

Autoencoders

Advanced Institute for Artificial Intelligence – AI2

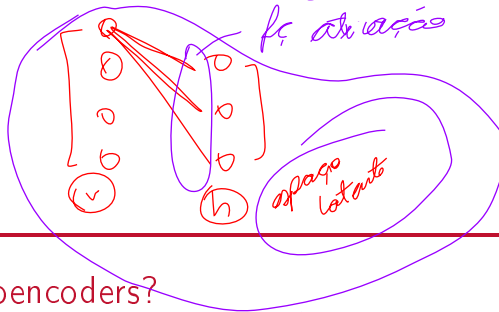
<https://advancedinstitute.ai>

- Red Dim
- Feat Learning

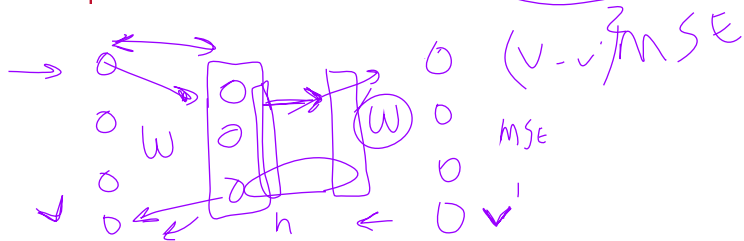
→ class. reg.
fc ativação



Introdução



O que são os Autoencoders?



Algumas definições:

- Algoritmo de aprendizado “não-supervisionado” que aplica retro-propagação de erro, definindo os valores alvo para serem iguais às entradas:
 - emprega $y^{(i)} = x^{(i)}$
- Internamente, possuem uma (ou mais) camada oculta que descreve uma codificação em espaço latente usada para representar os dados de entrada;
- Consistem em duas partes:
 - Função Encoder $h = f(x)$, e
 - Função Decoder que produz a reconstrução $r = g(h)$;
- Projetado para ser **incapaz de aprender a copiar perfeitamente os dados**;

Question

Por que simplesmente aprender a definir $g(f(x)) = x$ em todos os lugares não é especialmente útil?

Algumas definições:

- Caso especial de redes feedforward, e podem ser treinados com as mesmas técnicas, ou seja, descida de gradiente com minibatch, retro-propagação de erro, etc.;
- Geralmente é restrito a copiar apenas aproximadamente, ou seja, produzir dados **que apenas se assemelham aos dados de treinamento**.
- O modelo é forçado a priorizar apenas alguns “aspectos” da entrada;
 - Frequentemente aprende propriedades úteis dos dados, por exemplo, características relevantes;
- Tradicionalmente usado para redução de dimensionalidade ou aprendizado de características (feature learning);

Exemplo

Suponha que as entradas x sejam os valores de intensidade de pixel de uma **imagem 10×10** (100 pixels) - $n=100$, e que existam $s_2 = 50$ unidades ocultas na camada L_2 .

Da definição de Autoencoders temos $y \in \mathcal{R}^{100}$. Como existem apenas 50 unidades ocultas, a rede é forçada a aprender uma representação “comprimida” da entrada.

Dado apenas o vetor de ativações de unidades ocultas $a^{(2)} \in \mathcal{R}^{50}$, deve tentar “reconstruir” a entrada de 100 pixels x .

Algumas definições:

- Se houver alguma estrutura subjacente nos dados, por exemplo, alguns dos atributos de entrada estão correlacionados, então este algoritmo **será capaz de descobrir algumas dessas correlações**.
- Essa forma simples de autoencoder provavelmente aprenderá uma representação de baixa dimensão muito semelhante ao PCA, principalmente se não for utilizada quebra de linearidade.
- Pode ser pensado como **algoritmo de compressão de dados**.
- As funções de compressão e descompressão são:
 - 1 específicas de dados,
 - 2 com perdas, e
 - 3 aprendidas automaticamente a partir de exemplos, em vez de projetadas por nós;

