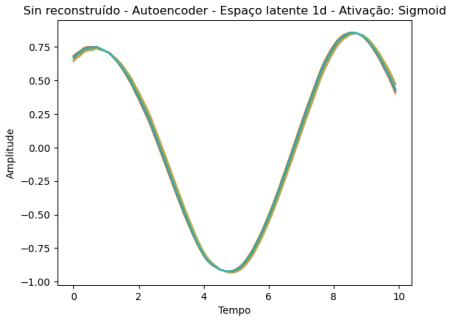
4.4.1 Espaço latente em 1 dimensão

Cada autoencoder recebe como entrada cada uma das amostras ruidosas, mapeia para o espaço latente z de 1 dimensão, e tenta reconstruir o sinal original. O SVD projeta cada sinal ruidoso em um subespaço de dimensão 1. Os gráficos abaixo ilustram as reconstruções dos autoencoders e do SVD de cada um deles.

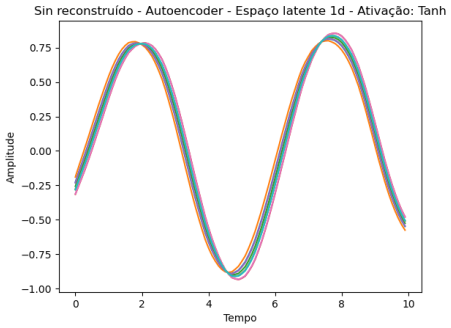
Devido ao fato de que $z \in \mathbb{R}$, os resultados não são satisfatórios e, em alguns casos, extremamente distantes de uma reconstrução ideal, como nos casos do autoencoder linear, o SVD e do que utiliza ReLU. O mais próximo de uma reconstrução razoável, que remete, de fato, à função seno, é a alcançada pelo uso da Tanh como função de ativação.



