Relatório LAMIA 27
Prática: Modelos Generativos (III)

Kaique Medeiros Lima

1 Introdução

Nesse card, é discutido sobre técnicas que possibilitam uma visão concisa de técnicas avançadas em redes neurais para geração e reconstrução de dados. São abordados conceitos de autoencoders, VAEs e GANs, com ênfase nos fundamentos matemáticos e nos desafios do treinamento desses modelos.

2 Descrição da atividade

2.1 Generative Models

2.1.1 Auto-Encoders

Os autoencoders são redes neurais projetadas para aprender representações compactas dos dados. Eles possuem duas partes:

1. Encoder: Reduz os dados de entrada a um conjunto de características latentes (usando técnicas como convoluções).

2. Decoder: Reconstrói os dados a partir dessas características latentes (usando transposed convolutions).

O objetivo é minimizar a diferença entre a entrada e a saída reconstruída. Depois de treinado, o encoder pode ser descartado e o decoder usado para gerar novos dados sintéticos. Eles são aplicados em redução de dimensionalidade, compressão, busca, remoção de ruído e colorização.

2.1.2 Transpose Convolution

Os autoencoders utilizam camadas Conv2DTranspose no decoder para reconstruir imagens a partir das características latentes. Essas camadas aprendem pesos para criar novos pixels a partir de representações de menor dimensão. Um stride de 2 é frequentemente usado, e há a possibilidade de utilizar max-unpooling (o inverso do max-pooling). Essencialmente, o decoder funciona como uma CNN reversa, transformando características latentes em imagens maiores e detalhadas.

2.1.3 Variational Auto-Encoders (VAE)

Nos VAEs, os vetores latentes não são valores fixos, mas distribuições de probabilidade. Cada ponto no espaço latente é representado por uma média e uma variância de uma distribuição Gaussiana. O processo envolve duas etapas:

1. Inferência: Estimar $p(z \mid X)$, ou seja, obter a distribuição latente a partir dos dados de entrada.
2. Geração: Amostrar z dessa distribuição e gerar $p(X \mid z)$, ou seja, reconstruir os dados.

Esse conceito inspirou o desenvolvimento de redes adversariais generativas (GANs), pois permite gerar novas amostras de forma mais realista.

2.1.4 The "Reparameterization Trick"

Um problema dos VAEs é que a amostragem direta de uma distribuição de probabilidade não pode ser diferenciada, dificultando o aprendizado via backpropagation. Para resolver isso, usa-se o truque da reparametrização:

- Em vez de amostrar z diretamente, reescrevemos como

$$Z = \mu + \sigma \cdot \epsilon,$$

  onde ϵ segue uma distribuição normal padrão.

Isso permite que a parte estocástica (ϵ) seja separada do modelo treinável, tornando o processo diferenciável. Essa abordagem transforma o processo em um grafo computacional contínuo, permitindo a retropropagação funcionar corretamente.

2.1.5 Kullback-Leibler Divergence

Para medir a diferença entre a distribuição inferida $q(z \mid x)$ e a verdadeira distribuição latente $p(z)$, usamos a divergência de Kullback-Leibler (KL):

$$KL(p \,\|\, q) = \sum p(x) \log \left(\frac{p(x)}{q(x)} \right).$$

Ela pode ser vista como uma medida de "distância" entre distribuições de probabilidade. No contexto dos VAEs, serve como parte da função de perda para garantir que a distribuição inferida não se desvie muito da distribuição prior imposta. Em outras palavras, ajuda o modelo a aprender uma representação latente bem estruturada e contínua.

2.1.6 Generative Adversarial Networks (GANs)

As Redes Adversariais Generativas (GANs) é o tipo de tecnologia que ficou famosa por permitir a criação de deepfakes, além de aplicações virais como troca de rostos e envelhecimento artificial. No entanto, o propósito original da pesquisa era bem mais amplo e inclui usos como:

- Geração de conjuntos de dados sintéticos para remover informações privadas e aumentar a privacidade.

