MAC05921 – *Deep Learning*

DCC / IME-USP — 2023

Tarefa 3

Data de entrega: 15/11/2023

O objetivo desta tarefa é o exercício prático de conceitos, técnicas e métodos vistos ou relacionados ao conteúdo coberto nas aulas. Na Task 2 exploramos segmentação semântica de imagens com arquiteturas do tipo *encoder-decoder*, particularmente a U-Net. Neste EP, o foco será a exploração de algumas arquiteturas do tipo *autoencoder*.

Descrição da tarefa

Implementar e experimentar arquiteturas de redes do tipo *autodecoder*, listadas a seguir. O objetivo é consolidar o entendimento sobre o funcionamento das mesmas, mas principalmente ganhar intuição/conhecimento sobre as diferenças/evoluções de uma para a outra. Para tanto, escolham algum dataset não muito complexo (pode ser o MNIST[1]). Usem e abusem de recursos de visualização (de imagens, dos pontos no espaço latente, ...)!

Auto-encoder: é uma arquitetura típica usada principalmente para redução de dimensão dos dados. Segue a estrutura *encoder-decoder*, tendo a particularidade de a entrada e a saída *target* serem as mesmas. Por isso é muitas vezes considerada uma rede para redução de dimensionalidade ou compressão de dados. O que se tenta otimizar é a reconstrução do dado de entrada.

Denosining auto-encoder: é basicamente um *auto-encoder*, com a diferença de que os dados de entrada são ligeiramente perturbados com ruídos aleatórios. Além de verificar que a rede "aprende" a filtrar ruídos, vale a pena verificar se existe alguma diferença em relação ao *auto-encoder* anterior, quando avaliamos o desempenho deste em relação a entradas sem perturbação com o desempho do anterior.

Variational auto-encoder: Também segue o mesmo princípio do *auto-encoder* em termos de entrada e saída, porém o principal foco neste modelo é fazer com que o espaço latente (a representação comprimida gerada entre a parte *encoder* e *decoder*) seja contínua (isto é, pequenas variações de uma variável latente devem refletir variações correspondentes na reconstrução). Para tanto é utilizada uma função de perda modificada de forma que o espaço latente modele a distribuição dos dados. Assim, pode-se gerar dados reconstruindo-se instâncias quaisquer do espaço latente. Por essa razão, essa rede muitas vezes é vista como uma rede generativa.

Para a exploração, fiquem à vontade para usar técnicas de visualização, animação, projeção (t-SNE, UMAP), PCA × representação latente, etc. Também pode ser interessante variar a quantidade de exemplos usados no treinamento.

[1] Essas arquiteturas podem ser basedas em redes *fully connected* ou *convolucionais*, aplicadas a dados tabulares ou images, respectivamente. Vocês podem escolher. Mas, as convolucionais são mais interessantes pensando na visualização da reconstrução.

Da mesma forma como foi feito na Tarefa 1 ...

Deve ser criado um notebook Python. Para a implementação das redes, pode ser usado o Keras/TensorFlow ou PyTorch. Para treinar a rede e fazer predições, é relevante termos acesso a GPUs. Recomendamos o uso do Google Colab.

Deve ser entregue um *notebook* e o correspondente `html` incluindo os *outputs* da execução. No topo do *notebook*, coloque sua identificação (nome) e um resumo do que está presente no *notebook*. Note que a implementação não precisa ser *from scratch*; pode ser baseada em implementação disponível na web[2].

Também deve ser informado:

- Todas as fontes utilizadas: *notebooks* de terceiros, páginas consultadas e que efetivamente serviram como referência, colegas, etc,
- Dificuldades enfrentadas
- Qualquer coisa positiva associada à execução desta tarefa

Essas informações podem estar no próprio *notebook* ou podem estar em um `pdf` à parte.

Discussões no fórum assim como em sala de aula serão apreciados!

[2]Espera-se, porém, uma reprodução consciente e reformulada/melhorada/ampliada/complementada da implementação-base utilizada, quando for o caso.