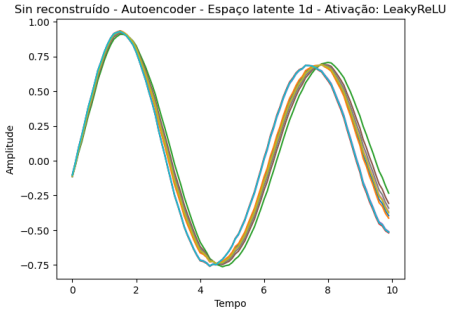
(a) ReLU



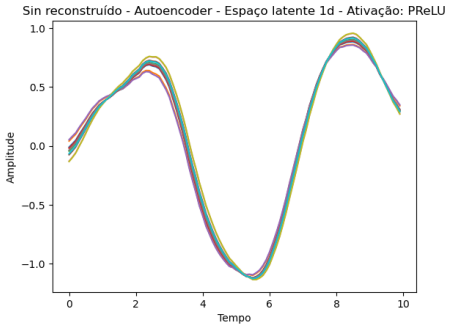
(b) Sigmoid



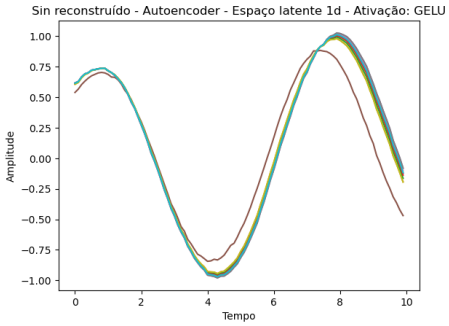
(c) Tanh



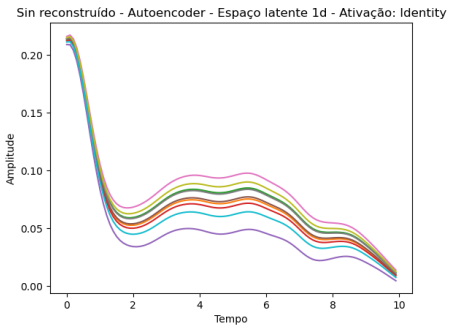
(d) LeakyReLU



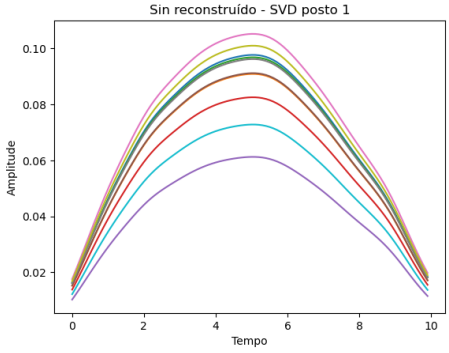
(e) PReLU



(f) GELU



(g) Identity

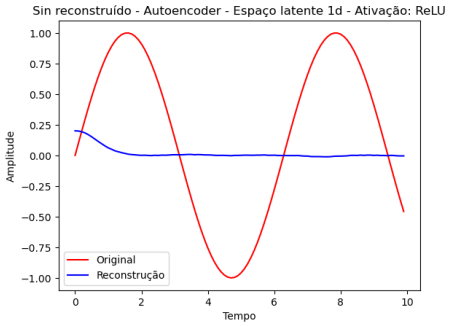


(h) SVD

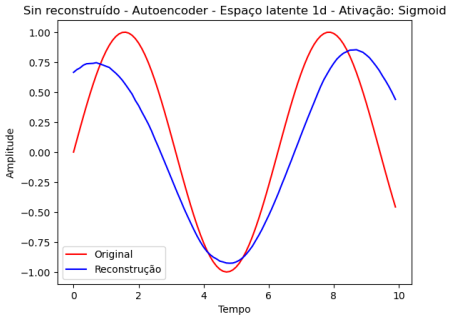
Figura 9: Reconstruções de sinais $sin(x)$ ruidosos

Uma observação é que os gráficos do autoencoder linear e do SVD, embora ambos insatisfatórios, se diferem.

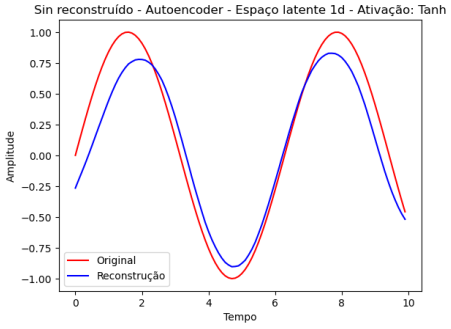
A partir dessas saídas, é calculada a média dos vetores reconstruídos, de maneira a tentar obter o sinal original:



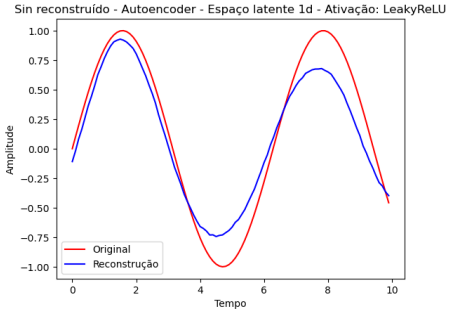
(a) ReLU



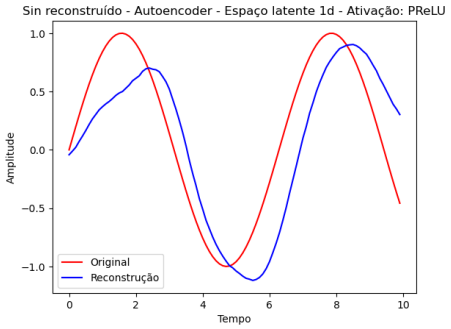
(b) Sigmoid



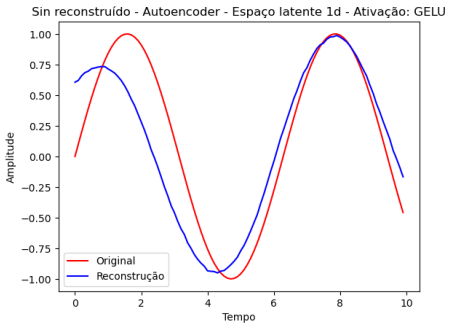
(c) Tanh



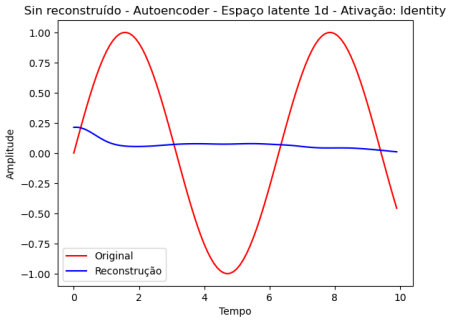
(d) LeakyReLU



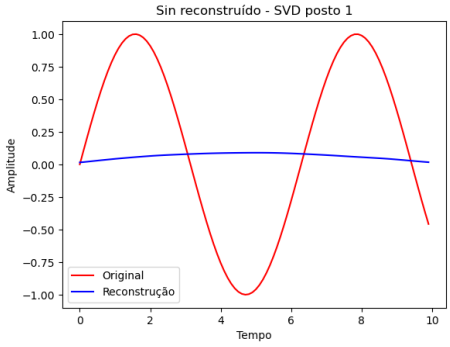
(e) PReLU



(f) GELU



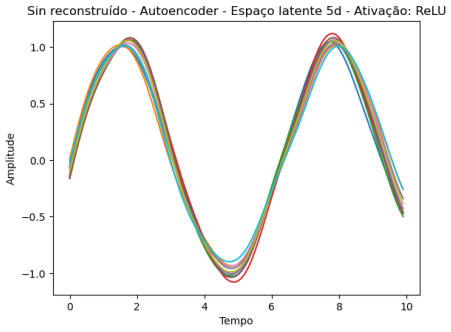
(g) Identity



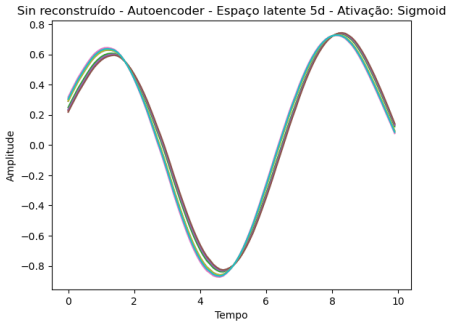
(h) SVD

Figura 10: Reconstrução do sinal $sin(x)$ original

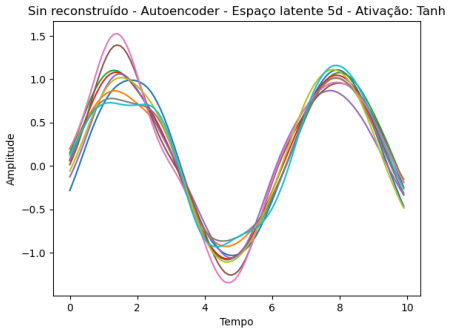
4.4.2 Espaço latente em 5 dimensões