$w^T x + b$

ReLU

PCA

var

MSE

Algumas definições:

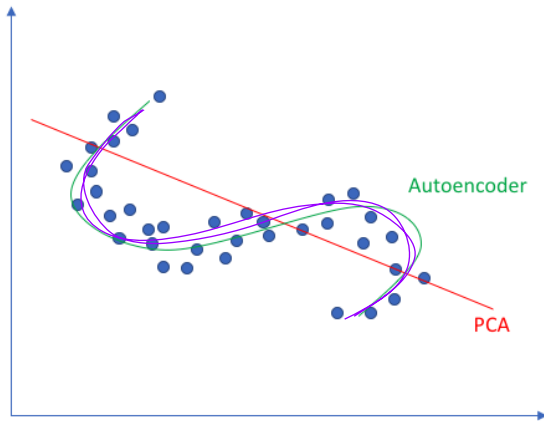



Figura: PCA vs. Autoencoder.

Algumas definições:

Handwritten notes: *MSE* (pointing to the reconstruction loss), *comp.* (pointing to the KL divergence), and *"Undercomplete"* (pointing to the latent space dimension s_2).

$$J_{sparse}(W, b) = \underbrace{J(W, b)}_{\text{reconstruction loss}} + \underbrace{\left(\beta \sum_{j=1}^{s_2} KL(\rho, \rho'_j) \right)}_{\text{sparsity constraint}}, \quad (1)$$


em que:

Handwritten notes: $0.2 \rightarrow \sum r(w \cdot x + b)$ (pointing to ρ), and a diagram of a vertical stack of rectangles with a dashed line and the symbol π (pointing to ρ'_j).

$$\sum_{j=1}^{s_2} KL(\rho, \rho'_j) = \rho * \log \frac{\rho}{\rho'_j} + (1 - \rho) * \log \frac{1 - \rho}{1 - \rho'_j}, \quad (2)$$

ρ é a constante de esparsidade, enquanto ρ'_j representa a média das ativações dos neurônios ocultos.

O comportamento da Equação 2:

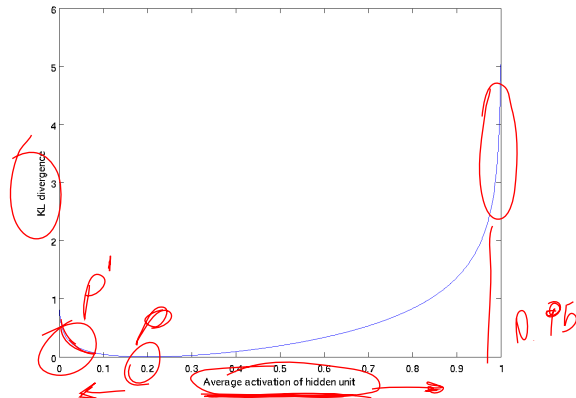



Figura: Comportamento da divergência de KL

Question

- ❶ Por que os Autoencoders não representam algoritmos ótimos de compressão de aplicação geral?
 - ❷ Por que eles precisam ter perdas?
 - ❸ Um autoencoder treinado em imagens de rostos faria um bom trabalho na compressão de imagens de árvores?
- 

Afinal, são bons em?

□ Compressão de dados?

- Quase impossível superar algoritmos padrão, como JPEG, MP3, etc.;
- Podemos **melhorar o desempenho restringindo o tipo de dados** que ele usa;
 - **Perda de capacidade de generalização!**
- Geralmente impraticável para problemas de compressão/compactação de dados do mundo real:
 - Só pode ser usado em dados semelhantes aos que foram treinados.