- Detecção de anomalias, auxiliando em áreas como segurança cibernética e diagnóstico médico.
- Aplicações em veículos autônomos, ajudando a treinar IA para navegação em ambientes variados.
- Criação de arte e música, possibilitando novas formas de expressão criativa com algoritmos.

GANs funcionam com dois modelos treinados em conjunto:

1. Gerador: Cria imagens (ou outros dados) tentando enganar o discriminador.
2. Discriminador: Tenta distinguir entre imagens reais e geradas.

2.1.7 GAN's (Generative Adversarial Networks)

As GANs aprendem a distribuição real dos vetores latentes sem assumir que seguem uma distribuição Gaussiana, como nos VAEs. O funcionamento básico envolve dois componentes principais:

1. Gerador (Generator)
    - Converte ruído aleatório em uma distribuição de probabilidade que representa os dados reais.
    - Tenta criar imagens realistas para enganar o discriminador.
2. Discriminador (Discriminator)
    - Aprende a diferenciar imagens reais das imagens geradas.
    - Está constantemente tentando pegar o gerador na mentira.

Como o nome sugere, gerador e discriminador têm um relacionamento adversarial. O objetivo final é que o gerador fique tão bom que o discriminador não consiga mais distinguir entre as imagens reais e as falsas. Nesse ponto, a GAN está "pronta" (pelo menos, em teoria).

2.1.8 Fancy Math — A Matemática por Trás das GANs

A equação apresentada é a função de perda adversarial das GANs, baseada em um jogo de min-max entre o gerador e o discriminador:

$$\min_G \max_D V(D,G) = \mathrm{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathrm{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))].$$

Isso significa que:

- O gerador (G) tenta minimizar essa função de perda, melhorando a qualidade das imagens sintéticas para enganar o discriminador.
- O discriminador (D) tenta maximizar essa função, aprimorando sua capacidade de distinguir imagens reais das geradas.

Chamamos esse processo de um "min-max game", onde:

- O gerador reduz sua perda tornando suas imagens mais realistas.
- O discriminador melhora sua detecção de falsificações.

2.1.9 Desafios no treinamento de GANs

Treinar uma GAN não é trivial, pois o processo é instável e envolve muitos ajustes finos. Alguns desafios comuns incluem:

- Alto grau de tentativa e erro na escolha de hiperparâmetros.
- Mode collapse, onde o gerador aprende a produzir apenas algumas poucas imagens repetidas.
- Vanishing gradients, quando os gradientes ficam muito pequenos, dificultando o aprendizado.

2.2 Let's build GPT: from scratch, in code, spelled out.

O vídeo começa explicando os fundamentos dos Transformers e como os modelos GPT evoluíram até chegarmos ao ChatGPT e ao GitHub Copilot, destacando o papel dos modelos de linguagem autoregressiva e a importância do PyTorch. Em seguida, ele mostra como preparar os dados – desde a tokenização e divisão entre treino e validação, até a criação de batches – e introduz um modelo simples de linguagem baseado em bigramas, passando por treinamento e geração de texto.

Depois, o foco se volta para a construção do mecanismo de self-attention. A explicação parte de uma versão básica com loops, evoluindo para implementações mais eficientes usando multiplicação de matrizes, softmax e o escalonamento necessário para uma melhor performance. Por fim, o vídeo demonstra como montar um Transformer completo, abordando multi-head self-attention, camadas feedforward, conexões residuais e layer normalization, além de discutir rapidamente as diferenças entre encoder, decoder.

3 Conclusão

Nesse card, decorre-se, de forma prática e teórica, as técnicas por trás dos autoencoders, VAEs e GANs. Embora o treinamento e a complexidade dos modelos representem desafios reais, essas abordagens abrem caminho para inovações surpreendentes na inteligência artificial.