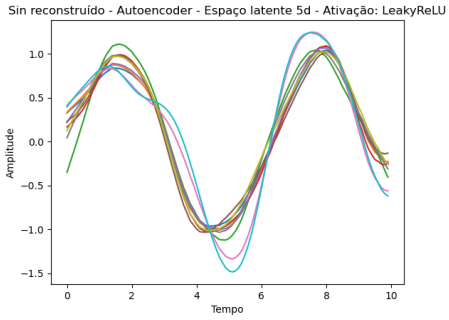
(a) ReLU



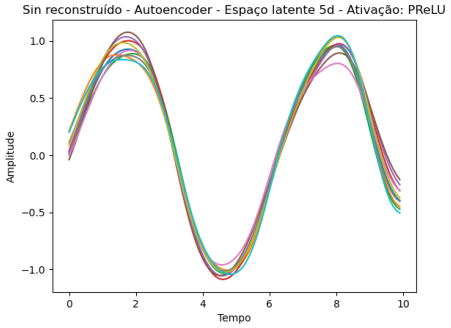
(b) Sigmoid



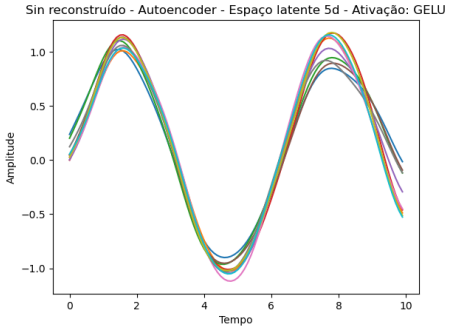
(c) Tanh



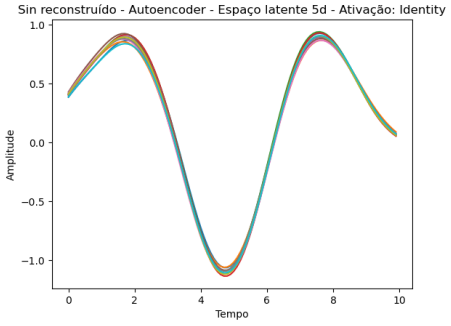
(d) LeakyReLU



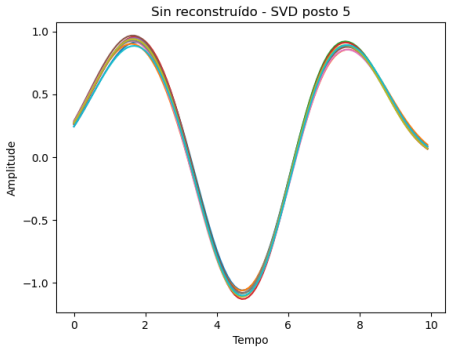
(e) PReLU



(f) GELU



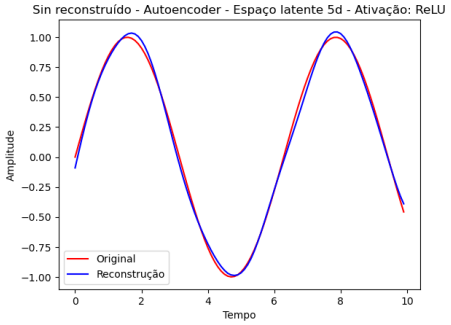
(g) Identity



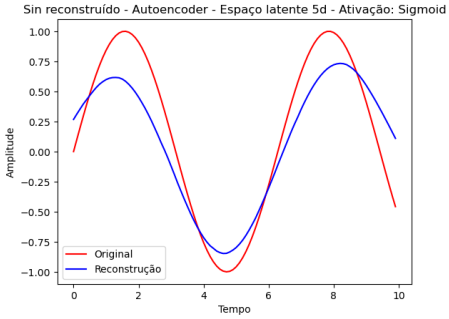
(h) SVD

Figura 11: Reconstruções de sinais $sin(x)$ ruidosos

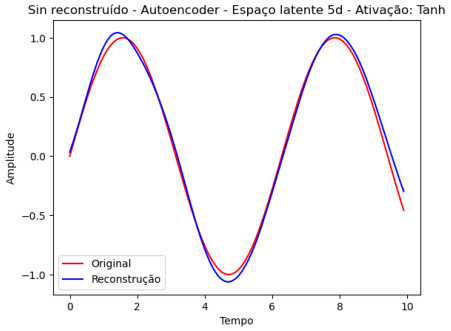
Calculando a média e a reconstruindo o sinal original:



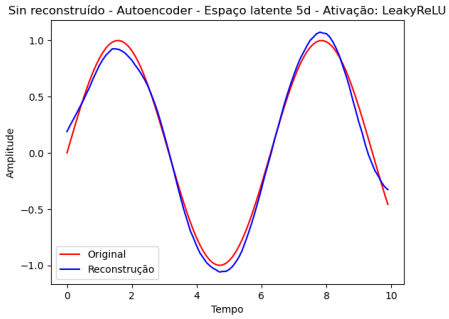
(a) ReLU



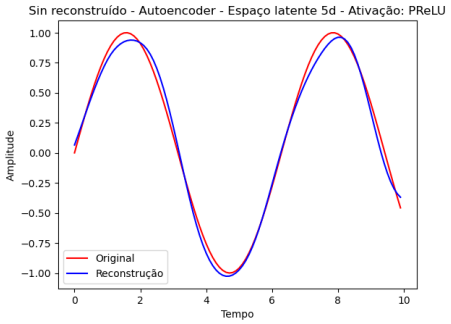
(b) Sigmoid



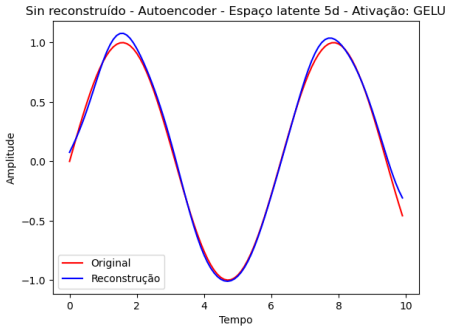
(c) Tanh



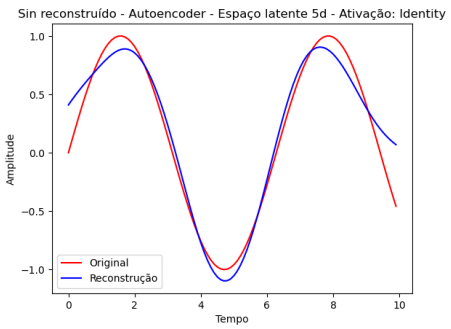
(d) LeakyReLU



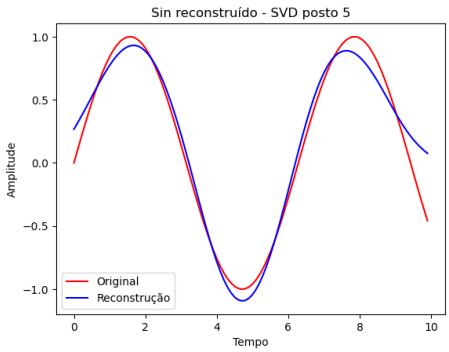
(e) PReLU



(f) GELU



(g) Identity



(h) SVD

Figura 12: Reconstruções de sinais $sin(x)$ ruidosos

Aumentando a capacidade dos autoencoders, com um espaço latente de 5 dimensões, e do SVD, permitindo uma reconstrução com U_5, há uma melhora nas aproximações do sinal original. Como esperado, com espaço latente e posto 5, as reconstruções do

autoencoder linear e do SVD são extremamente semelhantes. Há algumas diferenças sutis, possivelmente devido ao treinamento do autoencoder, que "converge para U_5", enquanto o SVD já possui a matriz exata.

4.5 Experimentação: $sin(x)$ com alto nível de ruído

Permitindo que $l = 1$, aumentamos a perturbação da função $sin(x) + l \cdot p$ e temos amostras com um alto nível de ruído. Como já sabemos que as reconstruções com dimensão e posto 1 são insatisfatórias, são feitas reconstruções apenas considerando um espaço latente em $\mathbb{R}^5$.

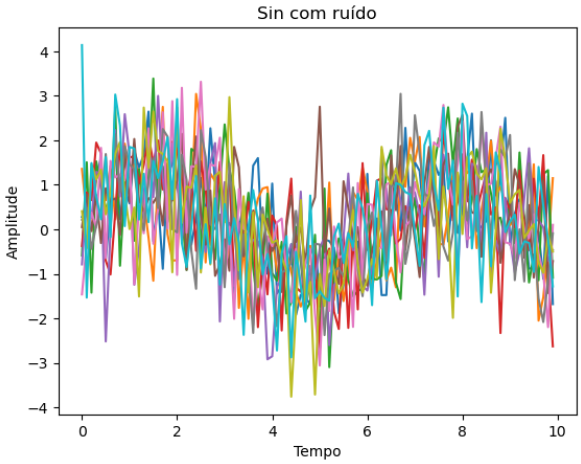
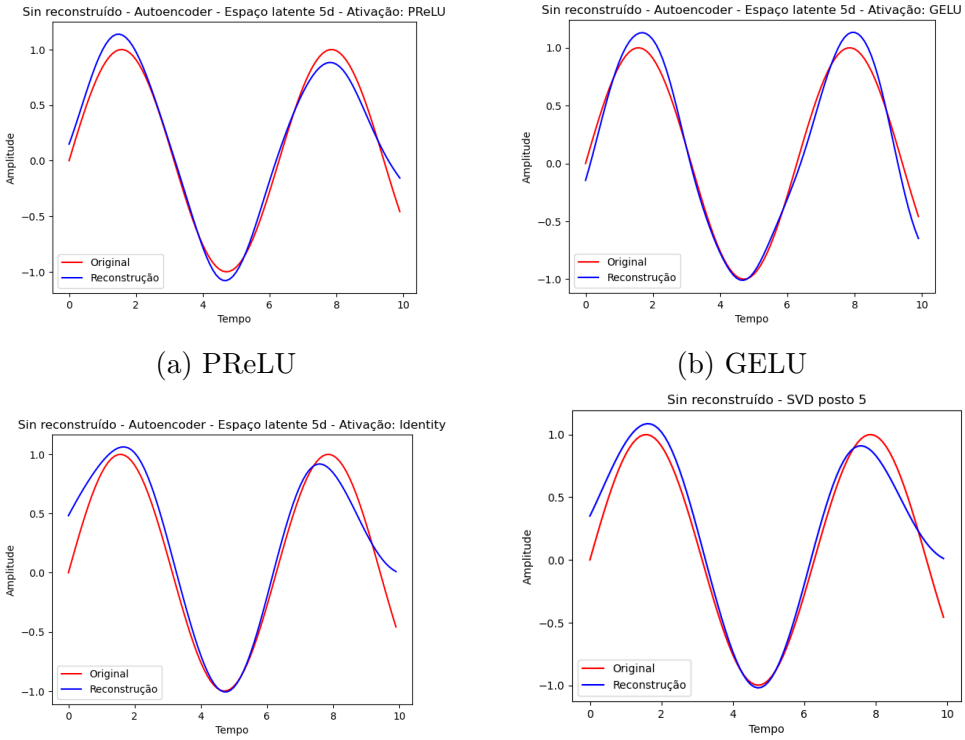