□ Redução de dimensionalidade:

- Se o decodificador for linear e a função de custo for o MSE, um Autoencoder aprende a abranger o mesmo subespaço que o PCA;

□ Remoção de ruídos

- os dados são parcialmente corrompidos por ruídos;
- o modelo é treinado para prever o ponto de dados original e não corrompido como sua saída;



Autoencoders Incompletos (*Undercomplete*)

Undercomplete Autoencoders

- Autoencoder cuja dimensão de codificação é menor que a de entrada;
- Força o autoencoder a capturar os atributos mais relevantes dos dados de treinamento;
 - Também conhecidos como gargalos *bottlenecks*;
- Minimiza a função de custo, onde f é a função aprendida pelo encoder e g é a função aprendida pelo decoder, ajustando os parâmetros θ e ϕ :

$$J(\theta, \phi) = \frac{1}{n} \sum_{i=1}^n (x^{(i)} - f_{\theta}(g_{\phi}(x^{(i)})))^2$$

Diagram illustrating the cost function $J(\theta, \phi)$ for an undercomplete autoencoder. The formula shows the reconstruction error as the squared difference between the input $x^{(i)}$ and the reconstructed input $f_{\theta}(g_{\phi}(x^{(i)}))$. Handwritten annotations include: a red circle around $J(\theta, \phi)$, a red circle around $\frac{1}{n}$, a red circle around the summation index $i=1$, a red circle around $x^{(i)}$, a green circle around $f_{\theta}(g_{\phi}(x^{(i)}))$, a purple arrow pointing to f_{θ} labeled "encode", and a green arrow pointing to g_{ϕ} labeled "decoder". A red bracket is drawn under the entire expression.

- As funções de encoders não linear f , assim como as de decoders g podem aprender uma generalização não linear mais poderosa que o PCA;

Undercomplete Autoencoders

Implementação

□ Arquitetura Geral:

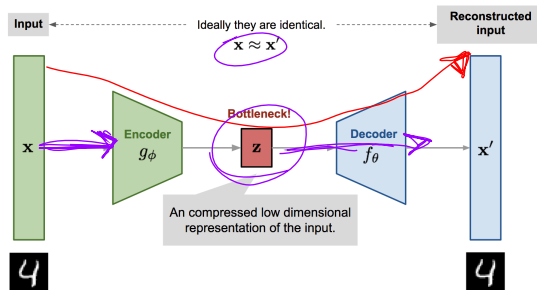


Figura: Arquitetura geral de um autoencoder undercomplete

Fonte: *Autoencoder architecture by Lilian Weng*

Implementação

- Redução de dimensionalidade para dados 3D

```
1 encoding_dim = 2
2 input_layer = keras.Input(shape=(3,))
3 encoded = layers.Dense(encoding_dim, activation="sigmoid")(input_layer)
4 decoded = layers.Dense(3, activation="sigmoid")(encoded)
5 autoencoder = keras.Model(input_layer, decoded)
6 autoencoder.compile(loss="mse", optimizer="SGD")
```

