Prática: Modelos Generativos (III)

Gabriel Vieira da Cruz

9 de Outubro de 2024

Variational Auto-Encoders (VAE's)

Auto-Encoders são um tipo de rede neural amplamente utilizado para a geração de dados. Eles também têm aplicações em compressão de dados, redução de dimensionalidade, colorização, e remoção de ruídos. Esse tipo de rede pode ser resumido a dois componentes principais: o Encoder e o Decoder.

Encoders

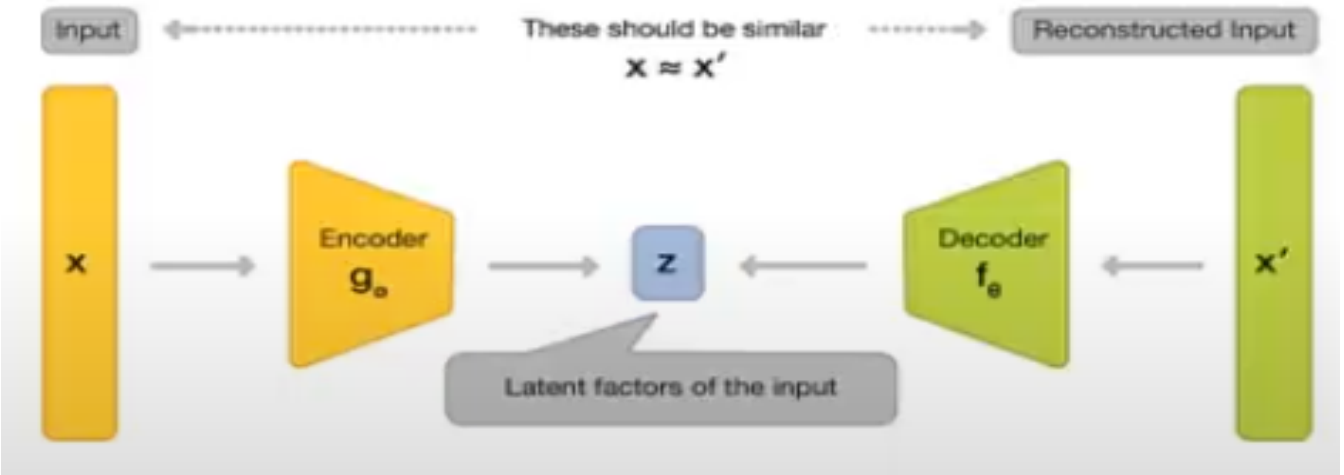
O encoder tem um funcionamento semelhante às Redes Neurais Convolucionais (CNNs). Ele recebe uma imagem como entrada e, através de uma série de convoluções, simplifica essa entrada, destacando atributos latentes. A ideia é que a rede neural aprenda a generalizar o conjunto de dados de treinamento, mapeando os dados de entrada para uma representação comprimida no espaço latente.

Decoders

O decoder realiza o processo inverso do encoder. A partir dos atributos latentes recebidos como entrada, a rede busca reconstruir o dado original. Para isso, utiliza a convolução transposta (ou deconvolução), que executa a operação inversa da convolução, expandindo ou "preenchendo" a informação comprimida para gerar a saída.

Resumidamente, enquanto o encoder reduz o dado de entrada, o decoder o expande para reconstruir o formato original.

Resumidamente, enquanto um encoder reduz o dado de entrada, o decoder aumenta/preenche.



$X \approx X'$

Sendo:

X: $Imagem/dado\ original\ dado\ como\ entrada\ no\ Encoder$

X': $Imagem/dado\ gerado\ pelo\ Decoder$

Os VAE's funcionam de forma diferente dos autoencoders tradicionais, em vez de apenas codificar o dado de entrada em um vetor latente, eles o fazem de forma probabilística. Os atributos latentes não são fixos, mas sim representados por uma distribuição de probabilidade que normalmente é uma distribuição Gaussiana descrita por sua média e variância.

Isso significa que, em vez de mapear diretamente um ponto específico no espaço latente, o VAE **aprende os parâmetros de uma distribuição** de onde podemos extrair uma amostra para gerar novos dados. Isso permite ao VAE não só reconstruir entradas, mas também gerar novos exemplos semelhantes aos dados de treinamento.

Reparametrization Trick

Os VAE's introduzem uma etapa de amostragem estocástica no espaço latente, o que significa que o processo envolve aleatoriedade. Isso cria um desafio para o uso de métodos de otimização baseados em gradiente (como o backpropagation) utilizados no treinamento de redes neurais. O problema é que a operação de amostragem não é diferenciável, o que impede a propagação normal dos gradientes para ajustar os pesos da rede.

Para resolver esse problema, a técnica propõe reestruturar o processo de amostragem para torná-lo diferenciável. Isso é feito ao utilizar os atributos da distribuição latente (a média e o desvio padrão) como parâmetros durante o treinamento da rede. Dessa forma, a amostragem estocástica é separada em uma parte determinística (os parâmetros da distribuição) e uma parte estocástica fixa (o ruído), permitindo que o gradiente flua durante o treinamento.

$z = \mu + \sigma \cdot \epsilon$

Sendo:

z: $Vetor\ latente\ extraído$

μ: $Média$

σ: $Desvio\ padrão$

ϵ: $Vetor\ de\ ruído$

Kullback-Leibler Divergence (KL Divergence)

O KL Divergence é uma métrica utilizada como **"loss function"** para medir a diferença entre a distribuição entre a distribuição latente aprendida e a distribuição normal padrão. Basicamente força a rede a aprender uma distribuição latente que seja o mais próximo possível da distribuição normal padrão, devido a sua simetria e regularidade na distribuição dos dados.