Figura 13: $sin(x) + p$

4.5.1 Espaço latente em 5 dimensões

São mostradas as melhores reconstruções médias dos autoencoders não lineares, PReLU e GELU, e as do autoencoder linear e do SVD.

Considerando o alto nível de ruído das amostras, os resultados são extremamente satisfatórios, até mesmo para os modelos lineares, que conseguem projetar em seus subespaços uma reconstrução x' realmente próxima do sinal original.



(a) PReLU

(b) GELU

(c) Identity

(d) SVD

Figura 14: Reconstruções de sinais $sin(x)$ ruidosos

5 Conclusão

Autoencoders são uma abordagem poderosa de redução de dimensão e podem superar a performance da decomposição em valores singulares devido à sua capacidade de aplicação de não linearidade, permitindo que o modelo consiga uma maior generalização, para além dos subespaços. No contexto de remoção de ruídos, eles podem ser treinados com dados limpos e potencialmente remover perturbações em uma nova amostra ruidosa.

Apesar de sua performance, há algumas desvantagens e dificuldades, ligadas especialmente aos problemas comuns das redes neurais:

- **Muitos hiperparâmetros**
  - Quantas camadas usar?
  - Quantos neurônios usar por camada?
  - Quais funções de ativação usar?
  - Qual learning rate usar?
  - Qual função de custo usar?
- **Muitos dados**
  - São necessárias muitas amostras de dados para treinar um autoencoder que dê bons resultados

- **Chance de overfitting**
  - Quando o conjunto de dados de treinamento é limitado em número de amostras distintas, pode haver overfitting

Como a área de inteligência artificial, em particular nas construções de redes neurais, carece de teoremas, muitas questões envolvem escolhas a serem tomadas. Implementar um modelo de autoencoder requer algum nível de "força bruta"e "artesanato", para encontrar, por exemplo, a melhor combinação de hiperparâmetros que maximizem sua performance para o objetivo em mente.

6 Referências

Stuart Russell, Peter Norving - Artificial Intelligence: A Modern Approach, Capítulo 22, Seção 7.1 - Unsupervised Learning - Autoencoders

Ian Goodfellow, Yoshua Bengio, Aaron Courville - Deep Learning, Capítulo 14 - Autoencoders

K. Berahmand et al. - Autoencoders and their applications in machine learning: a survey

P. Li, Y. Pei, J. Li - A comprehensive survey on design and application of autoencoder in deep learning

Elad Plaut - From Principal Subspaces to Principal Components with Linear Autoencoders

Mark A. Kramer - Nonlinear Principal Component Analysis Using Autoassociative Neural Networks

J. Karhunen e J. Joutsensalo - Generalizations of principal component analysis, optimization problems, and neural networks

P. Baldi e K. Hornik - Neural networks and principal components analysis: Learning from examples without local minima

H. Bourlard e Y. Kamp - Auto-association by multilayer perceptrons and singular value decomposition