Implementação

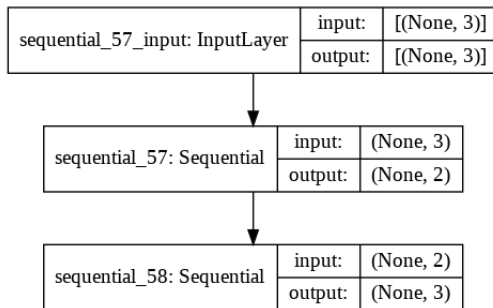


Figura: Arquitetura de um autoencoder

Undercomplete Autoencoders

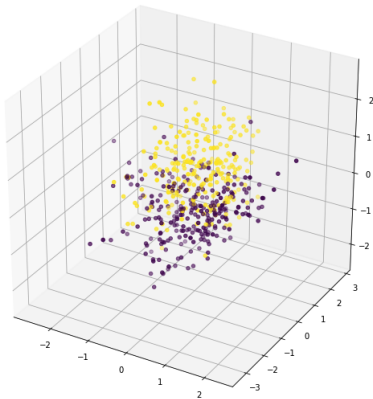


Figura: Dados em 3D antes do encoding

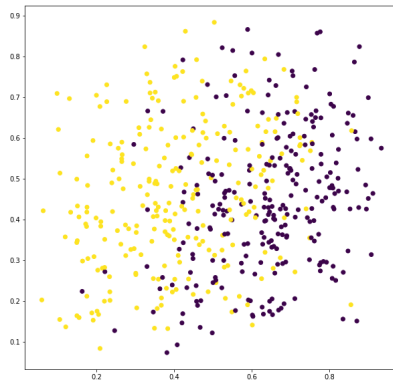
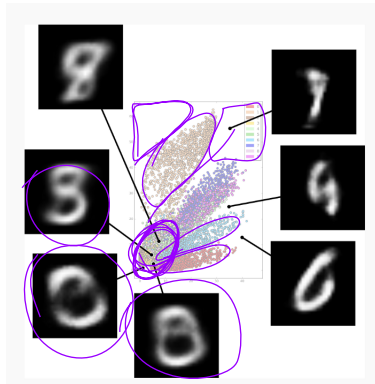


Figura: Dados em 2D depois do encoding

Undercomplete Autoencoders

No que eles não são bons?



784 \rightarrow 2
(28,28)

Figura: Representação do espaço latente para o dataset MNIST.

No que eles não são bons?

- Alguns dos maiores desafios em relação ao espaço latente são:
 - **Lacunas no espaço latente**: não sabemos como podem ser os pontos de dados nesses espaços
 - **Separabilidade no espaço latente**: também existem regiões onde o rotulado é intercalado/espalhado aleatoriamente
 - **Espaço latente discreto**: não temos um modelo estatístico treinado para uma entrada arbitrária

Limitações

- Infelizmente, autoencoders incompletos **não aprendem nada útil** se o encoder e o decoder **receberem muita capacidade**
- Também ocorre se o espaço latente tiver a mesma dimensão que a entrada;
- O mesmo acontece no caso overcomplete, onde o espaço latente tem dimensão maior que a entrada;
- Mesmo um encoder e um decoder linear podem **aprender a copiar a entrada** para a saída;
 - **Nada de útil é aprendido sobre a distribuição de dados.**



Regularized Autoencoders

Regularized Autoencoders

custo

- Usa uma função de perda que “encoraja” o modelo a ter outras propriedades, por exemplo, esparsidade da representação e robustez a ruídos, ou a entradas ausentes;
- Pode ser não-linear e overcomplete, mas ainda assim aprender algo útil sobre a distribuição de dados;
- Os dois Autoencoders Regularizados mais comuns são:
 - **Sparse Autoencoders:** penalidade de esparsidade adicionada à sua função de custo original; \rightarrow Eq. 1
 - **Denoising Autoencoders:** adicionar ruído (gaussiano, por exemplo) às entradas, forçando nosso modelo a aprender atributos/características importantes;

Sparse Autoencoders

- Basicamente é um autoencoder cujo critério de treinamento envolve a esparsidade como termo de penalização;
- Apresenta uma dimensão latente maior que as dimensões de entrada ou saída;
- Normalmente usado para **aprender características para outra tarefa**, como classificação;
- Pense na penalidade simplesmente como um **termo regularizador**;
- Gostaríamos de restringir os neurônios a serem **inativos a maior parte do tempo**;
- Reduzir a propensão de overfitting da rede;
- Ele pode **não mais copiar a entrada através de certos neurônios**:
 - em cada execução, **esses neurônios podem não ser os ativos**

Sparse Autoencoders

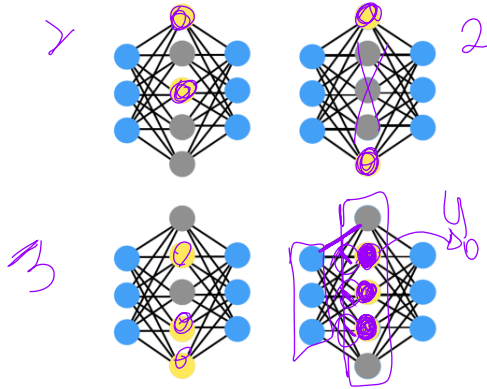


Image by Shreya Chaudhary

Sparse Autoencoders - Implementation

No Keras, isso pode ser feito adicionando um regularizador com uma `activity_regularizer` à camada Dense

```
1 encoded = layers.Dense(encoding_dim, activation='relu',  
2 activity_regularizer=regularizers.l1(10e-5))(input_img)
```