Generative Adversarial Networks (GAN's)

O GAN é um modelo de Inteligência Artificial generativa que treina duas redes neurais para gerar novos dados. É a tecnologia por trás dos *deepfakes*, mas também possui

aplicações mais úteis como: Geração de dados sintéticos para remover informações privadas, detecção de anomalias, *self-driving*, geração de arte e música, etc…

Mas diferente do VAE, o GAN não assume que a distribuição dos dados reais seja gaussiana, mesmo sendo mais conveniente na maioria dos casos.

Gerador

O gerador é uma das duas redes neurais do modelo. Ele recebe como entrada um ruído aleatório e busca mapeá-lo em uma distribuição de probabilidade, mantendo um certo nível de aleatoriedade na saída. Basicamente **aprende a como gerar o mesmo tipo de dado do input**.

Discriminador

O discriminador aprende a identificar dados reais de dados sintéticos (retornados pelo gerador).

Funcionamento

A ideia por trás do modelo, é que o gerador esteja tentando “enganar” o discriminador e o discriminador tentando “flagrar” o gerador, de forma que ambos estejam treinando um ao outro.

1. O gerador recebe um ruído, como se fosse uma *seed* que vai influenciar na saída, e, utilizando camada(s) de deconvolução, retorna um dado sintético que tenta imitar os dados reais.
2. O dado gerado é concatenado com dados reais e é utilizado como entrada no discriminador.
3. O discriminador deve diferenciar os dados falsos/sintéticos dos dados reais, fazendo uso de camada(s) de convolução.
4. Após o discriminador classificar as amostras, calcula-se a função de perda para ambas as redes.
5. O gradiente da perda é retropropagado. Para o gerador, isso significa que ele ajusta seus pesos com base no quão bem ele conseguiu enganar o discriminador.

Min-Max game

Min-Max game é o nome dado à métrica de perda do modelo. Recebe esse nome devido à influência de ambas as redes neurais na métrica:

- O gerador minimiza o valor da métrica, criando imagens cada vez mais realistas.
- O discriminador aumenta o valor, aumentando sua capacidade de identificar imagens falsas.

Mode Collapse

Muitas vezes, ao longo do treinamento, o gerador pode se tornar cada vez mais específico nos dados gerados, isso significa que ele vai perdendo sua capacidade de diversidade de amostras. Como consequência, o gerador tem uma produção monótona de dados (parecem pertencer à mesma classe).

Existem algumas estratégias para lidar com esse problema, segue algumas delas:

- **Ajustes na taxa de aprendizado:** Uma menor taxa de aprendizado para o gerador pode ajudar a prevenir que ele “aprenda” rapidamente uma solução fácil.
- **Treinamento alternado:** Alternar as atualizações do gerador e do discriminador com mais frequência pode ajudar a manter um equilíbrio dinâmico entre as duas redes.
- **Variedade no ruído:** Garantir que o vetor de ruído dado como entrada seja suficientemente variado para estimular o gerador a produzir uma maior gama de amostras.
- **Arquiteturas modificadas:** Arquiteturas como **WGAN,** modificam a função de perda e a maneira como o discriminador opera.
- **Outras técnicas avançadas:** mini-batch discrimination, historical averaging, feature matching, etc…

Generative Pre-trained Transformer

A descrição desse modelo pode ser feita através da etimologia do termo:

- **Generative:** Refere-se ao fato de que o modelo é capaz de gerar texto com base em padrões que aprendeu.
- **Pre-trained:** Modelo é treinado inicialmente com uma grande quantidade de dados de texto. Assim ajuda o modelo a “aprender” o básico da linguagem como gramática, contexto e até mesmo conhecimento geral.
- **Transformer:** Arquitetura neural do modelo. Baseado no mecanismo de atenção (self-attention), permite ao modelo focar em diferentes partes de uma sequência ao processá-la.

Tokenização

Ao receber os dados de texto para treinamento, o modelo faz o processo de tokenização com esses dados, que é efetuado principalmente para a quantificação individual dos tokens (representando-os como sendo vetores) e é definido o block_size, que indica o número de tokens que será utilizado por passo de treinamento.

Normalmente são utilizados 3 tipos de tokenização, sendo eles:

- **Tokenização por Palavras**: Cada palavra é um token.
- **Tokenização por Subpalavras**: Palavras são divididas em partes menores (subpalavras), o que ajuda a reduzir o número de tokens.
- **Tokenização por Caracteres**: Cada caractere é um token. Útil em certos idiomas e para modelos que precisam lidar com palavras novas.

Muitos recursos e/ou técnicas já estabelecidos em NLP podem ser utilizados para se ajustar melhor às preferências do modelo.

Self-Attention

O modelo recebe uma sequência de tokens, e, considerando os dados de treinamento, ele busca prever o token mais adequado para dar continuidade à sequência.

Mas para apresentar resultados promissores e mais próximos da linguagem natural, é necessário que o histórico (todos os tokens da sequência) sejam considerados na previsão.

Nessa etapa também é possível utilizar de recursos e/ou técnicas de NLP para melhorar a métrica da previsão (considerar informações gramaticais e sintaxe, por exemplo).