Com a regularização adicionada o modelo tem menos probabilidade de overfitting e **pode ser treinado por mais tempo**;

Denoising Autoencoders

gauss
branco
sal p.m

Cut out
int.off



- A entrada é parcialmente corrompida pela adição de ruídos, ou “mascaramento” de alguns valores do vetor de entrada de maneira estocástica;
- O modelo é treinado para recuperar a entrada original (nota: não a corrompida);

$$\tilde{\mathbf{x}}^{(i)} \sim \mathcal{M}_{\mathcal{D}}(\tilde{\mathbf{x}}^{(i)} | \mathbf{x}^{(i)})$$
$$J(\theta, \phi) = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}^{(i)} - f_{\theta}(g_{\phi}(\tilde{\mathbf{x}}^{(i)})))^2$$

onde $\mathcal{M}_{\mathcal{D}}$ define o mapeamento das amostras de dados verdadeiras para as ruidosas ou corrompidas.

Denoising Autoencoders

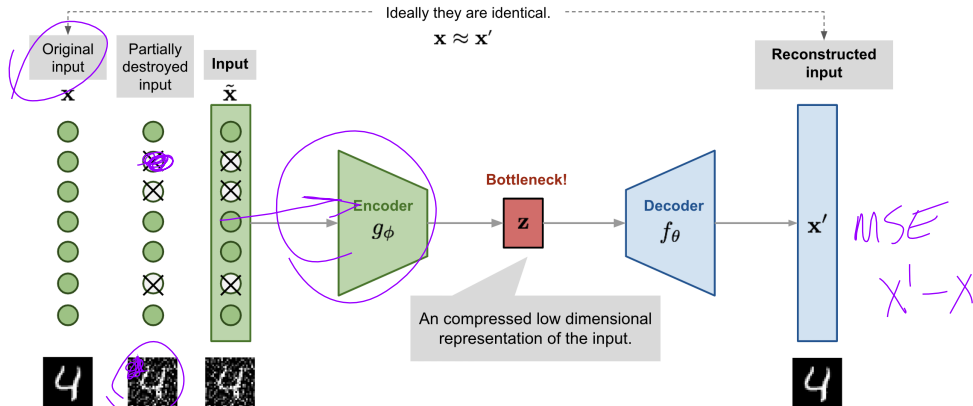


Image by Lilian Weng

Denoising Autoencoders

- Design motivado pelo fato de que os humanos podem reconhecer facilmente um objeto mesmo com a visão parcialmente ocluída;
- Para “reparar” a entrada, o DAE deve descobrir a relação entre as dimensões da entrada para inferir as partes que faltam;
- Em imagens, é provável que o modelo dependa de evidências coletadas de uma combinação de muitas dimensões de entrada para recuperar a versão sem ruído;
 - Isso cria uma boa base para aprender uma representação latente robusta;
- No paper original do DAE, uma proporção fixa de dimensões de entrada é selecionada aleatoriamente e seus valores são forçados a 0 (o mesmo que o dropout?);

Denoising Autoencoders - Implementation

Exemplo para o banco de dados MNIST:

```
1 noise_factor = 0.5
2 x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0,
    scale=1.0, size=x_train.shape)
3 x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=
    1.0, size=x_test.shape)
```

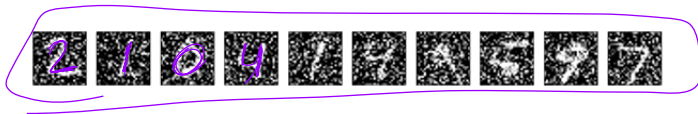


Figura: Dígitos do MNIST após adição de ruído.